IxTeT: an Integrated Approach for Plan Generation and Scheduling

Philippe Laborie Malik Ghallab

authors:	**Philippe Laborie**	**Malik Ghallab**
E-mail:	laborie@laas.fr	malik@laas.fr
postal adress:	LAAS-CNRS 7, Avenue du Colonel Roche 31077 Toulouse Cedex, France	
fax number:	(33)61.33.64.55	

IxTeT: an Integrated Approach for Plan Generation and Scheduling

Philippe Laborie   Malik Ghallab

Abstract

In a traditional approach to solve complex planning problems, plan generation and scheduling are addressed at two different steps of the resolution: plan generation answers the question "what to do?" and scheduling, the question "how to do it?". Recent works in plan generation as well as in scheduling tend to draw these two problems together claiming that the two questions are very interdependent. In this context, this paper describes a planning system, called IxTeT, that handles abreast and in an opportunistic way plan generation and scheduling. Thanks to a rich formalism to represent time, resources, state of the world and changes, our planning system can address a large panel of realistic problems. The search relies on the resolution of flaws on the current plan; it is controlled by a least-commitment opportunistic strategy coupled with an original abstraction hierarchy. A heuristic search driven by a near-admissible algorithm ensures at the same time the global efficiency of the system and the good quality of the solution in terms of flexibility.

Keywords

Temporal planning, Plan generation, Scheduling, Resource management, Opportunistic search, Least-commitment strategy, Abstraction hierarchy.

Submission area

Planning and Scheduling

IxTeT: an Integrated Approach for Plan Generation and Scheduling

Philippe Laborie Malik Ghallab

1 Introduction

On the necessity to integrate plan generation and scheduling

A dictionary defines the word **"Planning"** as "elaborating a project containing an ordered set of operations in order to achieve a goal"; and as "deciding **what** you are going to do and **how** you are going to do so".
Because of the large scope and complexity of the planning process, its automation is usually decomposed into two different steps:

1. **Plan generation** which aims at selecting the operators to achieve the goal; it corresponds to the question "**what** to do ?". Its result is a **plan**, that is a partially ordered set of operators; and
2. **Scheduling**, whose purpose is to organise the selected operators in order to satisfy some resource constraints of the environment and corresponds to the question "**how** to organise oneself to do it ?". The result, then, is a **schedule** that, if executed, will ensure the goal resolution while satisfying the physical constraints of the environment.

Until a few years ago, one of the most famous plan generation benchmark was the blocks-word domain, a very easy problem (for a human) with a very limited practical interest whereas one of the most studied problem in scheduling was the job-shop problem which is linked to a lot of industrial applications. These two examples are very revealing about the nature of the research efforts in these fields and about the nature and the complexity of these two problems in general.

Plan generation is a very complex process; it has been shown in [4] that planning within the context of the STRIPS formalism is a PSPACE-complete problem. Over the past few years, nevertheless, the algorithmic basis of classical plan generation have been well clarified, e.g. in [5] and [23] and many results have shown that, although the complexity is inherent in the problem, some techniques like *macro-operators*, *abstraction hierarchies* or *least commitment* can help saving a lot of unuseful search efforts [20][19][30].
Concurrently with these works based on the STRIPS formalism, there has been many investigations to extend this classical representation to a richest one in order to solve more realistic problems. DEVISER [28] was the first plan generation system to explicitly handle temporal informations. In O-Plan [6], the temporal constraints of the plan are handled by a specific constraint network: the *time map* whose variables are a set of *time-points* and metric constraints are represented by a range $[l, u]$ on the possible delay between two time-points. The time-point based approach is well suited for planning; it has been shown

to be less complex [29] than interval based approaches [1]. Concerning resource management, SIPE was the first planner to handle consumable and producible resources to prune the search space and avoid the exploration of some necessary over-consumer plans [31]. The same idea is used in O-Plan2 to handle sharable resources through a criterion based on optimistic and pessimistic resource profiles [7]. Furthermore, this criterion is used in O-Plan2 to post constraints on the plan to solve some potential resource conflicts that have been detected. This profiles, nevertheless, do not seem to be sufficient to ensure the soundness of the search with respect to resource conflicts in a general least-commitment planner.

Scheduling systems, for their part, have always been concerned with time and resource management. Many scheduling techniques come from Operational Research (branch and bound, simulated annealing, tabu search, lagrangian relaxation, ...) but since the second half of the 80's these conventional techniques are often combined with Artificial Intelligence approaches. The main reasons are that A.I. is much more flexible than O.R. with respect to knowledge representation and that it allows a more "informed" search possibly taking into account domain-specific knowledge [16].
ISIS [9] was one of the first attempt to deal with a large variety of realistic constraints for job-shop scheduling. Other systems like OPIS [26] or MICRO-BOSS [24] introduced the notion of *opportunistic search* that is the ability to dynamically revise the search procedure. In their approach, this consists in choosing first to schedule the most constrained operations i.e. those with the highest contribution to resource bottlenecks. Systems like OPAL [3] or MASCOT (for sharable resources) [8] work out a *constraint based analysis* to detect some necessary features of the solutions represented by the current partial schedule. In these approaches, opportunity of a conflict resolution depends on the number and characteristics of the different ways to solve it. TOSCA, a scheduling system inspired by the architecture and approach of O-Plan2 [2] implements the notion of *least-commitment* in scheduling together with a form of constraint based analysis to select a scheduling flaw.

This brief review of the state of art in planning shows how the frontier between plan generation and scheduling is getting more and more difficult to define: as some advances are made in plan generation to define new representations, algorithms and techniques, there is now a possibility to apply them for solving realistic problems and, as scheduling must handle more and more complex problems and constraints (scheduling with alternative ranges for example), it needs a more flexible set of techniques to solve them (c.f. figure 1). The integration of plan generation and scheduling into a single planning paradigm is a promising and necessary way of research when one considers the limits of the classical approach that dissociates the two processes.

In this paper we describe a planning system, called IxTeT, whose purpose is to address a very large range of problems between pure plan generation (e.g. the blocks-world) and pure scheduling (e.g. the job-shop). IxTeT relies on most of the features presented above mixed with more original ones:

- an explicit representation of time with different types of metric constraints between time-points [15, 14];
- a powerful representation of the world through multi-valuated attributes;
- an original representation of persistence and change of the world thanks to a reified logic formalism [13];

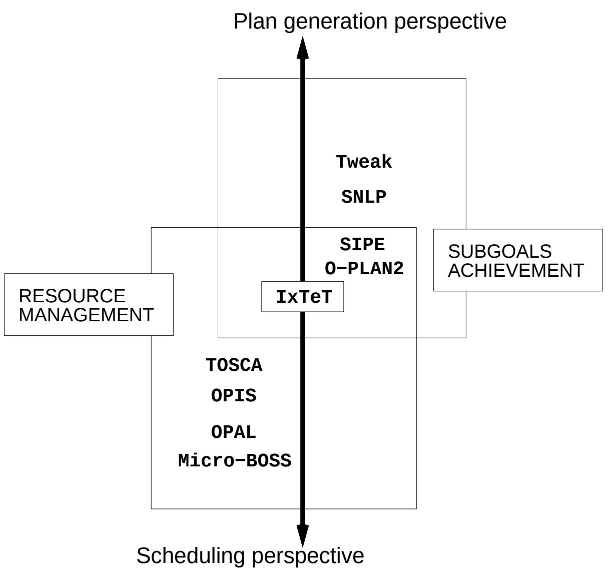

Figure 1: where is the frontier between plan generation and scheduling ?

- the management of a large range of resource types (unsharable, sharable, consumable, producible) [22];
- a task formalism allowing the representation of complex macro-operators;
- an extended least-commitment control strategy;
- an automatically generated dynamical abstraction hierarchy [10]; and
- a general control algorithm that ensures the completeness of the search;

The following section describes our temporal representation of planning operators and problems. The search space and its exploration is discussed in section 3 whereas the last section presents two examples to illustrate the diversity of the problems that can be solved by our system. The paper is a general presentation of the IxTeT approach and an illustration of the range of problems it can address; algorithms and properties are detailled in the cited references.

2 The IxTeT formalism

2.1 Description of the World

Properties of the world are described by a set of **multi-valued state attributes** and a set of **resource attributes**.
Each state attribute describes a particular feature of the world, it is a k-ary mapping, from some finite domains into a finite range, called the **value** of the attribute. Domains and their corresponding possible values are explicitly declared. e.g. the position of a robot:

```
attribute position(?robot){
  ?robot in {robot1, robot2};
  ?value in {RoomM, LabRoom1, LabRoom2, ...};
}
```

We define a resource as *any substance or set of objects whose cost or availability induces constraints on the actions that use them.*
A resource can be a single item (with a unit capacity, we speak then of *unsharable resource*) or an aggregate resource that can be shared simultaneously between different actions

seeing that its maximal capacity is not exceeded. Resources are gathered together into *resource types*: two resources belong to the same type if they can be indifferently used by all the operators of the domain.

```
resource robots(?robot){
  ?robot in {robot1, robot2};
  capacity = 1;
}
```

```
resource paper_on_robot(){
  capacity = 3;
}
```

2.2 Description of Change

IxTeT is based on a reified logic formalism [25] where state attributes are temporally qualified by the predicates *hold* and *event* and resource attributes by the predicates *use*, *consume* and *produce*.

- an assertion $hold(att(x_1, ...) : v, (t_1, t_2))$ asserts the persistence of the value of state attribute $att(x_1, ...)$ to v for each t: $t1 \leq t < t2$.
- $event(att(x_1, ...) : (v_1, v_2), t)$ states that an instantaneous change of value of $att(x_1, ...)$ from $v1$ to $v2$ occured at time t.

A more complete description of the IxTeT formalism concerning events and assertions attributes can be found in [13].
In the same way, resource availability profiles and the use of resources by the different operators are described by means of three predicates:

- $use(typ(r) : q, (t_1, t_2))$ represents a borrowing of an integer quantity q of resource r of type typ on the temporal interval $[t_1, t_2]$;
- $consume(typ(r) : q, t)$ states that a quantity q of resource r will be consumed at time t;
- $produce(typ(r) : q, t)$ represents a production of resource at time t.

As stated in the introduction, the IxTeT *time-map manager* [11, 14] relies on time-points as the elementary primitives. Time-points are seen as symbolic variables on which temporal constraints can be posted. We handle both *symbolic constraints* (precedence, simultaneity) and *numeric constraints* expressed as a bounded interval $[I^-, I^+]$ on the temporal distance between time-points. The *time-map manager* propagates these constraints to ensure the global consistency of the network and answers queries about the relative position of time-points.

Management of atemporal variables is achieved through a *variable constraint manager*. We consider variables ranging over finite sets and propagate *domain restriction*, *equality* and *unequality constraints*. Constraint propagation on atemporal variables is achieved through classical CSP techniques [27].

A planning operator, called a **task**, is a temporal structure composed of:

- a set of sub-tasks;
- a set of events describing the changes of the world induced by the task;
- a set of assertions on state attributes to express the required conditions or the protection of some fact between two task events;
- a set of resource usages; and

- a set of temporal and instantiation constraints binding the different time-points and variables of the task.

Tasks are deterministic operators without ramification effects. We assume a complete representation of the world and of its changes. A task may refer to other sub-tasks, specified with the same syntax. This feature makes the expression of complex macro-operators easier. Here is an example of elementary task (without any sub-task) for a robot in charge of the maintenance of a laboratory (cf §4.1) consisting in putting paper in a machine when it is out of paper:

```
task feed_machine(?machine) (start,end){
  variable ?room;
  place(?machine,?room);
  hold    (position(robot):?room, (start,end));
  event   (machine_state(?machine):(out_of_paper,ok), end);
  consume (paper_on_robot():1, end);
  produce (trunk_size():1, end);
  (end - start) in [00:01:00,00:02:00];}
```

The initial plan $\mathcal{P}_{init}$ is a particular task that describes a problem scenario, that is:

- the initial values for the set of instantiated state attributes (as a set of explained events);
- the expected changes on some contingent state attributes that will not be controlled by the planner (as a set of explained events);
- the expected availability profile of the resources (as a set of uses); and
- the goals that must be achieved (usually, as a set of assertions).

IxTeT allows a total flexibility on the expression of temporal constraints between these elements.

3 Searching for a solution

3.1 The global procedure: Flaws and Resolvers

Our planning system explores a search tree of partial plans whose root is the initial plan and branches represent some tasks or constraints inserted on the current plan in order to solve one of its **flaws**.
Given a partial plan, three kinds of flaws are distinguished. For each flaw, a disjunctive set of **resolvers** is computed. A flaw is always associated with a state or a resource attribute name.

- *pending subgoals* are those events or assertions that have not yet been established; resolvers of a pending subgoal consist in inserting an establisher event and an assertion that protects the state attribute value from the establisher to the pending subgoal (causal-link); such an establisher can already be in the plan (simple establishment) or it may need the insertion of a new task (task insertion);
- *threats* are event or assertion that can threaten a given assertion; when a threat is detected, it can be solved by posting temporal constraints (promotion, demotion) or variable constraints (separation, fusion of two assertions);
- *resource conflicts* are a set of resource usages (produce, use or consume) that could be conflicting in some instantiation of the current plan. Resource conflicts are detected as **minimal critical sets** thanks to an efficient algorithm furtherly described in [21]. Resolvers of a resource conflict include precedence constraints, unequality constraints between resources to allocate and insertion of resource production tasks.

The system architecture is described on figure 2. Three analysis modules are dedicated to the detection of the different flaws and the computing of their resolvers at a given node of the search tree. The control module executes the following non-deterministic algorithm:

```
        **Algorithm** $plan(\mathcal{P})$
1. **termination:** if $\mathcal{P}$ does not contain any flaw: return P
2. **analysis:** call the three flaw analysis modules:
   $FLAWS = pending\_subgoals(\mathcal{P}) \cup threats(\mathcal{P}) \cup resource\_conflicts(\mathcal{P})$
3. **flaw selection:** $\Phi \in FLAWS$;
4. **choice of a resolver:** $\rho \in resolvers(\Phi)$;
5. **recursive invocation:** $plan(insert(\mathcal{P}, \rho))$
```

One originality of our approach is that the selection of the current flaw is independent of its type, be it a pending subgoal, a threat or a resource conflict. This allows a complete integration of the processes of plan generation and scheduling as some opportunistic scheduling decisions are taken before the whole plan is generated.
As long as the complete panel of resolvers is computed for a given flaw and the choice of a resolver is a pending choice in the search tree on which a backtrack can occur, our system is complete in the sense that if there exists a solution, it will find it.

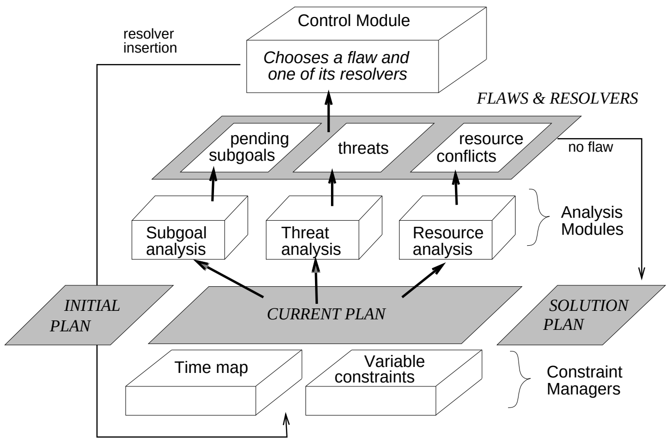

Figure 2: *the IxTeT Planner Architecture*

3.2 Non-deterministic choices: Abstraction and Least-Commitment

The global procedure given above shows two choosing points:

- **line 3,** selection of a flaw: this selection is very important for the global efficiency of the system; it corresponds to the *variable ordering heuristics* in CSP. It is a point at which backtracking is not needed for the completeness of the algorithm.
- **line 4,** choice of a resolver: this is a real non-deterministic choice, defining a possible backtrack point similar to the *value ordering heuristics* in CSP.

For a given flaw, the choice of one of its resolver is done according to a **least-commitment** strategy: the best resolvers are those that do not over-constrain too much the current plan. The **commitment** of posting resolver ρ on the current partial plan $\mathcal{P}$, is a local estimation of the ratio of instantiations of $\mathcal{P}$ that are eliminated by the posting of ρ [22].

The flaws are selected in an **opportunistic** way: priority is given to the flaws that maximise the easiness to make a choice between its resolvers. An opportunity criterion estimates how far the best resolver is "ahead" of the others.
In a partial plan, the number of flaws to analyse may be very large and, at a given level of the planning process, some flaws may be much more relevant than other ones. IxTeT uses an abstraction hierarchy on the different attribute names (and, as a consequence, on the different flaws) to structure the search space. This hierarchy verifies the **ordered monotonicity property** [17]: *at a given abstraction level, the resolution of a flaw only creates new flaws belonging to the current or less abstract levels.*
Our hierarchy presents two interesting features [10]:

- it can be automatically generated from the description of the planning operators by scanning the conditions and the main effects of the tasks as in [18]; and
- it is a dynamical and least-commitment hierarchy in the sense that, while generating the hierarchy, we do not commit to a total order between the different abstraction levels as it is traditionally done: the search uses a partial-ordered hierarchy (e.g. on figure 3) that is dynamically ordered while planning. This gives the possibility to progress more quickly along the most opportunistic paths of the abstraction lattice than along the most difficult ones whose resolution is delayed.

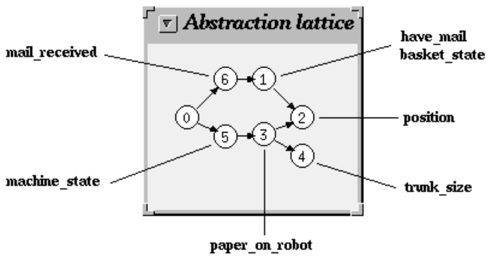

Figure 3: an abstraction lattice

The search tree is controlled by a near-admissible algorithm A_ϵ[12] which provides a trade-off between the efficiency of the search and the quality of the found solution. The estimate f of a partial plan $\mathcal{P}$ at a given node is a combination of g, the commitment along the path leading from $\mathcal{P}_{init}$ to $\mathcal{P}$, and h, a combination of the minimal commitment for each remaining flaw ϕ in $\mathcal{P}$.

4 Examples

4.1 Planning actions for a robot

In this short example, we must plan the activity of a mobile robot in charge of the maintenance of a laboratory (mail delivery, supervision of the good running of machines, ...). It is a plan generation problem: the initial scenario just states that some mail will have to be delivered in rooms LabRoom2 and LabRoom3 after the postman passed and that two printers (Ptr1 and Ptr4) will be out of paper after a given date. Initially the robot has no paper on itself; it can take paper from a warehouse (RoomM) but it can bear no more than 3 reams of paper (the paper carried by the robot is represented as a resource: *paper_on_robot*). The available task operators are: going from one room to another (*go_to*), loading some paper from the warehouse (*load_paper*), getting some mail from a set of baskets in room RoomD4 (*get_mail*), feeding a machine with paper (*feed_machine*) and delivering mail in a given room (*deliver_mail*).

The solution plan given on figure 4 needs about 1s of running time on a Sparc 10 station. The system develops 74 nodes, the average branching factor in the search tree is very low (1.2), this is an effect of our opportunistic search.

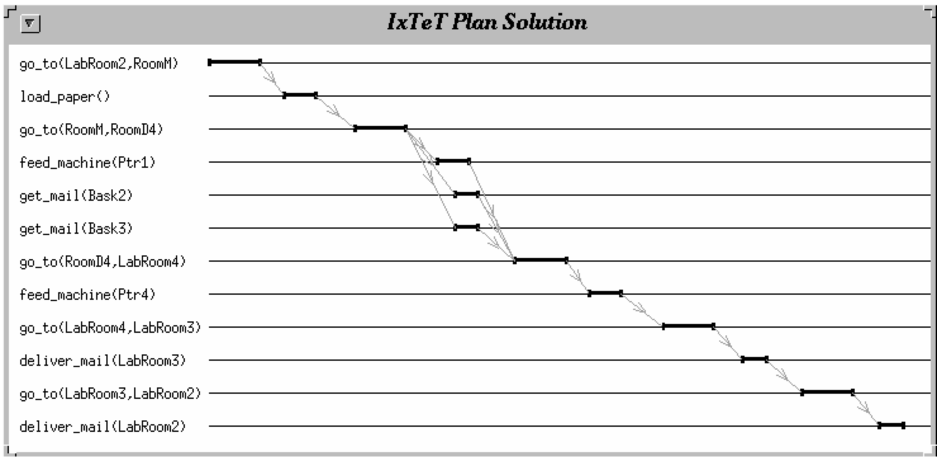

Figure 4: an example of solution plan

4.2 Scheduling with alternative ranges

The following problem consists in producing 5 different elements (A,...,E) with 3 machines (M1,M2,M3). Each element can be produced in two different ways. The problem is to find a schedule that minimise the makespan.

element A:	M2(duration=3), M1(duration=3), M3(duration=6) or
	M2(duration=3), M3(duration=6), M1(duration=3)
element B:	M2(duration=2), M1(duration=5), M2(duration=2), M3(duration=7) or
	M2(duration=2), M3(duration=7), M2(duration=2), M1(duration=5)
element C:	M1(duration=7), M3(duration=5), M2(duration=3) or
	M3(duration=5), M1(duration=7), M2(duration=3)
element D:	M2(duration=4), M3(duration=6), M1(duration=7), M2(duration=4) or
	M2(duration=4), M3(duration=6), M2(duration=4), M1(duration=7)
element E:	M2(duration=6), M3(duration=2) or
	M3(duration=2), M2(duration=6)

Although the size of the problem is very small, it is quite complex. This problem can easily be represented and solved with IxTeT. For each range of each element, a task is defined whose effect is to produce the element. The problem states that all the elements must be produced before a date d that we try to minimise[1]. A solution plan is given on figure 5; it requires about 3s of running time on a Sparc 10.

Conclusion and Future Work

Our integration approach has been tested and shown to be satisfactory in various others application domains such as house building, room finishing problem, task-level control for

[1]As we have not yet implemented some heuristic to minimise this makespan, we solve the problem by interactively decreasing d; the solution on fig. 5 corresponds to $d = 26$; for $d = 25$, IxTeT explores the complete search tree and shows that there is no solution so, $d = 26$ is the best schedule.

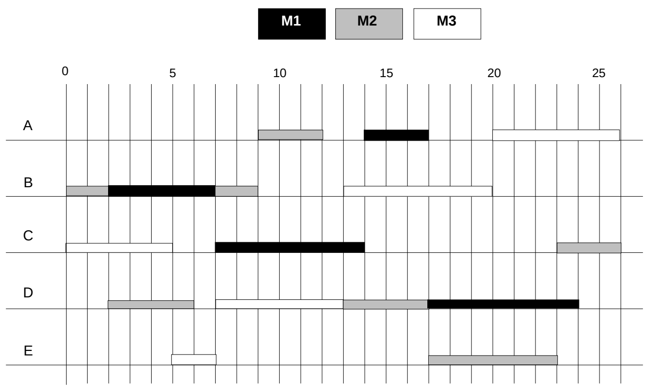

Figure 5: an example of schedule

a team of 3 mobile robots, planning for biological experiments in the Columbus space lab, scheduling of equipment compartments integration for the Ariane 4 rocket launcher, ...
We claim that the integration of plan generation and scheduling is a natural one because these processes are only but the different emanations of the same general activity: planning; and that this integration is useful because some early opportunistic scheduling decisions can help to generate the solution plan. More generally, this work shows how the crossing of various ideas and techniques from the plan generation and the scheduling communities offers many promising perspectives.
Future work in this domain will mainly consist in enriching the system heuristics to take into account more accurately the quality of the solution and increasing the expressiveness of the representation to handle an even larger panel of planning problems.

References

[1] J.F. Allen, H.A. Kautz, R.N. Pelavin, and J.D. Tenenberg. *Reasoning about Plans.* Morgan Kaufmann Pub., 1991.

[2] H. Beck. Tosca: A Novel Approach to the Management of Job-shop Scheduling Constraints. In *Proceedings 9th CIM-Europe Annual Conference, Amsterdam*, pages 138–149, may 1993.

[3] E. Bensana, G. Bel, and D. Dubois. OPAL: a multi-knowledge-based system for industrial job-shop scheduling. *International Journal of Production Research*, 26(5):795–819, 1988.

[4] T. Bylander. Complexity Results for Extended Planning. In *Proceedings of the First International Conference on AI Planning Systems*, 1992.

[5] D. Chapman. Planning for Conjunctive Goals. *Artificial Intelligence*, 32:333–377, 1987.

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

[7] B. Drabble and A. Tate. The use of optimistic and pessimistic resource profiles to inform search in an activity based planner. In *Proceedings AIPS-94*, pages 243–248, 1994.

[8] J. Erschler, P. Lopez, and C. Thuriot. Temporal reasoning under resource constraints: Application to task scheduling. In *Advances in Support Systems Research*, pages 189–194. George E. Lasker and Robbin R. Hough (eds.), 1990.

[9] M.S. Fox and S.F. Smith. ISIS: a knowledge-based system for factory scheduling. *Expert Systems*, 1(1):25–49, 1984.

[10] F. Garcia and P. Laborie. Hierarchisation of the search space in temporal planning. Technical report, LAAS/CNRS, 1994. Submitted to EWSP-95.

[11] M. Ghallab and A. Mounir Alaoui. Managing Efficiently Temporal Relations Through Indexed Spanning Trees. In *Proceedings IJCAI-89*, 1989.

[12] M. Ghallab and D.G. Allard. A_ε: an efficient near admissible heuristic search algorithm. In *Proceedings IJCAI-83*, 1983.

[13] M. Ghallab and H. Laruelle. Representation and Control in Ixtet, a Temporal Planner. In *Proceedings AIPS-94*, pages 61–67, 1994.

[14] M. Ghallab and T. Vidal. Focusing on a Sub-graph for Managing Efficiently Numerical Temporal Constraints. In *Proceedings FLAIRS-95 (to appear)*, 1995.

[15] M. Ghallab and T. Vidal. Temporal constraint in planning: Free or not free ? In *FLAIRS-95 Workshop on Constraint Reasonning (to appear)*, 1995.

[16] T.J. Grant. Lessons for O.R. from A.I.: A Scheduling Case Study. *Journal of the Operational Research Society*, 37(1):41–57, 1986.

[17] C. Knoblock. Learning abstraction hierarchies for problem solving. In *Proceedings AAAI-90*, pages 923–928, 1990.

[18] C. Knoblock. Automatically generating abstractions for planning. *Artificial Intelligence*, 68:243–302, 1994.

[19] C. Knoblock, J. Tenenberg, and Q. Yang. Characterizing Abstraction Hierarchies for Planning. In *Proceedings AAAI*, 1991.

[20] R. Korf. Planning as Search: A Quantitative Approach. *Artificial Intelligence*, 33:65–88, 1987.

[21] P. Laborie. Planifier avec des contraintes de ressources. Technical Report 94077, LAAS-CNRS, Toulouse (France), 1994.

[22] P. Laborie and M. Ghallab. Planning with Sharable Resource Constraints. In *Proceedings IJCAI-95 (to appear)*, 1995.

[23] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proceedings AAAI-91*, pages 634–639, 1991.

[24] N. Sadeh. *Look-Ahead Techniques for Micro-Opportunistic Job Shop Scheduling.* PhD thesis, Carnegie Mellon University, march 1991.

[25] Y. Shoham. *Reasoning About Change.* The MIT Press, Cambridge, MA, 1988.

[26] S.F. Smith, M.S. Fox, and P.S. Ow. Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory. *AI Magazine*, 7(4):45–61, 1986.

[27] E. Tsang. *Foundations of constraint satisfaction.* Academic Press, 1993.

[28] S. Vere. Planning in Time: Windows and Durations for Activities and Goals. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 5(3):246–267, may 1983.

[29] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings AAAI-86*, pages 377–382, 1986.

[30] D. Weld. An introduction to least commitment planning. *AI Magazine*, 1995.

[31] D. E. Wilkins. *Practical Planning.* Morgan Kaufmann, San-Mateo, CA, 1988.