

## On-line Planning and Scheduling: An Application to Controlling Modular Printers

**Minh B. Do**

Embedded Reasoning Area  
Palo Alto Research Center  
Palo Alto, CA 94304 USA  
minhdo at parc . com

**Wheeler Ruml**

Department of Computer Science  
University of New Hampshire  
Durham, NH 03824 USA  
ruml at cs . unh . edu

**Rong Zhou**

Embedded Reasoning Area  
Palo Alto Research Center  
Palo Alto, CA 94304 USA  
rzhou at parc . com

### Abstract

This paper summarizes recent work reported at ICAPS on applying artificial intelligence techniques to the control of production printing equipment. Like many other real-world applications, such as mobile robotics, this complex domain requires real-time autonomous decision-making and robust continual operation. To our knowledge, this work represents the first successful industrial application of embedded domain-independent temporal planning. At the heart of our system is an on-line algorithm that combines techniques from state-space planning and partial-order scheduling. For example, our planning-graph-based planning heuristic takes resource contention into account when estimating makespan remaining. We suggest that this general architecture may prove useful as more intelligent systems operate in continual, on-line settings. Our system has enabled a new product architecture for our industrial partner and has been used to drive several commercial prototypes. When compared with state-of-the-art off-line planners, our system is hundreds of times faster and often finds better plans.

### Introduction

It is a sustaining goal of AI to develop techniques enabling autonomous agents to robustly achieve multiple interacting goals in a dynamic environment. This goal also happens to align perfectly with the needs of many commercial manufacturing plants. In this paper, we focus on one particular manufacturing setting: high-speed Xerox digital production printing systems. Unlike traditional continuous-feed offset presses, digital xerographic printers can treat each sheet differently: printing a different image and performing different preparatory and finishing operations. Often, a single integrated machine can transform blank sheets into a complete document, such as a bound book or a folded bill in a sealed envelope. It is sometimes even possible to process different kinds of jobs simultaneously on the same equipment. Designing a high-performance yet cost-effective controller for such machines is made more difficult by the current trend towards increased modularity, in which each customer's system is unique and includes only those components best-tailored to their needs. Xerox has been exploring architectures in which systems can be composed of literally

hundreds of modules, possibly including multiple specialized printing modules, working together at high speed.

In this paper, we demonstrate how techniques from AI can be used to control such machines. Requests for printed sheets become goals for the system to achieve, the various actuators and mechanisms in the machine become actions and resources to be used in achieving these goals, and sensors provide feedback about action execution and the state of the system. To provide high productivity (and thus high return on investment) for the equipment owner, the planning and control techniques must be fast and produce optimal or near-optimal plans. To reduce the need for operator oversight and to allow the use of very complex mechanisms, the system must be as autonomic as possible. Because operators can make mistakes and even highly-engineered system modules can fail, the system must cope with execution failure and unexpected events. And because the system must work with legacy modules in order to be commercially feasible, its architecture must tolerate components that are out of its direct control.

To meet these requirements, we have developed a novel architecture for on-line planning, execution, and replanning that synthesizes techniques from state-space planning and partial-order scheduling. This integrated approach can potentially benefit any AI system that needs to interleave real-time decision making, planning, and execution, such as robot navigation. Goals are planned for using heuristic state-space planning, in order of arrival, without reconsidering the plans selected for previous sheets. To maintain maximum flexibility, all action times are managed using temporal constraints instead of absolute times. The planner uses no domain-dependent search control knowledge, allowing us to use the same planner to run very different printing systems at full productivity. The domain model can be automatically synthesized from models of the individual components. We also developed new heuristic evaluation functions for temporal planning that incorporate some of the effects of resource constraints. Although domain-independent planning is often regarded as too expensive for use in embedded real-time settings, our system achieves good performance without any hand-coded control rules, despite the additional requirements of reasoning with temporal actions and resources. Indeed, as we summarize later in this paper, our system outperforms state-of-the-art planners on this domain.

Copyright © 2008, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

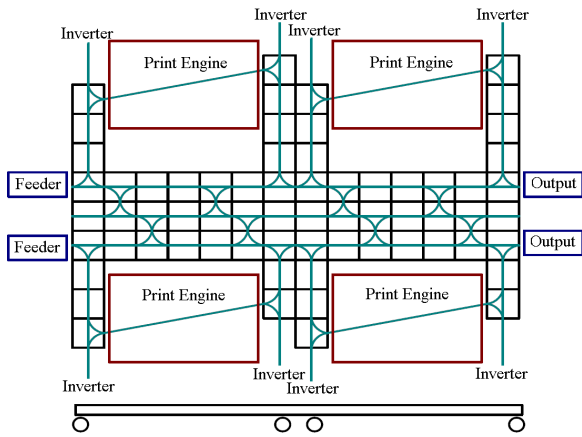


Figure 1: Side view of the four-engine prototype printer built at PARC with over 170 individually controlled modules.

Our work complements the trend in current planning research to extend the expressiveness of domains that AI planners can handle. While PDDL (Fox and Long 2003) has been extended to handle actions over continuous metric quantities and goals with complex preferences, we emphasize the middle ground between planning and scheduling. Choice of actions to perform is important in our domain, but managing resource conflicts is equally important. In particular, our domain emphasizes on-line decision making, which has received only limited attention to date. Our objective is to complete the known print jobs as soon as possible, so taking too long to synthesize a slightly shorter plan is worse than quickly finding a near-optimal solution. In contrast to much work on continual planning (desJardins *et al.* 1999), the tightly-constrained environment of a printing press demands that we produce a complete plan for each goal before its execution can begin. The current paper merely summarizes our results; further details are available from Ruml *et al.* (2005) and Do and Ruml (2006).

### The Modular Printer Domain

A modular printer can be seen as a network of transports linking multiple printing engines. Figure 1 shows a schematic side view of the four-engine prototype printer built at PARC. It has over 170 independently controlled modules and many possible paper paths linking the paper feeders to the possible output trays. Multiple feeders allow blank sheets to enter the printer at a high rate and multiple finishers allow several jobs to run simultaneously. Having redundant paths through the machine enables graceful degradation of performance when modules fail. Each module has a limited number of discrete actions it can perform, and for many of these actions the planner is allowed to control their duration within a range spanning three orders of magnitude.

Controlling these systems involves planning and scheduling a series of requests which arrive asynchronously over time. These printers run at high speed (up to several hundred pages per minute) possibly for many hours. Each request completely describes the attributes of a desired printed

sheet. The plan for printing each individual sheet is a linear sequence of actions. There may be many different sequences of actions that can be used to print a given sheet. For example, in Figure 1, a blank sheet may be fed from any of the two feeders, then routed to any one of the four print engines (or through any two of the four engines in the case of duplex printing) and then to any finisher (unless the sheet is part of an on-going print job). This on-line planning problem is complicated by the fact that many sheets are in-flight simultaneously and the plan for the new sheet must not interfere with those sheets. Moreover, plan synthesis and plan execution are interleaved in real-time. Since it is the wall clock end time that we try to minimize, the speed of the planner itself affects the value of a plan.

Currently, Xerox uses a constraint-based scheduler to control its high-end and mid-range printers (Fromherz *et al.* 1999). The scheduler enumerates all possible plans when the machine starts up and stores them in a database. When printing requests arrive on-line, the scheduler picks the first feasible plan from the database and uses temporal constraint processing to schedule its actions. This decoupling of planning and scheduling is insufficient for complex machines for two reasons. First, the number of possible plans is too large to generate ahead of time, and indeed becomes infinite if loops are present, as in the printer shown in Figure 1. Second, the precompiled plans can be poor choices given the existing sheets in the system. For example, sheets should be fed from different feeders depending on when the previous sheets were fed, how long they are, and how long they will dwell in the print engines (which can be a function of sheet thickness and material). For high performance, we must integrate planning and scheduling.

### Modeling the Domain

The movement of a sheet and the marking actions can be directly translated from the printer model into traditional logical preconditions and effects that test and modify attributes of the sheet. Our action representation is similar to the durative actions in PDDL2.1 with two notable differences. First, we allow actions with real-valued duration bounds. Thus, one can specify upper and lower bounds and let the planner choose the desired duration of the action; this is critical to modeling controllable-speed paper paths, which can be very useful in practice. Second, we use explicit representation of resources. Actions can specify the exclusive use of different types of resources for time intervals specified relative to the action's start or end time. Executing one action may involve allocating multiple resources of various types such as: *unit-capacity*, *multi-capacity*, *cyclic*, and *state* resources. Detailed descriptions of these resource types are given by (Ruml *et al.* 2005).

In addition to the static domain description, the on-line sheet requests are modeled by initial and goal state pairs describing the starting and desired sheet configurations. Each new initial/goal pair defines a new object (the sheet) and associated literals for the planner to track.

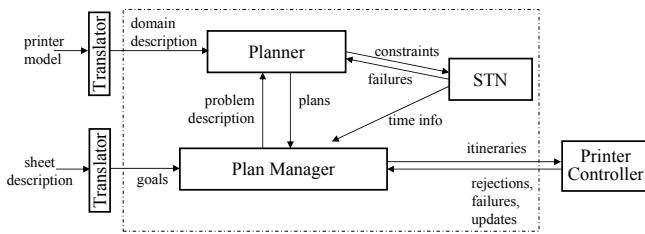


Figure 2: The system architecture, with the planning system indicated by the dashed box.

### On-linePlanner

1. plan the next sheet
2. if an unsent plan starts soon, then
3.   foreach plan, from the oldest through the imminent one
4.     clamp its time points to the earliest possible times
5.     release the plan to the printer controller

### PlanSheet

6. search queue  $\leftarrow$  {final state}
7. loop:
8.   dequeue the most promising node
9.   if it is the initial state, then return
10.   foreach applicable operator
11.     undo its effects
12.     add temporal constraints
13.     foreach potential resource conflict
14.        generate all orderings of the conflicting actions
15.        enqueue any feasible child nodes

Figure 3: Outline of the hybrid planner

## A Hybrid Planning Architecture

Fundamentally, a modular printer resembles any manufacturing plant, with raw materials (blank sheets) entering at the plant's inputs, getting routed through different machines that can change the properties of the materials, and the final products (printed sheets) being collected at the outputs. This domain also has certain properties of package routing or logistics problems. While our on-line planning framework was inspired by these problems, this mix of goal-decomposable planning with cross-goal resource constraints is quite common, and we believe our framework can be applied in a wide range of on-line continual decision-making settings. Multi-robot coordination is one example.

Figure 2 shows the core architecture of the planner and how it communicates with the printer controller. The major components are outlined below. The overall objective is to minimize the end time of all known sheets, ranging over all current print jobs. We approximate this by optimally planning one sheet at a time. Figure 3 gives a sketch of the planning algorithm, which we refer to below.

### Temporal and Plan Management

Printer control is a rich temporal domain with real-time constraints between wall-clock time and the plans for individual sheets, between plans for different sheets, and between the planner and the printer controller. Thus, fast temporal con-

straint propagation, consistency checking, and querying are extremely important in our planner. We maintain the temporal constraints using a Simple Temporal Network (STN) (Dechter *et al.* 1991), represented by the box named *STN* in Figure 2. Essentially, the network contains a set of temporal time points  $t_i$  and constraints between them of the form  $l_b \leq t_i - t_j \leq u_b$ . The time points managed by the STN include action start and end times and resource allocation start and end times. Temporal constraints maintained in the STN are (i) constraints on wall-clock action start time; (ii) action start and end times should be within the action duration range; (iii) constraints between action start time and resource allocation by that action; and (iv) conflicts for various types of resources. For propagation, we use a variation of the arc consistency algorithm described in (Cervoni *et al.* 1994).

Because we use an  $A^*$  search strategy that maintains multiple open search nodes, there is a separate STN for each node. Temporal constraints are added to the appropriate STN when a search node is expanded. Whenever a new constraint is added, propagation tightens the upper and lower bounds on the domain of each affected time point.

Lines 1–5 in Figure 3 correspond to the plan manager. After planning a new sheet, the plan manager checks the queue of planned sheets to see if there are any that could begin soon (line 2). If there are, those plans are released to the printer controller to execute. New temporal constraints are added that freeze the starting time of actions belonging to plans sent to the controller. Those constraints can cause significant propagation and in turn tighten the starting times of actions in the remaining plans.

The large number of potential plans for a given sheet and the close interaction through resource conflicts between plans for different sheets means that it is better to process scheduling constraints during the planning process. The planner uses state-space regression to plan each sheet, but maintains as much temporal flexibility as possible in the plans in the STN using the partial orders between different actions in plans for different sheets. Therefore, it can be seen as a hybrid between state-space search and partial-order planning. Our approach is perhaps similar in spirit to that taken by the IxTeT system (Ghallab and Laruelle 1994).

### Planning Individual Sheets

When planning individual sheets, the regressed state representation contains the (possibly partially-specified) state of the sheet.  $A^*$  search is used to find the optimal plan for the current sheet, in the context of all previous sheets. Because the plan must be feasible in the context of previous plans, the state contains information both about the current sheet and previous plans. More specifically, the state is a 3-tuple  $\langle Literals, STN, R \rangle$ . *Literals* describes the regressed logical state of the current sheet. We represent separately those literals that are currently true and those that are unknown, with false literals being represented implicitly. The *STN* contains all known time points for the state and the current constraints among them. This includes constraints between different plans, between actions in the same plan, as well as against the wall-clock time. Finally, the resource

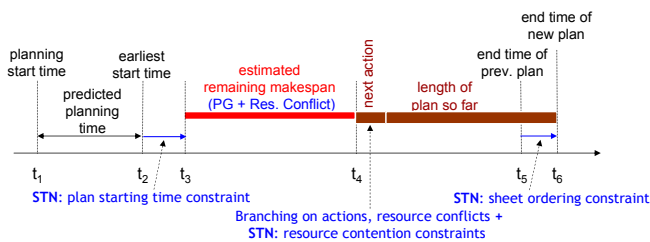


Figure 4: Important time points for evaluating a plan.

profile  $R$  is the set of current resource allocations, representing the commitments made to plans of previous sheets and the partial plan of the current sheet. After the optimal plan for a sheet is found, the resource allocations and  $STN$  used for the plan are passed back to the outer loop in Figure 3 and become the basis for planning the next sheet.

One unusual feature of our planning approach is that we seamlessly integrate planning and scheduling. Starting times of actions are not fixed but merely constrained by temporal ordering constraints in the  $STN$ . Note that in line 14 of Figure 3, we insist that any potential overlaps in allocations for the same resource be resolved immediately, resulting in potentially multiple children for a single action choice. This allows temporal propagation to update the action time bounds and guide plan search. While the plan for a single sheet is a totally-ordered sequence of actions, there are partial orders between actions that belong to plans of different sheets to represent the resource conflict resolutions.

### Objective Function and Heuristic Estimation

Our overall objective is to minimize the earliest possible end time of the plan for the current sheet. To support this, the primary criterion evaluating the promise of a partial plan (line 8 in Figure 3) is the estimate of the earliest possible end time of the partial plan’s best completion. To estimate this quantity, we compute a simple lower bound on the additional makespan required to complete the current regressed plan. This heuristic value is indicated in Figure 4 by *estimated remaining makespan*. It is inserted before the first action in the current plan ( $t_4$ ) and after the plan’s earliest start time ( $t_2$ ). By adding the constraint  $t_2 < t_3$ , the insertion may thus change the end time of the plan. It may also introduce an inconsistency in the temporal database, in which case we can safely abandon the plan. Given that the current plan should end after the end time of all previous sheets in the same print job ( $t_5 < t_6$ ), our objective function is to minimize  $t_6$  without causing any inconsistency in the temporal database. We break ties in favor of smaller predicted makespan ( $t_6 - t_3$ ) and then larger currently realized makespan ( $t_6 - t_4$ ). This is analogous to breaking ties on  $f(n)$  in  $A^*$  search with larger  $g(n)$ , and encourages further extension of plans nearer to a goal. Because our heuristic is admissible, the plan found is optimal according to our objective function.

To estimate the duration required to achieve a given set of goals  $G$  from the initial state, we do dynamic programming over the explicit representation of the bi-level temporal planning graph, in a manner similar to the Temporal Graph-

Plan system (Smith and Weld 1999). To compute more effective heuristic estimates in the presence of significant resource contention, we take into account resource mutexes. For details, see Do and Ruml (2006).

### Experience in Practice

We had several surprises while developing this system. First, we discovered that the forward state-space planning approach that has dominated the planning competitions is not effective for our domain. Our sheet ordering constraint (‘page 1 finishes before page 2’) allows powerful propagation during plan regression. This underscores how one’s objective function, together with domain-specific constraints, can interact with different planning algorithms to determine the most suitable planning approach. We also found that, for on-line continual settings, having a very fast and robust temporal reasoner is key. This is especially true when the planner needs to communicate with the physical modules through other controller components that can incur various network and setup delays. The expressiveness of temporal constraints was important for cleanly representing our objective and tie breaking criteria and allowing us to handle controllable-duration actions. In addition to planning for normal operation, exceptions happen frequently in the physical world and thus we needed to design the planner so that it can handle the most frequent exceptions. Finally, given the emphasis on algorithms in planning research, we were surprised by the importance of modeling. After settling on a language similar to PDDL2.1, we found there were many different ways to represent our domain. We experienced scenarios where using extra predicates sped up the planner by a factor of 10 for some printers. Therefore, achieving peak performance required attention to both good planner implementation and careful modeling of the problem that fit well with the chosen algorithm.

In collaboration with Xerox, we have deployed the planner to control three physical prototype multi-engine printers (one with the schematic view shown in Figure 1). These deployments have been successful and the planner has also been used in simulation to control hundreds of hypothetical printer configurations. The planner is written in Objective Caml, a dialect of ML, and communicates with the job submitter and the printer controller using ASCII text over sockets. The planner can also communicate with a plan visualizer to graphically display the plans. The shortest single plan for the machine shown in Figure 1 has 25 actions. Given that there are many sheets in the printer at any given time and the planner can plan ahead, the plan manager consistently manages dozens of plans and hundreds of actions. During planning, the planner needs to do temporal reasoning regarding the conflict between actions in the current plans and hundreds of actions in previous plans. Even so, the planner consistently on average produces plans within the 0.27 seconds required to keep the printer running at full productivity (220 pages/minute). For one of the most complex current Xerox commercial products, the planner can regularly find the optimal plan within 0.01 seconds and can plan ahead hundreds of sheets. The ability to use domain-independent planning techniques allows us to use the same planner for very differ-

#	LPG		SGPlan		Hybrid	
	Span	Time	Span	Time	Span	Time
1	9.3	0.01	8.3	0.45	8.3	< 0.001
2	13.3	0.02	9.3	308.46	9.4	< 0.001
3	26.6	0.08	-	-	9.9	0.02
4	15.2	0.07	-	-	10.6	0.02
5	21.3	0.12	-	-	11.1	0.03
6	22.4	0.23	-	-	11.8	0.03
7	30.3	8.73	-	-	12.3	0.04
8	19.6	52.55	-	-	13.0	0.06
9	24.2	16.69	-	-	13.5	0.07
10	23.0	20.02	-	-	14.2	0.07
11	29.7	40.14	-	-	14.7	0.08
12	18.3	138.53	-	-	15.4	0.09
13	42.6	29.09	-	-	15.9	0.18
14	34.9	427.41	-	-	16.6	0.21
15	35.3	18.95	-	-	17.1	0.28

Table 1: Comparison of LPG, SGPlan, and our hybrid planner, showing the makespan of the plans found (‘Span’) and planning times (‘Time’) in seconds for problems with various numbers of sheets (‘#’).

ent configurations, without needing any hand-tuned control rules.

Although our planner has features (e.g., variable action durations) beyond even the latest PDDL standard, comparison to PDDL-style planners remains important to validate our planner architecture. While our domain must be simplified to fit the limitations of PDDL, we observe that even these simplified problems are not easy to solve by state-of-the-art academic planners such as SGPlan (Chen *et al.* 2006) and LPG (Gerevini *et al.* 2003), winners of the last several international planning competitions. Since both planners cannot solve any problem for the machine shown in Figure 1, we tested them on a much simpler machine with 14 modules and four print engines. While we only tested a monochrome job with up to 15 simplex sheets, this already stretched the limits of LPG and SGPlan. Our planner can plan ahead hundreds of sheets for this machine. As can be seen in Table 1, SGPlan took more than 5 minutes to find a two-sheet plan that only took our planner less than 0.001 sec to find. Compared to SGPlan, LPG is much faster, although the quality of the plan LPG finds is much worse. On average, LPG returns plans with 86% longer makespan and is about 400 times slower than our planner. For the objective function of minimizing wall-clock finishing time (which combines planning time and plan makespan), our planner is more than 1000x better than both planners for this small printer configuration.

In addition to being faster, our hybrid planner is also more predictable. LPG’s planning time has much higher variance and it sometimes takes longer to plan for a smaller job than a bigger one. For example, it took LPG 22 times longer to plan for the 14-sheet job in Table 1 than it did for the 15-sheet job. This makes it unsuitable for real-time on-line planning, which depends on accurate estimation of planning times for efficient temporal event management.

## Conclusions

We described a real-world domain that requires on-line planning and scheduling and we formalized it using a temporal extension of STRIPS that falls between partial-order scheduling and temporal PDDL. We presented a hybrid planner that uses state-space regression on a per-sheet basis, while using a temporal constraint network to maintain flexibility through partial orderings representing resource conflicts between plans for different sheets. Our system has successfully controlled three hardware prototypes and outperforms state-of-the-art planners in this domain.

Our work provides an example of how AI planning and scheduling can find real-world application not just in exotic domains such as spacecraft or mobile robot control, but also for common down-to-earth problems such as printer control. The modular printer domain is representative of a wider class of AI applications that require continual on-line decision-making. Through a novel combination of fast state-space planning and flexible temporal coordination, we have shown how AI techniques can successfully enable robust, high-performance, autonomous operation without hand-coded control knowledge.

## References

- Roberto Cervoni, Amedeo Cesta, and Angelo Oddi. Managing dynamic temporal constraint networks. In *Proceedings of AIPS-94*, pages 13–18, 1994.
- Yixin Chen, Chih-Wei Hsu, and Benjamim Wah. Temporal planning using subgoal partitioning and resolution in sgplan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.
- Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
- Marie E. desJardins, Edmund H. Durfee, Charles L. Ortiz, Jr., and Michael J. Wolverton. A survey of research in distributed, continual planning. *AI Magazine*, 20(4):13–22, 1999.
- Minh B. Do and Wheeler Ruml. Lessons learned in applying domain-independent planning to high-speed manufacturing. In *Proceedings of ICAPS-06*, pages 370–373, 2006.
- Maria Fox and Derek Long. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 20:61–124, 2003.
- Markus P.J. Fromherz, Vijay A. Saraswat, and Daniel G. Bobrow. Model-based computing: Developing flexible machine control software. *Artificial Intelligence*, 114(1–2):157–202, October 1999.
- Alfonso Gerevini, Alessandro Saetti, and Ivan Serina. Planning through stochastic local search and temporal action graphs in lpg. *Journal of Artificial Intelligence Research*, 20:239–290, 2003.
- Malik Ghallab and Hervé Laruelle. Representation and control in IxTeT, a temporal planner. In *Proceedings of AIPS-94*, pages 61–67, 1994.
- Wheeler Ruml, Minh Binh Do, and Markus Fromherz. On-line planning and scheduling for high-speed manufacturing. In *Proc. of ICAPS-05*, pages 30–39, 2005.
- David E. Smith and Daniel S. Weld. Temporal planning with mutual exclusion reasoning. In *Proc. of IJCAI-99*, pages 326–333, 1999.