Online Planning for a Material Control System for Liquid Crystal Display Manufacturing

Minh Do[1], Kazumichi Okajima[2], Serdar Uckun[1], Fumio Hasegawa[2], Yukihiro Kawano[2], Koji Tanaka[2], Lara Crawford[1], Ying Zhang[1], and Aki Ohashi[1]

[1]Embedded Reasoning Area
Palo Alto Research Center
Palo Alto, CA 94304, USA
{minhdo,uckun,lcrawford,yzang,aki}@parc.com

[2]Products Development Center, IHI Corporation
Yokohama 235-8501, Japan
{kazumichi_okajima,fumio_hasegawa
yukihiro_kawano,koji_tanaka}@ihi.co.jp

Abstract

The hyper-modular printer control project at PARC has proven that a tightly integrated model-based planning and control framework can effectively control a complex physical system. Recently, we have successfully applied this framework to another application: planning for the Material Control System (MCS) of Liquid Crystal Display (LCD) manufacturing plant in a joint project between the Embedded Reasoning Area at PARC and the Products Development Center at the IHI Corporation. The model-based planner created at PARC was able to successfully solve a diverse set of test scenarios provided by IHI, including those that were deemed very difficult by the IHI experts. The short project time (2 months) proved that model-based planning is a flexible framework that can adapt quickly to novel applications. In this paper, we will introduce this complex domain and describe the adaptation process of the Plantrol online planner. The main contributions are: (1) introducing a successful application of general-purpose planning; (2) outline the timeline-based online temporal planner; and (3) description of a complex warehouse management problem that can serve as an attractive benchmark domain for planning.

Introduction

After the success of the Tightly Integrated Parallel Printer (TIPP) project (Ruml *et al.* 2005; Do *et al.* 2008a), the Embedded Reasoning Area (ERA) at PARC has been expanding the software suite developed within this project. Specifically, ERA has been developing the Plantrol framework that integrates the planning and control components for easy adaptation to a wide variety of applications.

In this paper, we will describe our efforts in adapting this framework to the Material Control System (MCS) for an LCD manufacturing plant in a joint project between ERA and the Products Development Center at the IHI Corporation. This project was conducted during a two month period in early 2010, and the planner was successfully validated using a variety of test cases.

The MCS domain presents a challenging online planning problem. The domain involves a complex 2-floor plant and a sequence of real-time orders to transfer LCD cassettes between different locations. The main challenges are:

- *Complex structure:* there are multiple transportation devices with vastly different characteristics and capabilities. Various named locations may have different capabilities and constraints associated with them (e.g., storage locations vs. named intersections).
- *Configuration changes:* transport equipments and storage locations may be added or changed in order to change manufacturing processes and capacity; such additions/changes should not stop the production lines.
- *Unexpected component failures:* different equipments (either transporters or storage locations) may fail or get repaired in real time. The planner needs to route cassettes around failures and avoid getting into deadlock situations.
- *Real-time unpredictable transfer orders:* transfer orders arrive in real-time fashion; the planner does not know in advance about future transfer orders.

Despite the short duration of the project, we have demonstrated that Plantrol model-based planning software is a good fit for this complex logistics domain. The planning software successfully solved all testing scenarios provided by the IHI team and the high-level PTDL modeling language developed for the Plantrol planner is flexible and expressive enough to model all critical constraints in this domain.

The remainder of this paper is organized as follows: in the next section, we will describe the MCS problem in more detail. We then outline the architecture and planning algorithm of the Plantrol online planner. The next two sections describe how we model and solve the MCS problem using model-based planning technology and the empirical evaluation. We conclude the paper with related work and a short discussion and future work.

Material Control System for the LCD Manufacturing Factory

IHI Corporation is a diversified Japanese company with over 23,000 employees that manufactures a wide range of products including jet engines, rockets, ships, bridges, storage and process plants, and industrial machinery. The Products Development Center (PDC) builds control software for various machinery and plants for IHI customers. The MCS system for the LCD manufacturing plant was developed by IHI for an industrial customer (Fumio *et al.* 2009) and the PDC was responsible for developing the *transfer route algorithm*. Even though the MCS project is successful, IHI desires to investigate model-based planning as a more powerful alternative to the current cassette routing algorithm in MCS. For

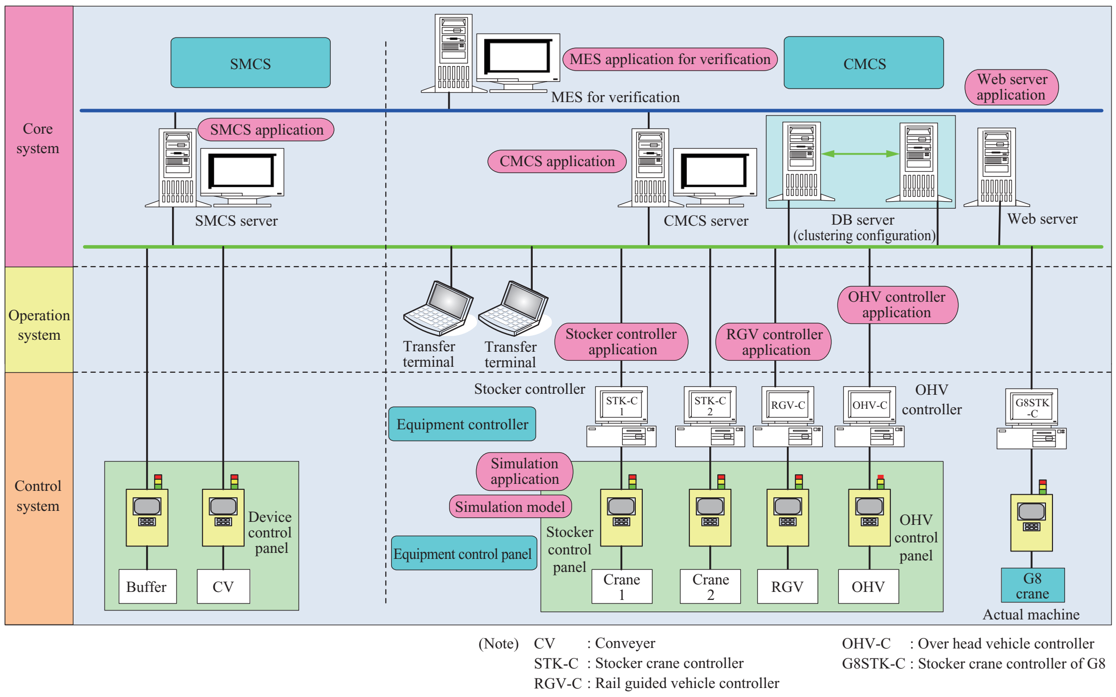


Figure 1: The complete MCS software suit. The CMCS with the transfer route calculation software component is at the top right corner

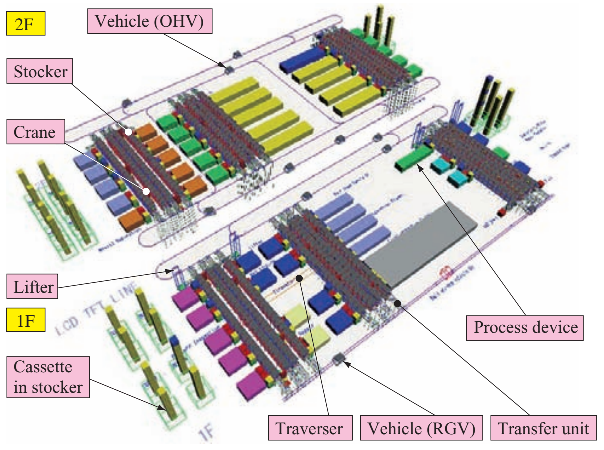
the rest of this section, we will describe the overall manufacturing process, transfer equipments, the MCS control architecture, and the current transfer route calculation algorithm.

LCD Manufacturing Process: IHI has been providing physical distribution equipments and control systems for new LCD factories. LCD manufacturing lines normally consist of four main processes: (1) an array process to make a thin film transistor (TFT) array circuit on a glass substrate; (2) a color filter process to generate the three primary colors, RGB, on another glass substrate; (3) a cell process to stick the completed array substrate and the color filter substrate together and inject liquid crystal; and (4) a module process to integrate a driver (driving circuit), backlight, and other components into the completed cell to make a display (Nishimura 2005). The TFT array process flow in turn consists of four or five cycles of (1) a washing process; (2) a photolithography process, (3) an etching process, (4) an inspection process, (5) a repair process, and other processes. The production flow is complicated and the production volume is large. The production requires a material control system (MCS) which controls transfers between process devices and glass substrate stockers in a TFT array process.

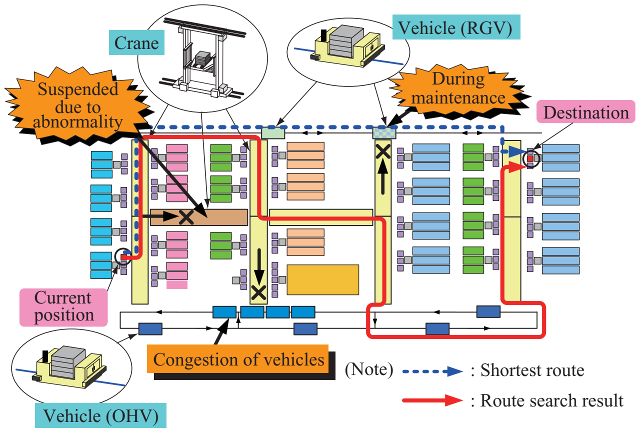
The MCS System: there are two computer systems that control the LCD manufacturing plant: MES and MCS. The manufacturing execution system (MES) performs (1) production planning, (2) liquid crystal manufacturing process control, (3) process device control, and (4) lot control. The MCS receives instructions for transferring glass substrates from the MES and issues control sequences to transfer glass substrates from the current process to the next process by using cranes, vehicles, conveyors, and other transfer devices in the factory. The MCS developed by IHI consists of (1) the MCS that controls cassette transfer (CMCS), (2) the MCS that controls single substrate transfer (SMCS), (3) equipment controllers that control individual transfer facilities, and (4) equipment control panels that control transfer devices. Figure 1 shows the overall architecture of the MCS.

The CMCS in turn has three main functions: (1) the transfer control function, (2) the inventory management function, and (3) the equipment control function (refer to the right side of Figure 1). The transfer control function mainly consists of three types of processing: transfer route search, transfer command creation, and transfer command execution. Upon receiving the transfer instructions from the MES in the $\langle CassetteID, StartingPoint, Destination\rangle$ format, the transfer route search finds the best route connecting *StartingPoint* and *Destination* among multiple valid transfer routes. After the transfer route is determined, the transfer instruction creation process creates transfer instructions for cranes, vehicles, and other transfer facilities on the transfer route. The transfer instructions are then transmitted to individual transfer facilities by the transfer instruction execution process. It is possible to cancel the transfer order or change the destination on the way. The transfer states are monitored and displayed in real time.

Transfer route search: Within the overall MCS system, we only investigated using planning technology as an alternative to the transfer route search component of the CMCS system, as described above. Therefore, throughout the remaining of this paper, when we refer to the MCS system, we mean this CMCS functionality. Given that liquid crystal lines are operated continuously 24 hours a day, 365 days a year (only stopping production for several days of maintenance), it is very important to reduce transfer time between processes and thus improve the overall production efficiency of the factory. Moreover, it is desirable to have a flexible software framework given that transport equipments and storage locations may be added/changed on a liquid crystal



(a) High-level view



(b) Sample route

Figure 2: (a) a two-floor LCD factory plant; and (b) sample cassette route on one floor

manufacturing line in order to change manufacturing processes and improve the manufacturing capacity. Such system additions/changes should not stop the production lines.

In summary, at the highest level, the MCS system receives instructions for transferring LCD cassettes from one manufacturing process to the next by using various types of transportation devices such as: cranes, overhead-vehicles (OHV), rail-guided vehicles (RGV), lifters, and different types of conveyors. Due to the complicated process flow and existence of many concurrent processes, it is virtually impossible to register all transfer routes in advance. Moreover, a process may have devices that are suspended due to failures or for maintenance; making some routes unavailable or causing congestion in real-time. These factors lead to the necessary of having an online algorithm that takes the real-time states of process and transfer devices into consideration and find quickly the route that will reach the destination at the earliest time.

Example: the left side of Figure 2 shows the overall layout of an exemplary plant. Two floors are connected by 4 uni-directional *lifters* at different locations of the plant. Each floor is roughly organized into $3 \times 2 = 6$ regions in which there are 3 pairs of regions that are located close to each other: two pairs to the left while the remaining pair is further to the right. Figure 3 shows a schematic view of the first floor of the plant in Figure 2a in which the 6 regions are more identifiable (each is labeled *BUF11** in red).

In each region, cassettes can be transferred between different locations using one of the two cranes that serves either the upper or lower half of each region. Thus, there are $6 \times 2 = 12$ cranes in each floor and a total of $12 \times 2 = 24$ cranes in the whole factory plant. Possible locations accessible by cranes are: buffer (label:*POT*****), intermediate buffer (label:*IMB*00** or *IMRGV*), input/output ports (label:*TEQ*****) and shifter/traverser/lifter locations (label:**SFT** or **TRV**).

There are also multiple ways to transfer cassettes between different regions: overhead vehicle (OHV), rail-guided vehicle (RGV), shifters (SFT) (label:*B**_SFT*****), traversers (TRV) (label:*TRV1001*) - both shifters and traversers are conveyors, and uni-direction lifters/elevator (LFT) located at locations *TEQ1021, TEQ1026, TEQ1031*, and *TEQ1036*.

All storage locations and transport equipments have different capabilities, capacities, and other equipment-specific constraints. Failures of any of the storage or transfer equipment can happen at any time. The control software needs to route transfer orders around failures or any congested area with an objective function of minimizing the overall makespan. Often, rerouting needs to be done in real time in response to failures. Figure 2b shows one example with a complex route around failures, maintenance, and congestion utilizing multiple transport equipments: $crane \rightarrow crane \rightarrow OHV \rightarrow crane \rightarrow traverser \rightarrow crane \rightarrow RGV \rightarrow RGV \rightarrow crane \rightarrow crane$. Figure 3 shows another exemplary scenario where four ports fail and the controller needs to finish two transfer orders in parallel: $POT1022 \rightarrow POT1032$ and $POT1043 \rightarrow POT1024$. Most complex routes involve routing cassettes between two locations in different floors and the lifters may become bottleneck resources in that case. Fortunately, routing between different floors is less frequent than within the same floor.

Current Approach: IHI's current routing approach is based on a shortest-path algorithm between two locations using a weighted directed connection graph with vertices representing locations where the cassettes can be stored/dropped/picked up, and directed edges between two vertices $v_1 \rightarrow v_2$ representing the availability of a transfer equipment that can move a cassette from v_1 to v_2. When a storage location or a transfer equipment fails or is put back in service, the corresponding vertex or edge is removed from or reinserted into the graph. It's easy to see that if the plant is empty, the shortest path represents the optimal route between any two locations. Difficulties arise when multiple transfer orders are done in parallel. The key innovation in the existing IHI solution is a method to adjust the weights in real time. For example, if the algorithm decides to use edge e for a route of a given cassette, then it will increase the weight for e by a factor w_e. One potential problem with the current approach is that customizing the weight adjustment factor can be difficult and time consuming, especially when mul-

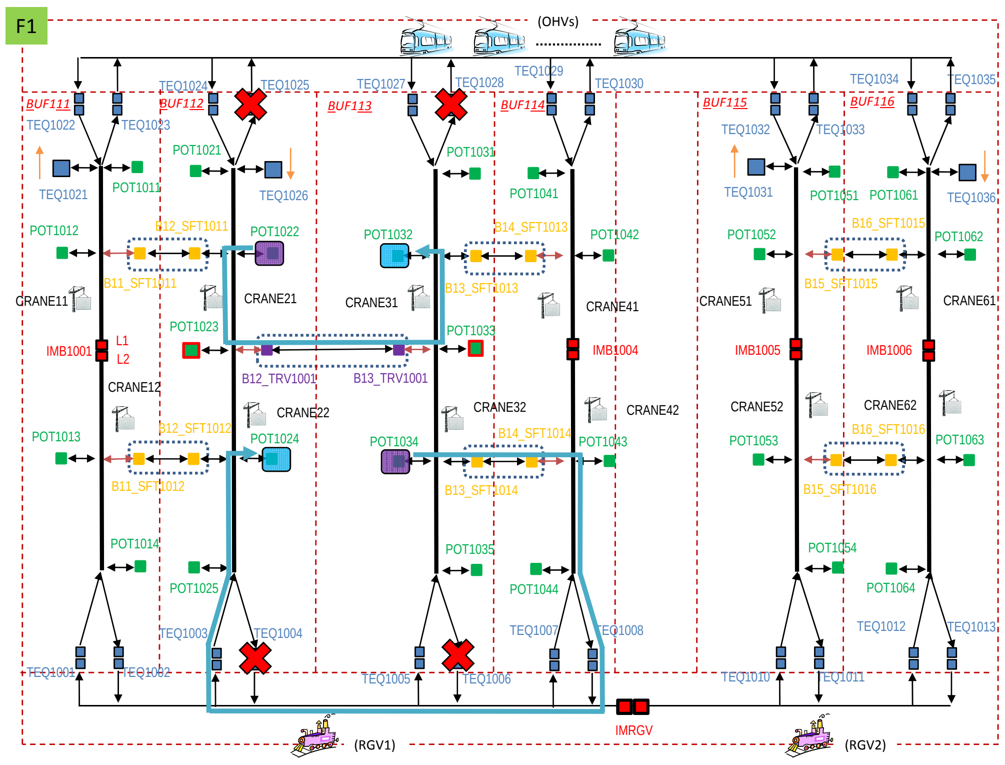


Figure 3: Schematic view of the LCD factory's first floor with: (1) possible failed components; (2) sample concurrent routes around failures.

tiple shortest paths overlap and equipment can fail at random. Moreover, weight adjustment may not avoid routes that cause deadlock. IHI has developed a special routine to detect and resolve deadlocks (Fumio *et al.* 2009). Figure 2b shows a sample route that avoids failures and congestions using weight adjustments.

Plantrol Planner

The Plantrol project is a successor to the TIPP project (Ruml *et al.* 2005; Do *et al.* 2008a). The goal of the Plantrol project is to develop a flexible automation framework that integrates online model-based planning and low-level control. In this section, we will outline the architecture of the online planner in this framework: the modeling language, state representation, online planning/replanning management, and planning algorithm. We will omit many technical details and instead focus on the application.

PTDL Modeling Language: PlanTrol Description Language (PTDL) is a variation of the standard PDDL 3.1 modeling language, thus naturally supporting multi-valued representation. While PTDL follows the familiar PDDL syntax, it concentrates on extending the capability of the PDDL to model more expressive temporal and resource constraints. The main extensions are:

- *Resources:* PTLD models resources explicitly, which makes it more convenient to model resource-rich domains such as the manufacturing sector.
- *Timed condition/effect:* action (pre)conditions and effects can happen within any duration relative to the action's start and end time point.

(:action ohv_move
:parameters (**?**o **- ohv ?**c **- cassette ?**l_1 **?**l_2 **- port)**
:duration [(+ (travel-time ?l_1 **?**l_2**) 84),** max_travel_time**]**
:condition
    (over-all (can-pickup ?o **?**l_1**) (can-dropoff ?**o **?**l_2**))**
:effect
    (over-all (use (space ?o**)))**
    ([start + 44, end] (change (location-of ?c**) ?**l_1 **?**l_2**))**
    ([start + 44, start + 64] (increase (space ?l_1**) 1))**
    ([end - 20, end] (decrease (space ?l_2**) 1)))**

Figure 4: Example of a *move* action represented in PTDL

- *Controllable action duration:* PTDL models the upper and lower bounds on an action's duration. Depending on the particular domain, the planner can either be conservative (e.g., reasoning with the upper-bound value) or opportunistic (e.g., reasoning with the lower-bound value) or through extensive temporal reasoning select the best value within the specified bounds.

Figure 4 shows a simplified version of the action of moving a cassette between two locations l_1 and l_2. Note that there are multiple OHVs but their initial locations are unknown. It takes an average of 44 seconds to retrieve an OHV to the initial location l_1. The OHV then takes 20 seconds to load the cassette and it also takes 20 seconds to unload the cassette into the destination location l_2 after the transfer is done. Thus, we need to model several different temporal effects of this action a happening at different periods within this duration: (1) a uses the OHV for the whole duration; (2) from 44 seconds after a starts until the end of a, the cassette will be moved from l_1 to l_2; (2) 20 seconds after the cassette starts being loaded into the OHV, the location occupied by the cassette at l_1 becomes free; (3) 20 seconds before the end of a, the cassette starts occupying a new location at l_2.

Overall Online Planning & Replanning Algorithm: The overall online continual planning algorithm is not much different from the TIPP planner (Ruml *et al.* 2005). Messages representing transfer orders or plant status updates are sent to the planner in real time. Depending on the message type, different actions are taken:

- *Transfer order*: The parser extracts the source and destination locations, and creates a goal for the online goal manager representing this transfer order.
- *Failures/Repairs*: The planner conducts a look-up operation using the pre-built database map to find all actions that have this failed/repaired object as one of the parameters, and either removes them (in case of failure) or re-enables them (in case of repair) in all future planning or heuristic computation procedures. The planner then invokes the exception handling routine (discussed later).

The planner continually inspects the online goal queue, retrieves the earliest goal g, computes a plan P_g for g that is consistent with all previous plans, and then stores it in the plan manager. Whenever a plan P in the plan manager is ready to execute (starting time of P is close to the current wall-clock time t_c), then it is sent to the execution engine.

Nominal Planning: Nominal planning involves retrieving from the online goal queue at current wall-clock time t_c a goal $g = \langle t_g, X, Y \rangle$ that arrives at the planner at wall-clock time $t_g < t_c$ and represents a transfer order $X \rightarrow Y$. The planner generates a transfer plan/route P that results in the cassette reaching Y at an earliest wall-clock time of $t_e > t_c$. Obviously, minimizing t_e involves minimizing the transfer time (makespan of P) and the planning time to find P, assume that P can be executed as soon as it is generated.

The Plantrol planner implements several different planning algorithms, such as forward-state-space and partial-order, all operate on a timeline-based state representation. Each planning algorithm can use one of several different search algorithms. For this project, we only used the forward-state-space temporal planner running Best-First-Search (BFS) algorithm. Thus, we will limit the discussion to this approach in this paper.

State representation: Our planner uses a timeline-based planning approach that operates by continually maintaining the *timelines* that capture how different variables change their values over time. The planner builds and maintains consistent plans by adding *tokens* to affected timelines. Each token represents the (pre)condition or effect of one action and thus represents a particular type of operation/change affecting the variable represented by that timeline. When the planner starts and reads in the domain description, it will create one empty timeline for each of the grounded state variables, which can be one of three types: (1) binary; (2) multi-valued; (3) continuous. As goals arrive and plans are generated, the timelines are populated with tokens representing action conditions and effects. Figure 5 shows one example of several timelines in the overall state representation, in which we have three different variables with different value domains, and ordered tokens that either change or maintain some certain values. A token T basically represents a durative (pre)condition or effect of a given action and is defined by: (i) start and end time points $start(T)$ and $end(T)$; (ii) a start value v (or bounds $[lb, ub]$ for continuous variables); (iii) start condition (e.g., $v = v_s$) specifies the condition that needs to be satisfied by the token at $start(T)$ (we currently supports: $=, \neq, >, <, \geq, \leq, NONE$; (iv) change operation $\langle operator, value \rangle$ (e.g., $v = v + 5$ or $v \leftarrow x$) specifies how the variable value changes within the token duration. Change operators include: :=, +=, -=, ×=, /=, *CHANGE, USE, MAINTAIN*. The variable value at the end of the token is calculated based on the start value and the change operation. A given timeline is *consistent* if the start values of all tokens are matched by the end values of the tokens preceding them. At any given wall-clock time t_c, there is a *global timelines* TL_G that represents the consistent timelines for all variables. Tokens in TL_G represents all actions in all found plans for previous transfer orders.

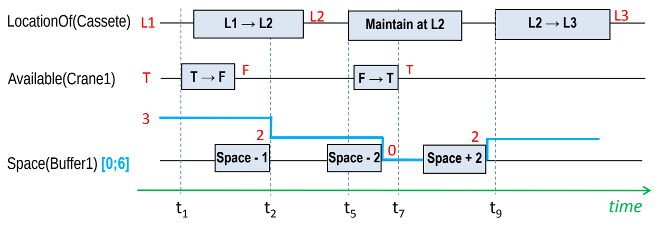


Figure 5: Exemplary timelines representing possible planning state at wall-clock time t_1 with three different variables:(1) multi-valued $LocationOf(Cassette)$; (2) binary $Available(Crane1)$ and (3) continuous $Space(Buffer)$. Tokens represent changes in the future for those three variables due to future plans in the execution queue.

When the planner starts planning for a goal $g = LocationOf(Cassette) : X \rightarrow Y$, at the wall-clock time t_c, it will first make a copy of the global timelines TL_G and create the initial planning state S_I by:

- Remove all tokens that end before t_c from all timelines.
- For new variable $v = LocationOf(Cassette)$, create a token T_1 representing the initial location $v = X$ that ends at t_c.

This initial state, which is a collection of timelines, with its timestamp: $\langle t_c, S_I \rangle$, will be used to setup the root node of the search tree.

Expansion: an action a is *applicable* for a given time-stamped state $\langle t_s, S \rangle$ if there exist a time point $t_a \geq t_s$ such that when a starts at t_a then all tokens representing a's conditions and effects will not cause any inconsistency in any timeline in S. For each such applicable action a at time t_a, the planner creates one child state $\langle t_s, S' \rangle$ by creating one token for each of a's conditions and effects and adds the newly created tokens to the corresponding timelines. For example, if action $a = ohv\_move$ as shown in Figure 4 is added at time $t_a = 100$ and a ends at $t_e = 200$, then 6 tokens are added to S: three spanning the

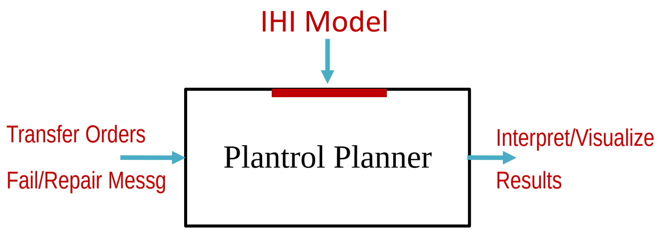


Figure 6: Adapting Plantrol planner for solving MCS problem

duration $d_1 = [100, 200]$ for variables $v_1 = can\_pickup$, $v_2 = can\_dropoff$, and $v_3 = space(ohv)$; one spanning $d_2 = [144, 200]$ for $v_4 = location\_of$; one spanning $d_3 = [44, 64]$ for $v_5 = space(l_1)$; and one spanning $d_4 = [180, 200]$ for $v_6 = space(l_2)$. Like (Do and Kambhampati 2002), there is also a special action *advance-time* that generates a child state of $\langle t_s, S\rangle$ by advancing t_s to the next significant time point $t_{s'} > t_s$, also removes all tokens from S ending before $t_{s'}$.

Goal satisfaction: The plan is complete and satisfies all goals if the end value of the last token in the timeline for each goal variable matches the goal value. For example, the state represented by the timelines in Figure 5 satisfies goals if the goal requirement is $LocationOf(Cassette) = L_3$.

In summary, our planner setting for this project uses a Man-U-Plan-like online continual plan management approach (Ruml *et al.* 2005; Do *et al.* 2008a) and a EUROPA-like timeline-based state representation (J. Frank 2000). Our particular approach used in this project extends the SAPA (Do and Kambhampati 2002) forward temporal planning algorithm to find the final plan. We use heuristics that extend Man-U-Plan resource-adjusted temporal planning graph heuristics (Do and Ruml 2006) to take advantage of the tokens representing previously-generated plans.

Exception Handling: in the MCS domain, exceptions represent unexpected changes in the plant such as equipment failures or repairs performed. As mentioned above, when exceptions happen, the planner will map failures into the set of actions that are either removed or added from the set considered for future plans. However, the Plantrol planner also computes new routes for all existing cassettes given the new plant configuration. It essentially uses a very similar approach to the replanning procedure in the planner used for rerouting sheets in the TIPP printer (Do *et al.* 2008a). For a given cassette c that was originally scheduled to follow a route from X to Y; at the time t_e that the planner receives an exception message, assume that c is transferred between two locations $L_1 \rightarrow L_2$ and is scheduled to reach L_2 at time $t_2 \geq t_e$. The Plantrol planner then creates a new goal $g = \langle t_2, L_2 \rightarrow Y\rangle$. The new goal g is then reinserted in the online goal list at time t_2 and is treated like a regular goal by the planner.

Adapting Plantrol Planner to the MCS Problem

A key promise of domain-independent model-based planning is that the planning software would be applicable across multiple domains, and the adaptation effort would be limited to modeling the problem and not on building a customized solver. This is one of the main reasons PARC and IHI agreed to a very aggressive project schedule.

Figure 6 outlines our adaptation work in which the main Plantrol planner is kept intact and our efforts have been spent on the peripheral issues: (1) building the domain model, (2) setting up the format of the online messages and writing the parser to interpret them, and (3) interpreting and visualizing the results.

Modeling: Much of the project effort was dedicated to the modeling task due to a variety of reasons:

1. The LCD plant is complex. Specifically, we have built models for: (1) different sections of the overall plant; (2) hypothetical plants up to two times bigger than the original problem in order to demonstrate scalability; and (3) some variations of the original plant with different layouts and equipment setups. The final PTDL problem file for the actual LCD plant consists of more than 1500+ lines of PTDL specifications. Furthermore, most of these lines were hand-coded to reflect plant actual geometry (timing and distance numbers)[1].
2. There are different ways to model the LCD plant in PTDL. We had to iterate through modeling choices as we exchange more information about the domain during the course of the project.
3. Due to the physical distance between the teams, most of the relevant plant information was communicated via email. As a result, it took time to elicit and clarify all critical constraints. Indeed, some critical constraints such as *"no-reverse-running for conveyor"* were identified a few days before the final demonstration date.

In the final model of the actual plant that meets IHI requirements, there are action templates that involve three main components: (1) a vehicle/conveyor that carries the cassette; (2) original location; (3) final location. However, given that each location accessed by a given vehicle has different characteristics (e.g., different capacities, special constraints), we use different action templates for different types of locations that are involved with a given action (even if the transfer vehicle is the same). The transfer action for vehicles, in general, consists of several steps: (1) retrieve the vehicle; (2) load a cassette from the source location into the vehicle; (3) move from the source location to the destination location; (4) unload the cassette into the destination location. We have already shown one example for the OHV in Figure 4. While the transfer actions by conveyors are simpler, there are additional constraints such as: no-reverse-running and at most one end location can be occupied at any given time.

Input Parsing: we added an option to parse/interpret MCS models faster: Given that the model file for the MCS problem can be quite large, parsing and grounding the model to generate suitable action sets can be time consuming. Therefore, we have added a command line option to the planner to read in MCS models faster. Note that this option does

[1]In our next phase of engagements, we will build tools to automatically generate the PTDL description file.

not change the final output of the parser at all and thus running with or without this command line option should lead to exactly the same behavior of the planner (i.e., same search space, result, and running time – just faster parsing).

We also designed a custom online MES incoming message syntax for the MCS application and built a small parsing module to interpret it. Input messages include: (1) transfer orders, (2) failures, and (3) repairs. While the underlying planner handles them exactly like for any other Plantrol application, given that the input formats are different, some adjustments to the parsing code were necessary.

Output Presentation: by default, the plan representation is in a standard format of a sequence of actions with their starting times. However, the action description can be quite unreadable with all action parameters included to identify the action instance. Therefore, we have also written code to add an option to translate the plan into a sequence of locations with time stamps to make it more intuitive. We also built a rudimentary visualizer using the Java 3D tool chain to illustrate how cassettes are transferred by different transportation devices according to the plan.

Project Evaluation

In this section, we will describe the evaluation and verification process, deliverables at the end of the project, and the general observations regarding the performance of our model-based planning approach on the MCS problem[2].

Verification using Complex Test Scenarios: As mentioned earlier, the IHI team understands this problem domain very well. Specifically, the Products Development Center has worked on the transfer route algorithm for about 6 months and IHI has developed the whole MCS system for a few years; as the result, IHI has the full software suite deployed in the field. Shortly before the demo, which is one month from the start of the project, the IHI team provided over 10 test cases to verify different planning and exception handling capabilities such as:

- *Load balancing:* test if the planner can balance transfer orders between alternative transportation options.
- *Fault avoidance:* test if the planner can route cassettes around failed components effectively.
- *Deadlock avoidance:* test if the planner can avoid failure scenarios that can cause deadlock between multiple cassettes. Figure 3 shows the simplest test case for this scenario in which deadlock can happen if both cassettes are routed simultaneously using their respective shortest paths that share the traverser *TRV1001*.
- *Reverse running avoidance:* a special (hard) constraint on the conveyor that prevent it to change direction under certain conditions.

The model-based Plantrol planner was able to generate good-quality solutions for all test scenarios. During the workshop, the team of IHI and PARC researchers also created additional complex scenarios to test the planner. We also demonstrated the capability to change plant configuration in real time and the ability of the planner to accommodate these changes. Besides the test cases based on difficult situations encountered by IHI during operation, we also built a random test generator. When given 70 cassette-delivery goals generated with the random start, end locations, and random arrival time averaging 10 cassettes/minute, the average planning time for each goal is 2.43 seconds with good plan quality. Note that the cassette arrival rate of 10 per minute is intentionally set to be higher than realistic scenario and would require the planner to manage approximately 50 packages in transition concurrently.

Comparing the results of the Plantrol planner and IHI routing software, we found that there are some test cases in which two approaches resulted in different plans. Further examination shows that the differences were caused by the differences in the level of detail in the two modeling approaches. The richer temporal and constraint representation of the Plantrol planner resulted in more efficient cassette routes in several cases. Moreover, our team observed that the adaptation time of Plantrol software to the plant modifications is small due to the model-based capability and the expressiveness of the PTDL modeling language.

Deliverables: at the end of the project, the PARC team has delivered to the IHI team: (1) the planner executable; (2) a detailed project report summarizing the modeling, outcome, performance, and complexity analysis; (3) all the PTDL modeling files for the actual LCD plant and other hypothetical variations of the plant; and (4) a simple visualization tool built using Java 3D to graphically display the output plans. The overall conclusions are:

- The PTDL modeling language is expressive enough to model all critical constraints in the MCS joint project.
- The Plantrol planner was able to solve all test scenarios reflecting difficult real-world situations learned from IHI's extensive experience with this problem. It also was able to solve hypothetical situations involving modified plant capabilities and configurations.
- Model-based planning can adapt to new applications rapidly: in a very short time, PARC and IHI joint team was able to develop PTDL models for not only the LCD plant but also for another different application. We demonstrated that the exact same planner executable could solve both problems.

Overall, we believe that the MCS domain is a great match for model-based online planning.

Related Work

From the planning application point of view, there are many related projects. Obviously, the closest application is TIPP (Ruml *et al.* 2005; Do *et al.* 2008a) in which sheets of paper are routed in a way that is similar to how LCD cassettes are routed in the MCS domain. However, the LCD factory plant has a much more diverse set of transporters, and locations have different characteristics and constraints. There are also benchmarks used in the previous IPCs with flavors of this domain such as Logistics, Depots, and DriverLog. The key differences are: (1) the MCS domain involves

[2]Our planner is written in C++. Most tests were executed on a Windows Vista QuadCore Xeon 2.0 GHz machine with 3GB RAM.

many more types of transporters and locations; and more importantly (2) it is not an abstract or simplified setup but a real problem setup with exact timing and distance information.

We have already mentioned earlier that the most related planners are: Man-U-Plan (Ruml *et al.* 2005; Do *et al.* 2008b), EUROPA (J. Frank 2000), and SAPA (Do and Kambhampati 2002). As expected, the closest planner to ours is the Man-U-Plan planner. The Plantrol planner indeed can be conceived as the next version of Man-U-Plan with the following key extensions: (1) a more flexible and expressive timeline-based state representation; (2) support for more types of variables; (3) support for different planning algorithms on timeline; and (4) a more expressive modeling language.

The EUROPA planner developed at the NASA Ames Research Center was another inspiration for our planner design. Like our planner, EUROPA maintains the set of timelines for all variables and adds tokens to different timelines with temporal constraints between related tokens. The key difference with our planner is that EUROPA has no notion of action (only tokens). Therefore, it is hard to adapt traditional action-based planning techniques or heuristics to guide EUROPA search. This was indeed one problem identified in EUROPA (Bernardini and Smith 2008). Moreover, given that EUROPA depends on domain-specific control knowledge to guide the planner, it is time-consuming to adapt it to a new application.

As mentioned earlier, while Plantrol planner incorporates several different planning algorithm (e.g., partial-order and forward state-space planning), forward-state-space was used for this project. In this regard, the algorithm is very similar to Sapa (Do and Kambhampati 2002) extended to handle complex timeline-based state representation and wall-clock time constraints. There are many other academic planners using the forward-state-space framework ranging from classical FF (Hoffmann and Nebel 2001), FastDownward (Helmert 2006) to temporal such as CRIKEY (Andrew Coles and Smith 2009). However, they do not use timelines to track how state variables change over time. This is partly due to the fact that they are offline planners where they do not need to consider wall-clock time evolution or interactions between planning, execution, future commitments, and goal arrivals. We believe that our timeline-based planning framework is more suitable for the online planning scenarios that dominate the class of applications that we are targeting.

Discussion: Given the close relation between the Plantrol modeling language PTDL and the standard planning language PDDL (especially PDDL3.1), it would be useful to see how current state-of-the-art PDDL planners can perform in this domain. After some consideration, we abandoned this idea for several reasons. First, the domain needs to be simplified a lot to accommodate current PDDL-based metric temporal planners such as: convert from multi-valued to binary representation, remove the online continual aspect of the domain, simplify the temporal aspects of our durative action (e.g., Figure 4) to map one PTDL action into multiple PDDL actions with constraints or clip actions to enforce their sequential execution. Moreover, previous experiences modeling a simplified version of the TIPP domain in PDDL to test other planners in (Do *et al.* 2008a) and IPC-08 indicate that current state-of-the-art temporal PDDL planners will not likely do well on this type of domain.

Conclusion & Future Work

In this project, the joint PARC and IHI team has demonstrated that a complex logistics problem such as the MCS problem can be modeled effectively using a general-purpose modeling language (PTDL), and that such complex planning problems can be solved effectively using a model-based planner with no major changes to the planning software or inclusion of domain knowledge. As a further demonstration of the flexibility of the Plantrol approach, we were able to model and solve a preliminary version of a different application using the exact same PTDL language and Plantrol planning software. Potential future enhancements to the Plantrol environment include the full integration of the plan visualizer as well as the development of a graphical modeling environment.

References

Keith Halsey Derek Long Andrew Coles, Maria Fox and Amanda Smith. Managing concurrency in temporal planning using planner-scheduler interaction. *Artificial Intelligence*, 173:1:1–44, 2009.

S. Bernardini and D. Smith. Automatically generated heuristic guidance for europa2. In *Proc. 9th International Symposium on AI, Robotics, and Automation in Space (iSAIRAS-08)*, 2008.

Minh B. Do and Subbarao Kambhampati. Sapa: A scalable multi-objective metric temporal planer. *Journal of Artificial Intelligence Research*, 20:155–194, 2002.

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.

Minh Do, Wheeler Ruml, and Rong Zhou. On-line planning and scheduling: An application to controlling modular printers. In *Proc. of AAAI08*, 2008.

Minh Do, Wheeler Ruml, and Rong Zhou. Planning for modular printers: Beyond productivity. In *Proc. of ICAPS-08*, 2008.

Hasegawa Fumio, Okajima Kazumichi, Shida Michinori, Muramatsu Yuya, and Nakama Mitsuaki. Development of material control system for next generation liquid crystal glass. *IHI Engineering Review*, 42(2):85–96, 2009.

Malte Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, pages 191–246, 2006.

Jörg Hoffmann and Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

P. Morris J. Frank, A Jonsson. On reformulating planning as dynamic constraint satisfaction. In *Symposium on Abstraction, Reformulation and Approximation*, 2000.

Y. Nishimura. Lcd fab, equipment, facilities. *Electronic Journal Corporation*, 9:50–59, 2005.

Wheeler Ruml, Minh B. Do, and Markus Fromherz. On-line planning and scheduling for high-speed manufacturing. In *Proc. of ICAPS-05*, pages 30–39, 2005.