Improving the Temporal Flexibility of Position Constrained Metric Temporal Plans

Minh B. Do & Subbarao Kambhampati
Department of Computer Science and Engineering
Arizona State University, Tempe AZ 85287-5406
{binhminh,rao}@asu.edu

Abstract

In this paper we address the problem of post-processing position constrained plans, output by many of the recent efficient metric temporal planners, to improve their execution flexibility. Specifically, given a position constrained plan, we consider the problem of generating a partially ordered (aka "order constrained") plan that uses the same actions. Although variations of this "partialization" problem have been addressed in classical planning, the metric and temporal considerations bring in significant complications. We develop a general CSP encoding for partializing position-constrained temporal plans, that can be optimized under an objective function dealing with a variety of temporal flexibility criteria, such as makespan. We then propose several approaches (e.g. coupled CSP, MILP) of solving this encoding.

1 Introduction

Of late, there has been significant interest in synthesizing and managing plans for metric temporal domains. Plans for metric temporal domains can be classified broadly into two categories– "position constrained" (p.c.) and "order constrained" (o.c.). The former specify the exact start time for each of the actions in the plan, while the latter only specify the relative orderings between the actions. The two types of plans offer complementary tradeoffs *vis a vis* search and execution. Specifically, constraining the positions gives complete state information about the partial plan, making it easier to control the search. Not surprisingly, several of the more effective methods for plan synthesis in metric temporal domains search for and generate p.c. plans (c.f. TLPlan[1], Sapa[4], TGP [22]). At the same time, from an execution point of view, o.c. plans are more advantageous than p.c. plans –they provide better execution flexibility both in terms of makespan and in terms of "scheduling flexibility" (which measures the possible execution traces supported by the plan [23; 20]). They are also more effective in interfacing the planner to other modules such as schedulers (c.f. [15]), and in supporting replanning and plan reuse [24; 13].

A solution to the dilemma presented by these complementary tradeoffs is to search in the space of p.c. plans, but post-process the resulting p.c. plan into an o.c. plan. Although such post-processing approaches have been considered in classical planning ([14; 24; 2]), the problem is considerably more complex in the case of metric temporal planning. The complications include the need to handle the more expressive action representation and the need to handle a variety of objective functions for partialization (in the case of classical planning, we just consider the least number of orderings)

Our contribution in this paper is to first develop a Constraint Satisfaction Optimization Problem (CSOP) encoding for converting a p.c. plan in metric/temporal domains into an o.c. plan. This general framework allows us to specify a variety of objective functions to choose between the potential partializations of the p.c. plan. Among several approaches to solve this CSOP encoding, we will discuss in detail the one approach that converts it to an equivalent MILP encoding, which can then be solved using any MILP solver such as CPLEX or LPSolve to produce an o.c. plan optimized for some objective function. Our intent in setting up this encoding was not to solve it to optimum–since that is provably NP-hard [2]–but to use it for baseline characterization of greedy partialization algorithmsm, which is presented in the extended version of this paper ([5]). In that paper (accepted to ICAPS03), the greedy value ordering strategy is designed to efficiently generate solutions with good makespan values for the CSOP encodings. We demonstrate the effectiveness of our greedy partialization approach in the context of a recent metric temporal planner named *Sapa* that produces p.c. plans. In the extended version, we also empirically compare the effects of greedy and optimal partialization using MILP encodings on the set of metric temporal problems used at the Third International Planning Competition.

The paper is organized as follows. First, we provide the definitions related to the partialization problem. Then, we discuss the CSOP encoding for the partialization problem and focus on how the CSOP encoding can be solved. Finally, we discuss the related work.

2 Problem Definition

Position and Order constrained plans: *A position constrained plan (p.c.) is a plan where the execution time of each action is fixed to a specific time point. An order constrained (o.c.) plan is a plan where only the relative orderings between the actions are specified.*

There are two types of position constrained plans: *serial* and *parallel*. In a serial position constrained plan, no concurrency is allowed. In a parallel position constrained plan, actions are allowed to execute concurrently. Examples of the serial p.c. plans are the ones returned by classical planners such as AltAlt[19], HSP[3], FF[10]. The parallel p.c. plans are the ones returned by Graphplan-based planners and the temporal planners such as *Sapa* [4], TGP[22], TP4[9]. Examples of planners that output order constrained (o.c.) plans are Zeno[21], HSTS[18], IxTeT[15].

Figure 1 shows, on the left, a valid p.c. parallel plan consisting of four actions A_1, A_2, A_3, A_4 with their starting time

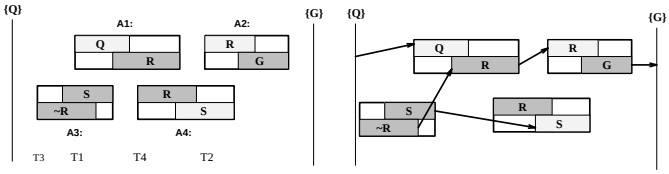

Figure 1: Examples of p.c. and o.c. plans

points fixed to T_1, T_2, T_3, T_4, and on the right, an o.c plan consisting of the same set of actions and achieving the same goals. For each action, the marked rectangular regions show the durations in which each precondition or effect should hold during each action's execution time. The shaded rectangles represent the effects and the white ones represent preconditions. For example, action A_1 has a precondition Q and effect R and action A_3 has no preconditions and two effects $\neg R$ and S.

It should be easy to see that o.c. plans provide more execution flexibility than p.c. plans. In particular, an o.c. plan can be "dispatched" for execution in any way consistent with the relative orderings among the actions. In other words, for each valid o.c. plan P_{oc}, there may be multiple valid p.c. plans that satisfy the orderings in P_{oc}, which can be seen as different ways of dispatching the o.c. plan.

While generating a p.c. plan consistent with an o.c. plan is easy enough, in this paper, we are interested in the reverse problem–that of generating an o.c. plan given a p.c. plan.

Partialization: *Partialization is the process of generating a valid order constrained plan P_{oc} from a set of actions in a given position constrained plan P_{pc}.*

We can use different criteria to measure the quality of the o.c. plan resulting from the partialization process (e.g. makespan, slack, number of orderings). One important criterion is a plan's "makespan." The *makespan* of a plan is the minimum time needed to execute that plan. For a p.c. plan, the makespan is the duration between the earliest starting time and the latest ending time among all actions. In the case of serial p.c. plans, it is easy to see that the makespan will be greater than or equal to the sum of the durations of all the actions in the plan.

For an o.c. plan, the makespan is the minimum makespan of any of the p.c. plans that are consistent with it. Given an o.c. plan P_{oc}, there is a polynomial time algorithm based on topological sort of the orderings in P_{oc}, which outputs a p.c. plan P_{pc} where all the actions are assigned earliest possible start time point according to the orderings in P_{oc}. The makespan of that p.c. plan P_{pc} is then used as the makespan of the original o.c. plan P_{oc}.

3 Formulating a CSOP encoding for the partialization problem

In this section, we develop a general CSOP encoding for the partialization problem. The encoding contains both continuous and discrete variables. The constraints in the encoding guarantee that the final o.c plan is consistent, executable, and achieves all the goals. Moreover, by imposing different user's objective functions, we can get the optimal o.c plan by solving the encoding.

3.1 Preliminaries

Let P_{pc}, containing a set of actions $\mathcal{A}$ and their fixed starting times $st^{pc}_{\mathcal{A}}$, be a valid p.c. plan for some temporal planning problem $\mathcal{P}$. We assume that each action A in P_{pc} is in the standard PDDL2.1 Level 3 representation [8].[1] To facilitate the discussion on the CSOP encoding in the following sections, we will briefly discuss the action representation and the notation used in this paper:

- For each (pre)condition p of action A, we use $[st^p_A, et^p_A]$ to represent the duration in which p should hold ($st^p_A = et^p_A$ if p is an instantaneous precondition).
- For each effect e of action A, we use et^e_A to represent the time point at which e occurs.
- For each resource r that is checked for preconditions or used by some action A, we use $[st^r_A, et^r_A]$ to represent the duration over which r is accessed by A.
- The initial and goal states are represented by two new actions A_I and A_G. A_I starts before all other actions in the P_{pc}, it has no preconditions and has effects representing the initial state. A_G starts after all other actions in P_{pc}, has no effects, and has top-level goals as its preconditions.
- The symbol "$\prec$" is used through out this section to denote the relative precedence orderings between two time points.

Note that the values of $st^p_A, et^p_A, et^e_A, st^r_A, et^r_A$ are fixed in the p.c plan but are only partially ordered in the o.c plan.

3.2 The CSOP encoding for the partialization problem

Let P_{oc} be a partialization of P_{pc} for the problem $\mathcal{P}$. P_{oc} must then satisfy the following conditions:

1. P_{oc} contains the same actions $\mathcal{A}$ as P_{pc}.
2. P_{oc} is executable. This requires that the (pre)conditions of all actions are satisfied, and no pair of interfering actions are allowed to execute concurrently.
3. P_{oc} is a valid plan for $\mathcal{P}$. This requires that P_{oc} satisfies all the top level goals (including deadline goals) of $\mathcal{P}$.
4. (Optional) The orderings on P_{oc} are such that P_{pc} is a legal dispatch (execution) of P_{oc}.
5. (Optional) The set of orderings in P_{oc} is minimal (i.e., all ordering constraints are non-redundant, in that they cannot be removed without making the plan incorrect).

Given that P_{oc} is an order constrained plan, ensuring goal and precondition satisfaction involves ensuring that (a) there is a causal support for the condition and that (b) the condition, once supported, is not violated by any possibly intervening action. The fourth constraint ensures that P_{oc} is in some sense an *order generalization* of P_{pc} [14]. In the terminology of [2], the presence of fourth constraint ensures that P_{oc} is a de-ordering of P_{pc}, while in its absence P_{oc} can either be a de-ordering or a re-ordering. This is not strictly needed if our interest is only to improve temporal flexibility. Finally, the fifth constraint above is optional in the sense that any objective function defined in terms of the orderings anyway ensures that P_{oc} contains no redundant orderings.

In the following, we will develop a CSP encoding for finding P_{oc} that captures the constraints above. This involves specifying the variables, their domains, and the inter-variable constraints.

Variables: The encoding will consist of both continuous and

[1] PDDL2.1 Level 3 is the highest level used in the Third International Planning Competition.

discrete variables. The continuous variables represent the temporal and resource aspects of the actions in the plan, and the discrete variables represent the logical causal structure and orderings between the actions. Specifically, for the set of actions in the p.c. plan P_{pc} and two additional dummy actions A_i and A_g representing the initial and goal states,[2] the set of variables are as follows:

Temporal variables: For each action A, the encoding has one variable st_A to represent the time point at which we can start executing A. The domain for this variable is $Dom(st_A) = [0, +\infty)$.

Resource variables: For each action A and the resource $r \in R(A)$, we use a pseudo variable[3] V_A^r to represent the value of r (resource level) at the time point st_A^r.

Discrete variables: There are several different types of discrete variables representing the causal structure and qualitative orderings between actions:

- *Causal effect:* We need variables to specify the causal link relationships between actions. Specifically, for each condition $p \in P(A)$ and a set of actions $\{B_1, B_2.....B_n\}$ such that $p \in E(B_i)$, we set up one variable: S_A^p where $Dom(S_A^p) = \{B_1, B_2....B_n\}$.
- *Interference:*Two actions A and A' are in logical interference on account of p if $p \in Precond(A) \cup Effect(A)$ and $\neg p \in Effect(A')$. For each such pair, we introduce one variable $I_{AA'}^p$: $Dom(I_{AA'}^p) = \{\prec, \succ\}$ ($A\ before_p\ A'$, or $A\ after_p\ A'$). For the plan in Figure 1, the interference variables are: $I_{A_1A_3}^R$ and $I_{A_2A_3}^R$. Sometimes, we will use the notation $A \prec_p A'$ to represent $I_{AA'}^p = \prec$.
- *Resource ordering:* For each pair of actions A and A' that use the same resource r, we introduce one variable $R_{AA'}^r$ to represent the resource-enforced ordering between them. If A and A' can not use the same resource concurrently, then $Dom(R_{AA'}^r) = \{\prec, \succ\}$, otherwise $Dom(R_{AA'}^r) = \{\prec, \succ, \perp\}$. Sometimes, we will use the notation $A \prec_r A'$ to represent $R_{AA'}^p = \prec$.

Following are the necessary constraints to represent the relations between different variables:

1. Causal link protections: If B supports p to A, then every other action A' that has an effect $\neg p$ must be prevented from coming between B and A:
   $S_A^p = B \Rightarrow \forall A',\ \neg p \in E(A') : (I_{A'B}^p = \prec) \vee (I_{A'A}^p = \succ)$
2. Constraints between ordering variables and action start time variables: We want to enforce that if $A \prec_p A'$ then $et_A^p < st_{A'}^p$. However, because we only maintain one continuous variable st_A in the encoding for each action, the constraints need to be posed as follows:

$$I_{AA'}^p = \prec \Leftrightarrow\ st_A + (et_A^{\neg p} - st_A) < st_{A'} + (st_{A'}^p - st_{A'}).$$
$$I_{AA'}^p = \succ \Leftrightarrow\ st_{A'} + (et_{A'}^p - st_{A'}) < st_A + (st_A^{\neg p} - st_A).$$
$$R_{AA'}^p = \prec \Leftrightarrow\ st_A + (et_A^r - st_A) < st_{A'} + (st_{A'}^r - st_{A'}).$$
$$R_{AA'}^r = \succ \Leftrightarrow\ st_{A'} + (et_{A'}^r - st_{A'}) < st_A + (st_A^r - st_A).$$

Notice that all values $(st_A^{p/r} - st_A)$, $(et_A^{p/r} - st_A)$ are constants for all actions A, propositions p, and resource r.

3. Constraints to guarantee the resource consistency for all actions: Specifically, for a given action A that has a resource constraint $V_{st_A^r}^r > K$, let U_A^r be an amount of resource r that A produces/consumes ($U_A^r > 0$ if A produces r and $U_A^r < 0$ if A consumes r). Suppose that $\{A_1, A_2, ...A_n\}$ is the set of actions that also use r and $Init_r$ be the value of r at the initial state, we set up a constraint that involves all variables $R_{A_iA}^r$ as follows:

$$Init_r + \sum_{A_i \prec_r A} U_{A_i}^r + \sum_{A_i \perp_r A, U_{A_i}^r < 0} U_{A_i}^r > K \quad (3)$$

   (where $A_i \prec_r A$ is a shorthanded notation for $R_{A_iA}^r = \prec$). The constraint above ensures that regardless of how the actions A_i that have no ordering relations with A ($R_{A_iA}^r = \perp$) are aligned temporally with A, the orderings between A and other actions guarantee that A has enough resource ($V_{st_A^r}^r > K$) to execute.
4. Deadlines and other temporal constraints: These model any deadline type constraints in terms of the temporal variables. For example, if all the goals need to be achieved before time t_g, then we need to add a constraint: $st_{A_g} \leq t_g$. Other temporal constraints, such as those that specify that certain actions should be executed before/after certain time points, can also be handled by adding similar temporal constraints to the encoding (e.g $L \leq st_A \leq U$).
5. Constraints to make the orderings on P_{oc} consistent with P_{pc} (optional): Let T_A be the fixed starting time point of action A in the original p.c plan P_{pc}. To guarantee that P_{pc} is consistent with the set of orderings in the resulting o.c plan P_{oc}, we add a constraint to ensure that the value T_A is always present in the live domain of the temporal variable st_A.

3.3 Objective function

Each satisficing assignment for the encoding above will correspond to a possible partialization of P_{pc}, i.e., an o.c. plan that contains all the actions of P_{pc}. However, some of these assignments (o.c. plans) may have better execution properties than the others. We can handle this by specifying an objective function to be optimized, and treating the encoding as a Constraint Satisfaction Optimization (CSOP) encoding. The only requirement on the objective function is that it is specifiable in terms of the variables of the encodings. Objective functions such as makespan minimization and order minimization readily satisfy this requirement. Following are several objective functions that worth investigating:

Temporal Quality:

- Minimum Makespan: $minimize\ Max_A(st_A + dur_A)$
- Maximize summation of slacks:

$$Maximize \sum_{g \in Goals} (st_{A_g}^g - et_A^g)\ :\ S_{A_g}^g = A$$

- Maximize average flexibility:
  $Maximize\ Average(Domain(st_A))$

Ordering Quality:

- Fewest orderings: $minimize\ \#(st_A \prec st_{A'})$

[2] A_i has no preconditions and has effects that add the facts in the initial state. A_g has no effect and has preconditions representing the goals.

[3] We call V a pseudo variable because the constraints involving V are represented not directly, but rather indirectly by the constraints involving U_A^r; see below.

4 Solving the partialization encoding

Given the presence of both discrete and temporal variables in this encoding, the best way to handle it is to view it as a leveled CSP encoding, where in the satisficing assignments to the discrete variables activate a set of temporal constraints between the temporal variables. These temporal constraints, along with the deadline and order consistency constraints are represented as a temporal constraint network [6]. Solving the network involves making the domains and inter-variable intervals consistent across all temporal constraints [23]. The consistent temporal network then represents the o.c. plan. Actions in the plan can be executed in any way consistent with the temporal network (thus providing execution flexibility). All the temporal constraints are "simple" [6] and can thus be handled in terms of a simple temporal network. Optimization can be done using a branch and bound scheme on top of this.

Although the leveled CSP framework is a natural way of solving this encoding, unfortunately, there are no off-the-shelf solvers which can support its solution. Because of this, for the present, we convert the encoding into a Mixed Integer Linear Programming (MILP) problem, so it can be solved using existing MILP solvers, such as LPSolve and CPLEX. In the following, we discuss the details of the conversion into MILP.

4.1 Optimal Post-Processing Using MILP Encoding

Given the CSOP encoding discussed in the previous section, we can convert it into a Mixed Integer Linear Program (MILP) encoding and use any standard solver to find an optimal solution. The final solution can then be interpreted to get back the o.c plan. In this section, we will first discuss the set of MILP variables and constraints needed for the encoding, then, we concentrate on the problem of how to setup the objective functions using this approach.

MILP Variables and Constraints

For the corresponding CSOP problem, the set of variables and constraints for the MILP encoding is as follows:

Variables: We will use the the binary integer variables (0,1) to represent the logical orderings between actions and linear variables to represent the starting times of actions in the CSOP encoding.

- *Binary (0,1) Variables:*
  1. Causal effect variables: $X^p_{AB} = 1$ if $S^p_A = B$, $X^p_{AB} = 0$ otherwise.
  2. Mutual exclusion (mutex) variables: $Y^p_{AB} = 1$ if $I^p_{AB} = \prec$, $Y^p_{BA} = 1$ if $I^p_{AB} = \succ$,
  3. Resource interference variables: $X^r_{AA'} = 1$ if $A \prec_r A'$ (i.e. $et^r_A < st^r_{A'}$). $N^r_{AA'} = 1$ if there is no order between two actions A and A' (they can access resource r at the same time).[4]
- *Continuous Variable:* one variable st_A for each action A and one variable st_{A_g} for each goal g.

Constraints: The CSP constraints discussed in the previous section can be directly converted to the MILP constraints as follows:

- Mutual exclusion: $Y^p_{AB} + Y^p_{BA} = 1$
- Only one supporter: $\forall p \in Precond(A) : \Sigma X^p_{BA} = 1$
- Causal-link protection: $\forall A', \neg p \in Effect(A') : (1 - X^p_{AB}) + (Y^p_{A'A} + Y^p_{BA'}) \geq 1$
- Ordering and temporal variables relation: $M.(1 - X^p_{AB}) + (st^p_B - et^p_A) > 0$; where M is a very big constant.[5]
- Mutex and temporal variables relation: $M.(1 - Y^p_{AB}) + (st^p_B - et^p_A) > 0$
- Resource-related constraints: Let U^r_A be the amount of resource r that the action A uses. $U^r_A < 0$ if A consumes (reduces) r and $U^r_A > 0$ if A produces (increases) r. For now, we assume that U^r_A are constants for all actions A in the original p.c plan.
  - Only one legal ordering between two actions: $X^r_{AA'} + X^r_{A'A} + N^r_{AA'} = 1$
  - Resource ordering and temporal ordering relations: $M.(1 - X^r_{AA'}) + (st^r_{A'} - et^r_A) > 0$
  - Constraints for satisficing resource-related preconditions:

$$Init_r + \sum X^r_{A'A}.U^r_{A_i} + \sum_{U^r_B < 0} N^r_{AB}.U^r_B > K \quad (4)$$

    if the condition to execute action A is that the resource level of r when A starts executing is higher than K.[6]
- Constraints to enforce that all actions start after A_{init} and finish before A_{goal}: $\forall A : st_A - st_{A_{init}} \geq 0, st_{A_{goal}} - (st_A + dur_A) \geq 0.$
- Goal deadline constraints: $st_{A_g} \leq Deadline(g)$

Note that in the equation (4) listed above, we assume that U^r_A are all constants for all resource-related functions r and actions A. The reason is that if U^r_A are also variables (non-constant), then equation (4) is no longer a linear equation (and thus can not be handled by a MILP solver).

MILP Objective Functions

Starting from the base encoding above, we can model a variety of objective functions to get the optimal o.c. plans upon solving MILP encoding as follows:

Minimum Makespan:

- An additional (continuous) variable to represent the plan makespan value: V_{ms}
- Additional constraints for all actions in the plan: $\forall A : \; st_A + dur_A \leq V_{ms}$
- MILP Objective function: $minimize\, V_{ms}$

[4] In PDDL 2.1, two actions A and B are allowed to access the same function (resource) overlappingly if: (1) A do not change any function that B is checking as its precondition; (2) A and B using the functions to change the value of r in a *commute* way (increase/decrease only).

[5] The big constant M enforces the logical constraint: $X^p_{AB} = 1 \Rightarrow et^p_A < st^p_B$. Notice that if $X^p_{AB} = 0$ then no particular relation is needed between et^p_A and st^p_B. In this case, the objective function would take care of the actual value of et^p_A and st^p_B. The big M value can be any value which is bigger than the summation of the durations of all actions in the plan.

[6] This constraint basically means that even if the actions that has no ordering with A ($N^r_{AA'} = 1$) align with A in the worst possible way, the A has enough r at its starting time. Notice also that the initial level of r can be considered as the production of the initial state action A_{init}, which is constrained to execute before all other actions in the plan.

Maximize minimum slack[7] *value:*

- An additional (continuous) variable to represent the minimum slack value: V_{ms}
- Additional constraints for all goals: $\forall g \forall A: \ V_{ms} - (M.X^g_{AA_g} + (st_{A_g} - et^g_A)) \geq 0$, M is a very big constant.
- MILP objective function: $minimize\, V_{ms}$

Minimum number of orderings:

- Additional binary ordering variables for every pair of actions: O_{AB}
- Additional constraints: $\forall A, B, p : O_{AB} - X^p_{BA} \geq 0, O_{AB} - Y^p_{AB} \geq 0$
- MILP objective function: $minimize \Sigma O_{AB}$

5 Related Work

The complementary tradeoffs provided by the p.c. and o.c. plans have been recognized in classical planning. One of the earliest efforts that attempt to improve the temporal flexibility of plans was the work by Fade and Regnier [7] who discussed an approach for removing redundant orderings from the plans generated by STRIPS system. Later work by Mooney [17] and Kambhampati and Kedar [14]characterized this partialization process as one of explanation-based order generalization. Backstrom [2] categorized approaches for partialization into "de-ordering" approaches and "re-ordering" approaches. The order generalization algorithms fall under the de-ordering category. He was also the first to point out the NP-hardness of maximal partialization, and to characterize the previous algorithms as greedy approaches.

The work presented in this paper can be seen as a principled generalization of the partialization approaches to metric temporal planning. Our novel contributions include: (1) providing a CSP encoding for the partialization problem and (2) characterizing the greedy algorithms for partialization as specific value ordering strategies on this encoding. In terms of the former, our partialization encoding is general in that it encompasses both de-ordering and re-ordering partializations–based on whether or not we include the optional constraints to make the orderings on P_{oc} consistent with P_{pc}. In terms of the latter, the work in [24] and [14] can be seen as providing a greedy value ordering strategy over the partialization encoding for classical plans. However, unlike the greedy strategies presented in this paper, their value ordering strategies are not sensitive to any specific optimization metric.

It is interesting to note that our encoding for partialization is closely related to the so-called "causal encodings" [12]. Unlike casual encodings, which need to consider supporting a precondition or goal with every possible action in the action library, the partialization encodings only need to consider the actions that are present in P_{pc}. In this sense, they are similar to the encodings for replanning and plan reuse described in [16]. Also, unlike causal encodings, the encodings for partialization demand optimizing rather than satisficing solutions. Finally, in contrast to our encodings for partialization which specifically handle metric temporal plans, causal encodings in [12] are limited to classical domains.

[7] The objective function of *maximize maximum slack* and *maximize summation of slack* can be handled similarly.

References

[1] Bacchus, F. and Ady, M. 2001. Planning with Resources and Concurrency: A Forward Chaining Approach. *Proc IJCAI-2001*.

[2] Backstrom, C. 1998. Computational Aspects of Reordering Plans *Journal of Artificial Intelligence Research 9, 99-137*.

[3] Bonet, B., Loerincs, G., and Geffner, H. 1997. A robust and fast action selection mechanism for planning. *Proc AAAI-97*

[4] Do, M., and Kambhampati, S. 2001. Sapa: A Domain-Independent Heuristic Metric Temporal Planner. *Proc ECP-01*

[5] Do, M., and Kambhampati, S. 2003. Improving the Temporal Flexibility of Position Constrained Metric Temporal Planning. To appear in *Proc. ICAPS-03*.

[6] Dechter, R., Meiri, I., and Pearl, J. 1990. Temporal Constraint Network. *Artificial Intelligence Journal 49*.

[7] Fade, B. and Regnier, P. 1990 Temporal Optimization of Linear Plans of Action: A Strategy Based on a Complete Method for the Determination of Parallelism *Technical Report*

[8] Fox, M. and Long, D. 2001. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Technical Report.*

[9] Haslum, P. and Geffner, H. 2001. Heuristic Planning with Time and Resources. *Proc ECP-2001*

[10] Hoffmann, J. 2000. http://www.informatik.uni-freiburg.de/ hoffmann/ff.html

[11] ILOG Solver Suite. http://www.ilog.com/products/solver/

[12] Kautz, H., McAllester, D. and Selman B. Encoding Plans in Propositional Logic *In Proc. KR-96.*

[13] Ihrig, L., Kambhampati, S. Design and Implementation of a Replay Framework based on a Partial order Planner. *Proc. AAAI-96.*

[14] Kambhampati, S. & Kedar, S. 1994. An unified framework for explanation-based generalization of partially ordered and partially instantiated plans. *Artificial Intelligence Journal 67, 29-70.*

[15] Laborie, P. and Ghallab, M. Planning with sharable resource constraints. *Proc IJCAI-95*.

[16] Mali, A. Plan Merging and Plan Reuse as Satisfiability *Proc ECP-99.*

[17] Mooney, R. J. Generalizing the Order of Operators in Macro-Operators *Proc. ICML-1988*

[18] Muscettola, N. 1994. Integrating planning and scheduling. *Intelligent Scheduling.*

[19] Nguyen, X., Kambhampati, S., and Nigenda, R. 2001. Planning Graph as the Basis for deriving Heuristics for Plan Synthesis by State Space and CSP Search. *In AIJ.*

[20] Nguyen, X., and Kambhampati, S. 2001. Reviving Partial Order Planning. *Proc IJCAI-01*.

[21] Penberthy, S. and Weld, D. 1994. Planning with Continous Changes. *Proc. AAAI-94*

[22] Smith, D. & Weld, D. Temporal Planning with Mutual Exclusion Reasoning. *Proc IJCAI-99*

[23] Tsamardinos, I., Muscettola, N. and Morris, P. Fast Transformation of Temporal Plans for Efficient Execution. *Proc. AAAI-98.*

[24] Veloso, M., Perez, M, & Carbonell, J. 1990. Nonlinear planning with parallel resource allocation. *Workshop on Innovative Approaches to Planning, Scheduling and Control*.

[25] Wolfman, S. and Weld, D. 1999. The LPSAT system and its Application to Resource Planning. *In Proc. IJCAI-99.*