Integrating Planning and Scheduling In a CP Framework :
A Transition-based Approach

Abstract

Many potential real-world planning applications are on the border of planning and scheduling. To handle the complex choices of actions and temporal and resource constraints of these problems we need to integrate planning and scheduling techniques. Here we propose a transition-based formulation of temporal planning problems, that enables us to represent features like deadlines, time windows, release times etc. in a simple way. We describe a CSP encoding of the transition-based formulation and its potential advantages in integrating planning and scheduling techniques.

Motivation

Many potential real-world planning applications are on the border of planning and scheduling. For example, consider a group of robots (like NASA's Mars Rovers) that can maneuver over a designated area, perform soil tests, and can take pictures of interesting objects. To achieve a set of given goals (like testing soil of different locations and taking picture of different objects), the robots have to decide how to achieve those goals (planning choices) and also they need to achieve these goals without violating any temporal and resource constraints (scheduling choices). For any automated system, that would able to handle the complexity of this type of problems, it is essential to integrate planning and scheduling techniques.

In recent years there is a growing interest in integrating planning and scheduling to solve problems similar to the above example. Smith et al.(2000) described three possible integration approaches: separating planning and scheduling and solving them individually (Stratified P&S), interleaving planning and scheduling (Interleaved P&S), and compiling a bounded planning problem into a scheduling problem (Homogeneous P&S). Systems, like IxTeT, (Ghallab & Laruelle 1994), HSTS (Muscettola 1993), and EUROPA (Jónsson *et al.* 2000), are examples of interleaved P&S, These partial order planners resolve conflicts by making ordering decisions among the actions. CPT (Vidal & Geffner 2006), which compiles a temporal planning problem into a CSP, is an example of homogeneous P&S. One advantage of this approach is that action choice and action ordering decisions can be made in a unified fashion.

Here we propose a new homogeneous P&S approach based on a **transition-based formulation**[1] for problems, that are in between planning and scheduling, represented in the multi-valued state variable representation (Jonsson & Bäckström 1998). The advantage of this formulation is that it allows us to incorporate important features such as deadlines, time windows, release times and others, into the planning formalism in a straightforward way, and to make use of sophisticated constraint-based scheduling techniques in solving such planning problems. The main building-blocks of this formulation are **transitions**. Each transition represents an effect of an action, which can be either a change of values of a state variable or a persistent requirement on a specific value of a state variable. The motivation behind the transition-based formulation is to exploit the fact that evolution of each state variable, over a time period[2], can be represented by a sequence of transitions, which induces temporal constraints between the transitions and ordering constraints between the related actions. A solution to a planning problem, in the transition-based formulation, is a sequence of transitions for each state variable, such that goals are achieved. This can be modelled as a CSP.

The rest of the paper is organized as follows: First we describe the basics of the transition-based formulation with an example of temporal planning problem. Then we describe a CSP encoding of the transition-based formulation. We conclude the paper with a discussion about the advantages of the transition-based formulation and future directions.

Basics: Transition-based Formulation

A planning problem can be represented with 4 components: a set of state variables, a set of (grounded) actions, an initial state and a goal description. In the multi-valued state variable representation each state variable has a finite discrete domain of possible values. Each action is made up of a set of transitions, i.e. changes or persistent value requirements on a subset of state variables. We say that the action *causes* these transitions or that the transitions are *associated* with the action. The initial state description is a complete assignment of the state variables to their initial values and the goal

---

[1]Banerjee et al. (2008) reported the formulation for classical planning based on the ideas from Tuan A. Nguyen and Minh Do.

[2]this can be seen as a timeline of the state variable

description is a partial assignment of a subset of state variables to their goal values.

For a transition T, **T.var** denotes the state variable affected by the transition, **T.from** denotes the value that the transition changes, **T.to** denotes the changed value and **T.act** is the action associated with the transition. There are two types of transitions: **EFFECT** transitions and **PREVAIL** transition. An EFFECT transition T, represents a change (i.e. $T.to \neq T.form$) in the related state variable's value caused by the action $T.act$. A PREVAIL transition T, represents that the associated action requires $T.var$'s value to be same (i.e $T.from = T.to$) throughout its execution.

Domain Transition Graph (DTG)

DTG (Jonsson & Bäckström 1998) of a state variable V, is a digraph where, each node represents a value of the state variable V, and there is an edge between two nodes n_1 and n_2 labelled by a, *iff* there exists a transition T such that: $T.var = V$, $T.from = n_1$, $T.to = n_2$, and $T.act = a$. The DTG of a state variable represents all possible ways its values can be changed (possibly infinite if there are no restriction on the number of actions).

Local-Plan

A **Local-Plan** of a state variable V is a sequence of transitions, possibly empty, that forms a contiguous path in the DTG of the variable. Each local-plan induces a transitive ordering relation, "before", between the actions that are related to the transitions in the local-plan as follows:

- For each pair of EFFECT transitions T_i and T_j, such that $i < j$ and there is no other EFFECT transition in between them, induces the action ordering $T_i.act \xrightarrow{before} T_j.act$.

- For each pair of a PREVAIL transition T_p and an EFFECT transition T_e, where there is no EFFECT transition in between them, if $p < e$, then $T_p.act \xrightarrow{before} T_e.act$, otherwise $T_e.act \xrightarrow{before} T_p.act$.

Note that two PREVAIL transitions that appear in a local-plan without an EFFECT transition between them do not induce an ordering of their related actions. This is because the PREVAIL transitions represent persistent value requirements on the same value, which can be achieved without committing on any ordering between the associated actions.

A transition may need to occur multiple times in a contiguous path. This will cause cycles in the action ordering, which is not desirable. To avoid this we will assume that each local-plan is **canonical** i.e. each transition will occur at most once in a local-plan. If a transition is needed more than once in a local-plan, then we will create distinct copies of the transition.

Temporal Action Model

In the temporal model, each transition T has a start time **T.start**, an end time **T.end**, and a duration **T.duration**. Here we assume that each transition is uninterruptable (i.e. non-preemptive), so the end time of a transition must be consistent with its start time and the duration, i.e $T.end =$

$T.start + T.duration$. Each local plan induces the following temporal constraints between the transitions based on their ordering, as described in the previous section:

- For each pair of EFFECT transitions T_i and T_j, such that $i < j$ and there is no EFFECT transition in between them, $T_i.end \leq T_j.start$.

- For each pair of a PREVAIL transition T_p and an EFFECT transition T_e where there is no EFFECT transition in between them, if $p < e$, then $T_p.end \leq T_e.start$, otherwise $T_e.end \leq T_p.start$.

In this temporal model we assume that all transitions, that are caused by same action, start at same time. For each action A we associate a start time $A.start$, that synchronizes the start times of its associated transitions, i.e. for all transition T, $T.start = A.start$ where $T.act = A$. Each ordering relation between a pair of actions A_i and A_j constrains the start times of the actions as follows:

$$A_i \xrightarrow{before} A_j \Rightarrow A_i.start \leq A_j.start$$

Solution to the Transition-based Formulation

Solution of a planning problem in the transition-based formulation, can be formulated as a problem of constructing a local-plan (contiguous path in the DTG) for each state variable, such that the union of the local-plans, we will call it **Global-Plan**, will satisfy the following conditions:

1. **Initial State and Goal Achievement**: Each non-empty local-plan starts from the initial value of the state variable. If the goal condition specifies a value of a state variable, then the local-plan of the variable must end at that value.

2. **Synchronization**: If a transition is part of a local-plan for some variable, then all the related transitions, i.e transitions caused by the same action, must be part of the local-plans of their corresponding state variables.

3. **Consistency**: Ordering between the actions, induced by the set of local-plan must be globally consistent.

The final plan represents a set of actions induced by the Global-Plan. If we execute the actions according to their start times from the initial state, then it will achieve a state where goal conditions are true.

Example

Consider a simple instance of the Logistic planning domain, where we have one truck, three locations A, B and C, and a package to deliver. The only route between locations A and B is through C. Initially, the package is at A, and the truck is at C. The goal is to deliver the package to location B. State variables **loc_truck** (with values "At_A", "At_B", and "At_C"), and **loc_pack** (with values "At_A", "At_B", "At_C", and "In_Truck") represent the locations of the truck and the package, respectively. A boolean variable, **empty_tuck**, enforces that the truck can carry only one package at a time. Actions are to load/unload the package at each of the three locations, and to move the truck. Figure 1 describes the DTGs of the state variables. The start and end time of a transition is shown at the start and end of the corresponding
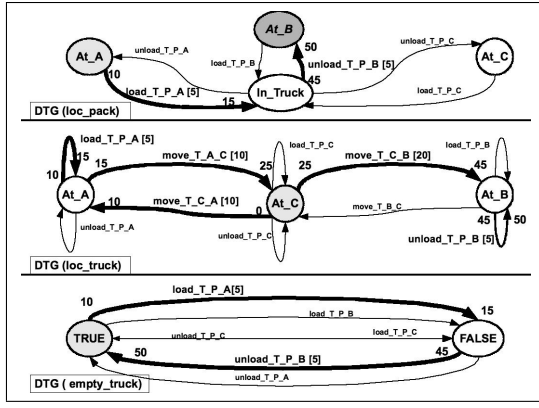
Figure 1: DTGs of the example problem

edge, respectively[3]. The bold edges in the DTGs represent the local-plans, which are temporally consistent and the ordering between the actions are globally consistent.

CSP Encoding of the Transition-based Formulation

A planning problem, in the transition-based formulation, is a problem of constructing local-plans for the state variables. To construct a local-plan for a state variable we need to decide, for each transition in the DTG, if the transition is going to be in the plan or not, and if it is going to be in the plan, then which transitions will come before and after it. In this section we describe a CSP model for a planning problem that will answer the above questions.

For each state variable, we add one **initial transition** (that changes a dummy value to the variable's initial value). If the goal condition mentions the variable, there is also a **goal transition** (that changes the goal value to a dummy value). Initial and goal transitions must be present in any plan, and are constrained to appear first and last, respectively, in the local-plans.

Bounding

To encode a transition-based formulation into a CSP we need to estimate how many times a transition, i.e. the corresponding action, can appear in the plan, because we need to make copies of that action accordingly as discussed in the previous section. The theoretical upper bound (number of possible states) is too large for solving the resultant CSP efficiently. The class of planning problems we are interested in, often need an action at most once in a solution (Vidal & Geffner 2006). This observation leads us to start with a CSP encoding where each action can occur at most once. If the encoding proved to be insoluble, then we increase the bound by one. That means, we add an extra copy of each action and search again. This process will be repeated until a solution is found. This is similar to other constraint-based planning approaches where, generally, the bound is on the makespan

---

[3]To make the picture clearer, temporal information is shown only for those transitions that are in the solution plan.

of the plan ((Kautz & Selman 1992), (Do & Kambhampati 2001)).

CSP Variables and Domains

To formulate the problem as a CSP we create the following CSP variables:

- **next[T]**: For each transition T, except for the goal transitions, next[T] represents which EFFECT transition immediately follows T. Domain of next[T] is a set of EFFECT transitions, which belongs to DTG(T.var), that can immediately *follow* T, i.e.

$$\forall T' \in Dom(next[T])$$
$$T'.from = T.to \wedge T'var = T.var$$

- **previous[T]**: For each transition T, except for the initial transitions, previous[T] represents which EFFECT transition is immediately before T. Domain of previous[T] is a set of EFFECT transitions, which belongs to DTG(T.var), that can appear immediately *before* T, i.e.

$$\forall T' \in Dom(previous[T])$$
$$T'.to = T.from \wedge T'var = T.var$$

- **inplan[A]**: For each action A, inplan[A] represents if the action A is in the plan or not. There are two possible values for inplan[A], *true* or *false*.

There are two interval variables for each transition T: **start[T]**, which represents the start time of T, and **end[T]** representing the end time of T. For each action A, an interval variable **start[A]** represents the start time of A.

The next[T] and previous[T] variables can be assigned to a not-in-plan value $\perp$, which will denote that the transition T will not be part of the final plan. The next[T] variables, where T.var is a non-goal state variable, can be assigned to a special value **IG** (Induced Goal), which will mean that T is the last transition in the Local-Plan(T.var).

Constraints

1. **EFFECT Position Constraints**: If an EFFECT transition T appears before another EFFECT transition T', then T' must appear after T and vice versa, i.e. $\forall T', T' \in$ EFFECT

$$previous[T'] = T \Leftrightarrow next[T] = T'$$

2. **PREVAIL Position Constraints**: The following constraints holds for all PREVAIL transition T_p that can appear next to an EFFECT transition T_e and before N, where $N = IG$ or N is an EFFECT transition.

$$previous[T_p] = T_e \wedge next[T_p] = N \Rightarrow next[T_e] = N$$
$$previous[T_p] = T' \wedge next[T_e] = N \Rightarrow next[T_p] = N$$
$$next[T_p] = N \wedge next[T_e] = N \Rightarrow previous[T_p] = T_e$$

3. **Action Synchronization Constraints**: If an action is in the plan then all the transitions caused by the action must also be in the plan and vice versa, i.e for all action A,

$$inplan[A] = true \Leftrightarrow \forall_{T.act=A} T : \neg(next[T] = \perp).$$

4. **Transition Exclusion Constraint**: If a transition T is excluded from the plan, then no transition can appear before or after it, i.e. for all transition T [4],

$$next[T] = \bot \Leftrightarrow \text{previous[T]} = \bot.$$

5. **Consistent Ordering Constraints**: Each assignment of next or previous variable implies the ordering of the transitions, and therefore an ordering of the associated actions, as described before. The transitive closure of the ordering between all actions in the plan must be acyclic.

6. **Start Time Synchronization Constraints:** Start times of transitions must be consistent with the start time of their corresponding actions and vice versa.

$$start[A] = \forall_{T.act=A} T : start[T]$$

7. **Temporal Position Constraints**: Each assignment of a next variable implies a temporal constraint between start and end time of the related transitions.

$$next[T] = T' \Rightarrow start[T'] \geq end[T]$$

Similar constraints apply for the assignment of previous variables.

8. **Non-preemptive Transition Constraints:** Since transitions are non-preemptive, the following condition must hold for all transition T.

$$end[T] - start[T] = T.duration$$

Each solution of the CSP, which is an assignment of the next, previous and inplan variables that satisfies all the constraints, corresponds to a solution of the planning problem, because it satisfies all the conditions of a Global-Plan described in the previous section. The final plan corresponds to the following set of tuples:

$$\mathcal{PLAN} = \{< A, A.start > \ | \ inplan[A] = \text{true}\}.$$

Discussion And Future Work

Many potential planning applications lie between planning and scheduling. We have proposed a transition-based formulation for this type of planning problems which allows us to represent temporal constraints like time-windows, goal-deadlines, release times etc. which are common in these problems, in a simple way. Since all these temporal constraints restrict when can a state variable can change to or from a particular value, they can modelled by constraining the start and end times of the transitions that represent the change. For example, if in our Logistic example the package needs to be At_B before 5 pm, this can be modelled by constraining the end time of all transition T, where T.var = loc_pack and T.from = At_B, to less than or equal to 5 pm. The transition-based formulation provides a framework where sophisticated constraint propagation techniques, like time-tabling, edge-finding etc. can be adopted to solve a problem efficiently.

To encode a planning problem as a CSP, we start with the assumption that each action can only occur at most once,

and add extra copies in case of failure. This approach is unlikely to be effective for any larger problem, because we need to perform an exhaustive search to prove that there exists no solution for the given bound. Based on the observation that in the class of problems, which we are interested in, only a small subset of the actions may need to occur more than once, we would like to investigate how can we identify this subset of actions and the number of times they will be needed. This could be done in either in a pre-processing phase or during the search where we will add additional copies of actions (and transitions) on-the-fly.

References

Banerjee, D., and Haslum, P. 2008. Planning As CSP : A Transition Based Encoding. In *ICAPS 2008 Doctoral Consortium, Sydney, Australia*.

Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence*. 132(2):151–182.

Ghallab, M., and Laruelle, H. 1994. Representation and Control in IxTeT, a Temporal Planner. In *AIPS*, 61–67.

Jonsson, P., and Bäckström, C. 1998. State-variable planning under structural restrictions: Algorithms and complexity. *Artif. Intell.* 100(1-2):125–176.

Jónsson, A. K.; Morris, P. H.; Muscettola, N.; Rajan, K.; and Smith, B. D. 2000. Planning in interplanetary space: Theory and practice. In *AIPS*, 177–186.

Kautz, H. A., and Selman, B. 1992. Planning as satisfiability. In *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, 359–363.

Muscettola, N. 1993. HSTS: Integrating planning and scheduling. Technical Report CMU-RI-TR-93-05, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Smith, D. E.; Frank, J.; and Jonsson, A. K. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15(1):47–83.

Vidal, V., and Geffner, H. 2006. Branching and pruning: An optimal temporal pocl planner based on constraint programming. *Artificial Intelligence* 170(3):298–335.

---

[4]both EFFECT and PREVAIL transitions