Integrating Planning and Scheduling: From A Resource Perspective

Debdeep Banerjee
The Australian National University and NICTA
debdeep.banerjee@rsise.anu.edu.au

Motivation and Introduction

Many real world problems include a planning component, where some choices have to be made, and a scheduling component where resources are allocated to these committed choices. Scarcity of resources makes these choices to have cascading effects on other choices. For example, consider production scheduling problem (Focacci, Laborie, & Nuijten 2000; Barták 2003) where a set of order has to be made within given deadlines. Each task of an order can be processed on a set of alternative machines, and may require additional resources such as workers, special tools etc. To make a plan, one needs to decide which task must be assigned to which machines (planning choice), and when it must be processed (scheduling choice), depending on the availability of machines, workers, and tools. Once a decision is made for a task, it will have effects on other tasks that need the same set of resources. Other examples of problems where action choices occur with complex temporal and resource constraints include, surgical case scheduling in hospitals (Pham & Klinkert 2008), vehicle routing problems (Kilby, Prosser, & Shaw 2000) and many more. Although these problems have been studied in academic literatures, complexity of constraints over choices and resources are ever increasing. For example, when assigning drivers to trucks in a VRP, complex labor laws must be respected (Goel 2009). For any automated system, that would able to handle the complexity of this type of problems, it would essential to unify planning (choices among alternatives) and scheduling (temporal and resource reasoning) techniques to be efficient.

In recent years there is a growing interest in integrating planning and scheduling to solve problems similar to the above examples. Constraint-based scheduling techniques model planning choices via allowing alternative activities (Laborie & Rogerie 2008; Barták & Čepek 2007). Planners like CPT (Vidal & Geffner 2006) and timeline-based approaches (Pralet & Verfaillie 2008; Fratini, Pecora, & Cesta 2008), compiles a bounded planning problem into a scheduling problem (Homogeneous P&S (Smith, Frank, & Jonsson 2000)), where action selection and action ordering decisions can be made in a unified fashion.

In this PhD project we would like to solve planning problems with temporal and resource constraints within a constraint-based framework. Solution to a planning problem here is a set of actions and precedence ordering between them such that all goals are achieved and none of the temporal and resource constraints are violated. To find a solution we need to decide on two things: which action should we choose to achieve (sub)goals, and if these choices create resource conflicts, then how to resolve them by applying precedence ordering between them. We specify a planning problem in a transition-based formulation (Banerjee 2009) where we represent resources explicitly and represent actions as set of effects on the state variables and resources of the planning problem. The main building-blocks of this formulation are **transitions** and **resource requirements**. Each transition represents an effect of an action on a state variable, which can be either a change of values or a persistent requirement on a specific value of the state variable. A resource requirement denotes an effect of an action on a resource. The motivation behind the transition-based formulation is to exploit the fact that the evolution of each state variable, over a time period , can be represented by a sequence of transitions, which induces temporal constraints between the transitions and ordering constraints between the related actions. Similarly, we would also like to exploit that over time resource requirements on a resource have to be partially ordered (given the capacity constraint of the resource). The advantage of this formulation is that it allows us to incorporate important features such as deadlines, time windows, release times, resource reasoning into the planning formalism in a straightforward way. Given a planning problem, we compile it to a CSP by bounding the problem with the number of occurrences of actions. There are two important aspects of this CP model: (1) we model multi-valued state variables as a special type of unary resource, which provides an unified framework for resource reasoning and (2) the bound lets us reason about not only the actions that are in the partial plan, but all possible actions that can appear in the plan in future. A solution to a planning problem, in the transition-based formulation, is a partially ordered set of actions that achieves the goal while maintaining the temporal and resource constraints, thus producing a flexible schedule.

Problem Representation

A planning problem in transition-based formulation can be represented with 4 components: a set of state variables ($\mathcal{SV}$), a set of resources ($\mathcal{R}$), a set of (grounded) actions ($\mathcal{A}$), an initial state ($\mathcal{I}$) and a goal description ($\mathcal{G}$). 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, and a set of resource requirements on a subset of resources. 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. $T.duration$ denotes the time the transition takes. There are two types of transitions: **EFFECT** transitions and **PREVAIL** transition. An EFFECT transition T, represents a change ($T.to \neq T.form$) 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 ($T.from = T.to$) throughout its execution. Similarly, for each resource requirement RR, $RR.act$ and $RR.duration$ has the same meaning as transitions, and $RR.var$ and $RR.amount$ denotes the associated resource and the quantity needed respectively. Note that, when $RR.amount > 0$, it represents consumption and when $RR.amount < 0$, it represents production. We say that the action *causes* these transitions (resource requirements) or that the transitions (resource requirements) are *associated* with the action. $\mathcal{I}$ is a complete assignment of the state variables to their initial values and $\mathcal{G}$ is a partial assignment of a subset of state variables to their goal values.

For notational simplicity from now on, we will use the term "EFFECT transition" to refer both EFFECT transitions and resource requirements, and we will use "transition" to denote any effect of actions (EFFECT and PREVAIL transitions and resource requirements).

Precedence Relations: We define a binary *precedence relation* $\rightarrow$ between two actions as $Act_1 \rightarrow Act_2$, to represent that Act_1 must be executed before Act_2. The precedence relation $\rightarrow$ is a transitive relation that is conditioned on the inclusion of the actions. That means, if $Act_1 \rightarrow Act_2$ and $Act_2 \rightarrow Act_3$, then $Act_1 \rightarrow Act_3$ *iff* Act_2 is part of the final plan.

A precedence relation between two actions induces a precedence relation between the related transitions of the actions and vice versa. That means, if T_1 and T_2 are two transitions or resource requirements on the same state variable or resource ($T_1.var = T_2.var$), then $T_1 \rightarrow T_2 \Leftrightarrow T_1.act \rightarrow T_2.act$.

As a special case of the precedence relation between two transitions, we define an "**immediate**" precedence relation, $\dashrightarrow$, where $T_1 \dashrightarrow T_2$ means that T_2 must be executed after T_1 and there exists no EFFECT transition or resource requirement T', such that $T_1 \rightarrow T'$ and $T' \rightarrow T_2$ holds.

Solution

A solution to a planning problem represented in the transition-based formulation is a set of action and consistent (acyclic) precedence constraint between them, such that goals, and all the temporal and resource constraints are satisfied. An action can occur more than once in a plan, this will cause cycles in the action ordering, which is not desirable. To avoid this we will assume each action can occur at most once in the plan, i.e. plans are **canonical**. If an action is needed more than once, then we will create distinct copies (renaming) of that action.

Resources in Planning and Scheduling

Resources are important features of the most real world problems. Actions in planning problems consumes or produce resources. Resource availability serves as a precondition of enabling actions to execute. Each resource R has a finite capacity $Cap(R)$. A **consumption profile** for R is a function over time, $ConProf(R,t) \in [level_{min}, Cap(R)]$, that denotes the total consumption (recall produce requirements have negative consumption) of the resource R at time point t.In general resource requirements on a resource R can be executed simultaneously given that at each time point t, $ConProf(R,t)$ is defined. In this work we will only consider a class of resources where resource requirements can only be executed sequentially as defined below.

- **Sequential Borrow (Unary Resources)**: A sequential borrow resource, or a unary resource, allows sequential execution of borrow resource requirements. There can be possibly infinite number of borrow requirements that can be executed on this type of resources. Example of this type of resource includes machines in job shop scheduling problems.
- **Sequential Discrete**: A sequential discrete resource allows sequential execution of finite number of consume resource requirements. One example of sequential discrete resource would be available space in a truck in a vehicle routing problem with only pick-up operation. Note that the consumption profile of a sequential discrete resources is a monotonically increasing function.
- **Sequential Reservoir**: A reservoir resource is a general class of resource model that allows execution of both consume and produce resource requirements while maintaining a consistent consumption profile. The sequential reservoir model add an extra restriction that each resource requirements must be executed sequentially. For example, available space in a truck in a vehicle routing problems with pick-up (consumes space) and delivery (produces space) operation. Note that the consumption profile of a sequential reservoir resource is a non monotonic function, it goes up when a consume resource requirement is executed, and goes down when a produce (negative consumption) resource requirement is executed.

We will use the notation $Setup(RR_1, RR_2)$ to denote the setup time between RR_1 and RR_2, when they are executed in the given order on the same resource. Each resource can be restricted to be available within specified interval. Let $TimeBound(R) = [T_{min}, T_{max}]$ denote such an interval for a resource R. Each resource requirement on R must be executed within $TimeBound(R)$.

State Variable as Unary Resource: Transitions on a state variable either change a state (EFFECT) or require a state (PREVAIL) over an interval of time. Only one EFFECT transition can be executed at any one point of time, whereas many (unrestricted) PREVAIL transitions can be executed simultaneously on a given state. In this sense a state variable without any PREVAIL transition is same as an unary resource with additional precedence relations. Given this similarity we will consider each state variable as an unary resource, with special care for PREVAIL transitions. A time bound $[T_{min}, T_{max}]$ on a state variable will mean that initial state of the state variable can only be changed after T_{min} and goal value must be achieved before T_{max}. Note that, a time bound on a state variable makes the representation of time window, deadline etc. on the state variable simpler.

From now on we will use the term "resource" to mean sequential resource, and leave the constraint modeling of the general resources (where actions can execute parallelly) as future work.

CSP Encoding

In this section we describe a CSP model for solving transition-based formulation that can be automatically derived from the problem description. This CSP encoding is similar to the CP-based path-model developed in context of job shop problems with alternative resources (Focacci, Laborie, & Nuijten 2000) and vehicle routing problems (Kilby, Prosser, & Shaw 2000), and constraint-based representation of precedence constraint (Barták & Čepek 2006).

For each state variable, we add one **initial transition** and a **goal transition** such that they must be present in any plan, and are constrained to appear first and last, respectively. If the goal condition doesn't mention the variable we add a dummy goal value and add dummy EFFECT transitions from each original value of the variable to the dummy goal value. We do the same for the resources. Here we list the CSP variables and constraints and refer to the previous work (Banerjee 2009) for more detail discussion.

CSP Variables and Constraints

- **Temporal Variables:** For each transition T, $start\_time[T]$, and $end\_time[T]$ represent the start and end time of T respectively. The constraint

$$start\_time[T] + T.duration = end\_time[T]$$

  holds for each transition T. For each action A, $start\_time[A]$ denotes the start time of A and it is synchronized with the start time of its related transitions.

- **Resource Consumption Variables:** For each resource requirement T on a sequential discrete or reservoir resource, $start\_cons[T]$ represents the total consumption of resource $T.var$ before T can start executing, and similarly $end\_cons[T]$ denotes the total consumption after T finished its execution. In other words, $start\_cons[T]$ and $end\_cons[T]$ denotes the consumption profile of $T.var$ over the time interval $start\_time[T]$ and $end\_time[T]$ respectively. Similar to the temporal variables, for each resource requirement T the following constraint holds.

$$start\_cons[T] + T.amount = end\_cons[T]$$

- **Precedence Variables:** For each transition T, $next[T]$ represents which EFFECT transitions can occur immediately *next* to T, and $previous[T]$ represents the set of EFFECT transitions that can appear immediately *before* T. An assignment $next[T] = T'$ ($\Leftrightarrow previous[T'] = T$) implies $T \dashrightarrow T'$, which means that T must be executed before T' on the resource $T.var$, which implies the following constraints,

$$start\_time[T'] \geq end\_time[T]$$
$$start\_cons[T'] \geq end\_cons[T]$$

  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. For each action A, a boolean variable $inplan[A]$ represents if A is in the plan or not. If an action is excluded from the plan, all the related transitions must be also excluded, i.e. $\forall T : T.act = A$,

$$\neg inplan[A] \Leftrightarrow next[T] = previous[T] = \perp$$

Since canonicity assumption is implicit in the transition-based formulation, we always assume that each action can occur in the plan at most once, i.e. each action is either in the plan or not in the plan. Problems that are in between planning and scheduling, often need an action at most once in a solution (Vidal & Geffner 2006) or could be modeled that way. For example, in a transportation problem move actions can be modeled as setup times between two pickup (or delivery) actions on a truck. Thus issue of making copies of move actions can be avoided. Canonicity allows us to reason with not only the actions that are included in the plan, but also with those actions that can appear in the future.

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. The final plan corresponds to the following set of tuples:

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

Example

Consider a simple vehicle routing problem where we have three packages A, B and C with load requirements 10, 20 and 30 respectively, and two trucks T1 and T2 with capacity 35. The problem is to pickup these packages within the time windows [2,12], [5,12] and [2,12] respectively where each pickup operation takes 3 units of time. Let assume that trucks are initially at a depot (D) and travel time between the location are described in Figure 1. For each package we define a state variable with two states: ready(R) and picked up (P), and a time bound based on the time window. For example, state variable for package A have time bound [2,(12+3)]. Each action represents a pickup operation of a package into a truck as decried below:

```
ACTION: T1_A
A:<ready, picked_up>:3
T1:3:10:A
```

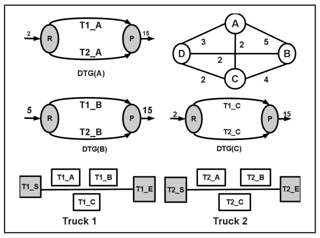

Figure 1: Example Problem

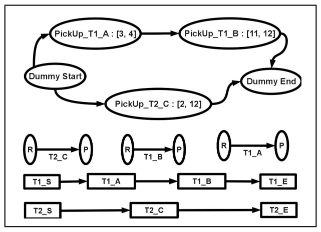

Figure 2: Solution

The second line describes that the action changes the value of the state variable A from ready to picked_up and the duration of the change is 3. The next line describes that it requires the resource T1 for 3 units of time, and consumes 10 unit of space, and its action location[1] is A. To avoid modeling explicit move actions for trucks, we model each truck as a sequential discrete resource and associate a setup time matrix to each truck that denotes the time needed for the truck to go from one action location to another. Figure 1 describes the initial state of the problem in the transition-based formulation. Figure 2 describes the solution of the problem.

Initial Results and Future Work

We have implemented a system, **TransPlan**, that reads a planning problem in transition-based formulation (similar to the example above) and converts it into a CSP and solves the CSP using the propagation and inference techniques as discussed in this paper. The first problem we have solved is meeting scheduling problem, where a set of meetings need to be scheduled among a group of teams of agents within a given time horizon. Each team consists of two team members, and exactly one member from each team can attend a meeting. Each meeting has a duration, and a team member needs an x amounts of time to travel from one meeting to other. TransPlan solves all 27 problems[2] under a second. As the second problem set, TransPlan has solved a set of jobshop problems[3] , where each problem consists of 20 jobs with due dates, where each job has 10 tasks, and each task can be executed on two alternative resources. TransPlan was able to solve all 20 problems within on average 5ms. The third set of problems that TransPlan has solved is the Solomon's instances[4] of vehicle routing problem. Each problem consists of 100 packages with time windows and load requirement, 25 trucks with capacity 100. TransPlan was able to solve all these instances on average under 6 seconds, trying to fill up one truck at a time.

Note that, the current implementation of TransPlan doesn't guarantee any quality measure on a solution. The main purpose of these experiments is to show that variety of scheduling problems with planning choices can be modeled in the transition-based formulation, and the corresponding CSPs are scalable. To find good quality solutions for these problem we need good heuristic to guide the CSP search. We consider developing such heuristics as our next step.

In addition to developing heuristics to guide the CSP search, there are 2 other directions that we would like to pursue next. The first goal is to extend our sequential resource model to more general case where resource requirements can be executed parallelly, and the second goal is to relax our canonicity assumption.

[1]An action location is the location of the associated package.

[2]These problems downloaded and modified from CSPLib `http://www.cs.st-andrews.ac.uk/~ianm/CSPLib/prob/prob046/MSP.html`

[3]These problems downloaded and modified from `http://palette.ecn.purdue.edu/~uzsoy2/Problems/classic/nomenclature.html`

[4]`http://www.sintef.no/Projectweb/TOP/Problems/VRPTW/Solomon-benchmark/`

References

Banerjee, D. 2009. Integrating planning and scheduling in a cp framework: A transition-based approach. In *Proceedings of ICAPS'09, September 19-23, Thessaloniki, Greece*.

Barták, R., and Čepek, O. 2006. Incremental maintenance of double precedence graphs: A constraint-based approach. In *Proceedings of ICAPS 06, June 6-10, Lake District, UK*.

Barták, R., and Čepek, O. 2007. Temporal networks with alternatives: Complexity and model. In *Proceedings of FLAIRS'07*.

Barták, R. 2003. Real-life manufacturing problems: A challenge. In *Proceedings of ICAPS'03 Workshop on the Competition: Impact, Organization, Evaluation, Benchmarks, Trento, Italy*.

Focacci, F.; Laborie, P.; and Nuijten, W. 2000. Solving scheduling problems with setup times and alternative resources. In *Proceedings of AIPS'00, USA*.

Fratini, S.; Pecora, F.; and Cesta, A. 2008. Unifying Planning and Scheduling as Timelines in a Component-Based Perspective. *Archives of Control Sciences* 18(2):231–271.

Goel, A. 2009. Vehicle scheduling and routing with drivers' working hours. *Transportation Science* 43(1):17–26.

Kilby, P.; Prosser, P.; and Shaw, P. 2000. A comparison of traditional and constraint-based heuristicmethods on vehicle routing problems with side constraints. *Constraints* 5(4):389–414.

Laborie, P., and Rogerie, J. 2008. Reasoning with conditional time-intervals. In *Proceedings of FLAIRS'08*.

Pham, D.-N., and Klinkert, A. 2008. Surgical case scheduling as a generalized job shop scheduling problem. *European Journal of Operational Research* 185(3):1011–1025.

Pralet, C., and Verfaillie, G. 2008. Using Constraint Networks on Timelines to Model and Solve Planning and Scheduling Problems. In *Proceedings of ICAPS'08, Sydney, Australia*.

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.