A timeline, event, and constraint-based modeling framework for planning and scheduling problems

Gérard Verfaillie and **Cédric Pralet**
ONERA - The French Aerospace Lab, F-31055, Toulouse, France
{Gerard.Verfaillie,Cedric.Pralet}@onera.fr

Abstract

This paper presents a framework dedicated to the modeling of deterministic planning and scheduling problems. This framework is based on the notions of timelines (evolutions of state variables and of resource levels), events which impact timelines, and constraints on and between events. Building it has been motivated by the observation that many real planning and scheduling problems involve a finite set of events and that the main question when solving them is to assign values to event parameters (presence in the plan, position in the event sequence, precise date, specific type-dependent parameters) by satisfying a finite set of constraints on these parameters and optimizing some function of some of these parameters. This framework, we refer to as TECK for *Timelines, Events, and Constraints Knowledge*, makes it possible to express four kinds of knowledge that must be combined when planning and scheduling: knowledge about events (event presence and parameters), time (event positions and dates), state (state variable evolutions), and resources (discrete or continuous resource evolutions).

Introduction

In classical scheduling problems, in Operations Research (OR) (Baptiste, Pape, and Nuijten 2001), the basic notions are tasks, time, and resources. The goal is to schedule a finite set of tasks by satisfying a set of temporal and resource constraints and by optimizing a given criterion, for example the time at which all tasks are performed (makespan). On the contrary, in classical planning problems, in Artificial Intelligence (AI) (Ghallab, Nau, and Traverso 2004), the basic notions are state variables, actions, and action preconditions and effects. The goal is to build a finite sequence of actions that leads the system from a given initial state to a state that satisfies some goal conditions. Similar notions of state variables, events, event preconditions (guards) and effects (transitions) are at the basis of the modeling of discrete event dynamic systems (Cassandras and Lafortune 2008) (automata, Petri nets, Markov processes). On the other hand, in more basic combinatorial optimization frameworks (linear programming, constraint programming, boolean satisfiability), the basic notions are variables and constraints. The goal is to assign values to variables in such a way that all constraints are satisfied and a given criterion is optimized.

It is well recognized that neither classical scheduling, nor classical planning frameworks can correctly model real-world problems and that most of these problems combine scheduling and planning features: need for reasoning on time and resources as in scheduling and need for reasoning on states and event preconditions and effects as in planning. In fact when looking at the planning and scheduling literature and at the many real-world problems we had to face, we get convinced that four kinds of knowledge must be expressed and handled: knowledge about *events* (how events can or must be activated), about *time* (how events are ordered and at which time they occur), about *state* (how events change state values or evolutions), and about *resources* (how events impact resource levels or evolutions). Moreover, whereas classical planning considers that there is no predefined bound on the number of actions in a plan, we can claim that many real-world planning and scheduling problems involve a *predefined finite set of events* and that the main question when solving them is to assign values to event parameters (presence in the plan, position in the event sequence, precise date, specific type-dependent parameters) by satisfying a finite set of constraints on these parameters and optimizing some function of some of these parameters.

These observations led us to the definition of a new framework dedicated to the modeling of deterministic planning and scheduling problems (certain initial state and event effects). Its main ingredients are: static variables, dynamic state variables, event types, events, constraints on events and states, and some optimization criterion.

In this paper, we present this framework, we refer to as TECK for *Timelines, Events, and Constraints Knowledge*, by using as an illustrative example the Petrobras planning and scheduling problem of ship operations for petroleum ports and platforms, which has been one of the challenges of the ICKEPS competition in 2012 (International Competition on Knowledge Engineering for Planning and Scheduling, http://icaps12.poli.usp.br/icaps12/ickeps). See (Vaquero et al. 2012) for a problem description and modeling, and (Toropila et al. 2012) for three modeling and solving approaches.

We start with a short presentation of the illustrative example we use. Then, we show how a planning domain and a planning problem are defined and what an optimal solution is in the TECK framework. Before concluding, we discuss

complexity and solving issues, and subsumed and related frameworks.

Illustrative example

The Petrobras planning and scheduling problem is a kind of logistics problem which involves a set of ports and platforms, a set of vessels, and a set of items to be delivered from ports to platforms. The goal is to deliver all items by satisfying constraints on vessel capacities and speeds and constraints on docking, undocking, loading, unloading, and refueling operations at ports or platforms, and by optimizing some combination of three criteria (makespan, fuel consumption, and docking cost). Its data is:

- a set **Po** of ports;
- a set **Pf** of platforms;
- a set **Ve** of vessels;
- a set **It** of items to be delivered;
- a set **Wa** of waiting areas;
- a subset **Pfr** $\subseteq$ **Pf** of platforms where refueling is possible; refueling is possible at all the ports and at some platforms, but not at the waiting areas;
- for each pair of locations $l_1, l_2 \in$ **Po**$\cup$**Pf**$\cup$**Wa**, a distance $\mathbf{Di}_{l_1,l_2}$ from l_1 to l_2;
- for each vessel $v \in$ **Ve**:
  - a capacity $\mathbf{Ci}_v$ (resp. $\mathbf{Cf}_v$) in terms of item weight (resp. of fuel);
  - a speed $\mathbf{Sp}_v$: distance per time unit;
  - a fuel consumption rate $\mathbf{Rfe}_v$ (resp. $\mathbf{Rfc}_v$) when v is empty (resp. not empty): fuel per distance unit;
  - a docking/undocking duration $\mathbf{Dpo}_v$ (resp. $\mathbf{Dpf}_v$) at a port (resp. at a platform);
  - a loading/unloading rate $\mathbf{Ri}_v$ at a port or platform: weight per time unit;
  - a refueling rate $\mathbf{Rpo}_v$ (resp. $\mathbf{Rpf}_v$) at a port (resp. at a platform): fuel per time unit;
  - an initial waiting area $\mathbf{Il}_v \in$ **Wa**;
  - an initial level of fuel $\mathbf{If}_v$;
- for each item $i \in$ **It**:
  - a loading location $\mathbf{Ll}_i \in$ **Po** $\cup$ **Pf**;
  - an unloading location $\mathbf{Ul}_i \in$ **Po** $\cup$ **Pf**;
  - a weight $\mathbf{We}_i$;
- a docking cost **Co** per hour at a port;
- a maximum number **Ns** of steps in the sequence of visits of each vessel: at each step, a vessel visits a port ot a platform; a port or platform can be visited several times; initial and final steps in waiting areas are excluded.

It is assumed that all vessels are initially empty in their initial waiting areas. The goal is that all items be delivered and that all vessels go empty in a waiting area, with a level of fuel that allows them at least to reach a refueling location.

For each waiting area $l \in$ **Wa** and each vessel $v \in$ **Ve**, let $\mathbf{Mf}_{l,v} = (\min_{l' \in \mathbf{Po} \cup \mathbf{Pfr}} \mathbf{Di}_{l,l'}) \cdot \mathbf{Rfe}_v$ be the minimum level of fuel allowing v to reach a refueling station from l.

Planning domain definition

We adopt the usual distinction in planning between planning domain and planning problem. In the TECK framework, a planning domain is defined by:

- a finite set **Vs** of static variables;
- a finite set **Vd** of dynamic variables;
- a finite set **De** of dependencies between dynamic variables;
- a finite set **Et** of event types.

Static variables A static variable represents a planning domain parameter that does not evolve due to successive events. To model the Petrobras problem, we introduce two sets of static variables:

- for each vessel $v \in$ **Ve**, the number $n_v \in [\![0; \mathbf{Ns}]\!]$ of steps in its sequence of visits, initial and final steps in waiting areas excluded;
- for each item $i \in$ **It**, the vessel $v_i \in$ **Ve** that transports it (it is assumed that any item is transported by a unique vessel from its loading to its unloading location, directly or not).

These variables must not be mistaken for problem data: they are really variables whose value must be decided by the planning process.

Dynamic variables A dynamic variable represents a planning domain parameter that evolves due to successive events. We distinguish dynamic variables whose value remains constant between two successive events (piecewise constant evolution) and those whose value evolves continuously with time between two successive events (linearly or not; non piecewise constant evolution). See Figure 1 for an illustration of the two kinds of evolution. To model the Petrobras problem, we introduce four sets of dynamic variables:

- for each vessel $v \in$ **Ve**, the current location $l_v \in$ **Po**$\cup$**Pf**$\cup$ **Wa**;
- for each vessel $v \in$ **Ve**, the current charge $c_v \in [0; \mathbf{Ci}_v]$ in terms of item weight;
- for each vessel $v \in$ **Ve**, the current level $f_v \in [0; \mathbf{Cf}_v]$ of fuel;
- for each item $i \in$ **It**, its status $s_i \in \{\mathbf{W}, \mathbf{T}, \mathbf{D}\}$ (**W** for waiting, **T** for transit, and **D** for delivered).

The evolution of all these variables is piecewise constant. We do not need to model the continuous evolution of the level of fuel in each vessel. However, if we should model it, the current level of fuel would be an example of dynamic variable whose evolution is not piecewise constant (linear in this case).

Dependencies Dependencies allow functional dependencies between dynamic variables to be represented: the fact that some variables functionally depend on other static or dynamic variables. In the Petrobras problem, the current

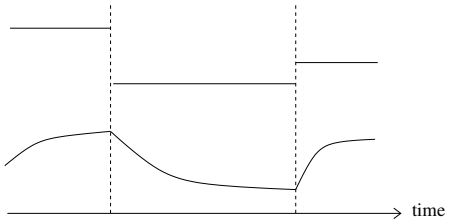

Figure 1: Piecewise constant evolution of a dynamic variable (above) and non piecewise constant evolution (below).

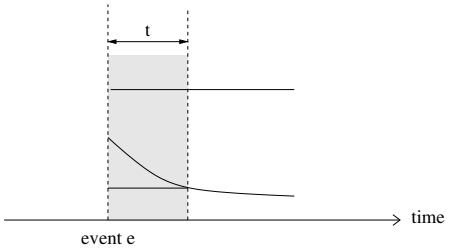

Figure 2: Value of a dynamic variable after an event in case of a piecewise constant evolution (above) and of a non piecewise constant evolution (below).

charge of a vessel v is the sum of the weights of the items that are assigned to v and currently in transit:

$$\forall v \in \mathbf{Ve} :\ c_v = \sum_{i \in \mathbf{It}} \mathbf{We}_i \cdot (v_i = v) \cdot (s_i = \mathbf{T}) \quad (1)$$

Event types With each event type *et*, are associated a finite set $\mathbf{par}(et)$ of parameters, a finite set $\mathbf{pre}(et)$ of preconditions, and a finite set $\mathbf{eff}(et)$ of effects.

Each parameter is a variable.

Each precondition is a constraint on a subset of $\mathbf{Vs} \cup \mathbf{par}(et) \cup \mathbf{Vd}$, where **Vd** represents the value of the dynamic variables just before the event.

Each effect is either (1) a functional constraint on a subset of $\mathbf{Vs} \cup \mathbf{par}(et) \cup \mathbf{Vd} \cup \mathbf{Vd'}$, where **Vd'** represents the value of the dynamic variables just after the event, which expresses the value of a dynamic variable after the event, in case of a piecewise constant evolution, or (2) a functional constraint on a subset of $\mathbf{Vs} \cup \mathbf{par}(et) \cup \mathbf{Vd} \cup \mathbf{Vd'} \cup \{t\}$, where t represents the time that went on from the event, which expresses how a dynamic variable evolves continuously after the event, in case of a non piecewise constant evolution. See Figure 2 for an illustration. Obviously, a dynamic variable cannot be the target of several dependencies or effects (unique definition) and the graph induced by dependencies and effects must be acyclic (no loop in definitions). If a variable appears in no effect of an event, its value (in case of a piecewise constant evolution) or its evolution function (in case of a non piecewise constant evolution) remains unchanged.

To model the Petrobras problem, we use six event types **StDo**, **StUl**, **StLo**, **StRf**, **StUd**, and **StTr**:

- event type **StDo** represents the starting of a docking operation at a port or platform; it has two parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$ and a location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pf}$; it has neither precondition, nor effect;
- event type **StUl** represents the starting of an unloading operation; it has three parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$, a location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pf}$, and a set of items $\mathbf{Iu} \subseteq \mathbf{It}$ to be unloaded; it has two preconditions which respectively express that one unloads all the items that are in transit in **v** and whose unloading location is **l**, and that at least one item is unloaded:

  $$\mathbf{Iu} = \{i \in \mathbf{It} \,|\, (s_i = \mathbf{T}) \wedge (v_i = \mathbf{v}) \wedge (\mathbf{Ul}_i = \mathbf{l})\} \quad (2)$$
  $$\mathbf{Iu} \neq \emptyset \quad (3)$$

  It has a set of effects, expressing that all the unloaded items are now delivered:

  $$\forall i \in \mathbf{Iu} :\ s'_i = \mathbf{D} \quad (4)$$

  The charge c_v does not appear in event effects because its new value can be inferred by dependency (see Eq. 1).
- event type **StLo** represents the starting of a loading operation; it has three parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$, a location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pf}$, and a set of items $\mathbf{Il} \subseteq \mathbf{It}$ to be loaded; it has three preconditions which respectively express that one loads only items that are waiting in **l** for loading in **v**, that at least one item is loaded, and that the vessel capacity cannot be exceeded:

  $$\mathbf{Il} \subseteq \{i \in \mathbf{It} \,|\, (s_i = \mathbf{W}) \wedge (v_i = \mathbf{v}) \wedge (\mathbf{Ll}_i = \mathbf{l})\} \quad (5)$$
  $$\mathbf{Il} \neq \emptyset \quad (6)$$
  $$c_{\mathbf{v}} + \sum_{i \in \mathbf{Il}} \mathbf{We}_i \leq \mathbf{Ci_v} \quad (7)$$

  It has a set of effects, expressing that all the loaded items are now in transit:

  $$\forall i \in \mathbf{Il} :\ s'_i = \mathbf{T} \quad (8)$$

- event type **StRf** represents the starting of a refueling operation; it has three parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$, a location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pfr}$, and a quantity $\mathbf{f} \in [0; \mathbf{Cf_v}]$ of fuel; it has two preconditions which respectively express that the quantity of fuel must be strictly positive, and that the vessel capacity cannot be exceeded:

  $$\mathbf{f} > 0 \quad (9)$$
  $$f_{\mathbf{v}} + \mathbf{f} \leq \mathbf{Cf_v} \quad (10)$$

  It has one effect on the current level of fuel:

  $$f'_{\mathbf{v}} = f_{\mathbf{v}} + \mathbf{f} \quad (11)$$

- event type **StUd** represents the starting of an undocking operation from a port or platform; it has two parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$ and a location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pf}$; it has neither precondition, nor effect;
- finally, event type **StTr** represents the starting of a transit from a location to another one; it has three parameters: a vessel $\mathbf{v} \in \mathbf{Ve}$, a starting location $\mathbf{l} \in \mathbf{Po} \cup \mathbf{Pf} \cup \mathbf{Wa}$, and a target location $\mathbf{tl} \in \mathbf{Po} \cup \mathbf{Pf} \cup \mathbf{Wa}$; it has six preconditions

which express that the starting location **l** must be the current one, the target location **tl** must be different from the starting one, there must be enough fuel to reach the target location, and, when the target location **tl** is a waiting area, all the items assigned to **v** must have been delivered and **tl** will be reached with enough fuel to reach a refueling station:

$$\mathbf{l} = l_{\mathbf{v}} \tag{12}$$
$$\mathbf{tl} \neq \mathbf{l} \tag{13}$$
$$(c_{\mathbf{v}} = 0) \rightarrow (f_{\mathbf{v}} \geq \mathbf{Di}_{\mathbf{l},\mathbf{tl}} \cdot \mathbf{Rfe}_{\mathbf{v}}) \tag{14}$$
$$(c_{\mathbf{v}} > 0) \rightarrow (f_{\mathbf{v}} \geq \mathbf{Di}_{\mathbf{l},\mathbf{tl}} \cdot \mathbf{Rfc}_{\mathbf{v}}) \tag{15}$$
$$(\mathbf{tl} \in \mathbf{Wa}) \rightarrow (\sum_{i \in \mathbf{It}} ((s_i \neq \mathbf{D}) \wedge (v_i = \mathbf{v})) = 0) \tag{16}$$
$$(\mathbf{tl} \in \mathbf{Wa}) \rightarrow (f_{\mathbf{v}} \geq \mathbf{Di}_{\mathbf{l},\mathbf{tl}} \cdot \mathbf{Rfe}_{\mathbf{v}} + \mathbf{Mf}_{\mathbf{tl},\mathbf{v}}) \tag{17}$$

On the other hand, it has three effects on the current location and the current level of fuel:

$$l'_{\mathbf{v}} = \mathbf{tl} \tag{18}$$
$$(c_{\mathbf{v}} = 0) \rightarrow (f'_{\mathbf{v}} = f_{\mathbf{v}} - \mathbf{Di}_{\mathbf{l},\mathbf{tl}} \cdot \mathbf{Rfe}_{\mathbf{v}}) \tag{19}$$
$$(c_{\mathbf{v}} > 0) \rightarrow (f'_{\mathbf{v}} = f_{\mathbf{v}} - \mathbf{Di}_{\mathbf{l},\mathbf{tl}} \cdot \mathbf{Rfc}_{\mathbf{v}}) \tag{20}$$

To model the Petrobras problem, we consider only events associated with the starting of operations. Operation durations will be taken into account via temporal constraints between events (see the next section).

Planning problem definition

In the TECK framework, a planning problem is defined by:

- a planning domain $\mathbf{Pd} = \langle \mathbf{Vs}, \mathbf{Vd}, \mathbf{De}, \mathbf{Et} \rangle$;
- an initial state **I**;
- a temporal horizon **H**;
- a finite set **E** of events;
- a finite set **Cs** of constraints on static variables;
- a finite set **Ce** of constraints on events;
- a finite set **Cd** of constraints on states;
- a criterion **Cr** to be optimized.

Initial state The initial state is defined by a finite set of initial effects. As with event effects, each initial effect is either (1) a functional constraint on a subset of **Vs** $\cup$ **Vd'**, where **Vd'** represents the value of the dynamic variables just after initialization, which expresses the value of a dynamic variable after initialization in case of a piecewise constant evolution, or (2) a functional constraint on a subset of **Vs** $\cup$ **Vd'** $\cup \{t\}$, where t represents the time that went on from initialization, which expresses how a dynamic variable evolves continuously after initialization in case of a non piecewise constant evolution. Obviously, a dynamic variable cannot be the target of several dependencies or effects (unique definition) and the graph induced by dependencies and effects must be acyclic (no loop in definitions). An initial effect must be defined for every dynamic variable that is not the target of a dependency.

In the Petrobras problem, the initial state is defined by the following equations:

$$\forall v \in \mathbf{Ve} : \ (l'_v = \mathbf{Il}_v) \wedge (f'_v = \mathbf{If}_v) \tag{21}$$
$$\forall i \in \mathbf{It} : \ s'_i = \mathbf{W} \tag{22}$$

Temporal horizon The temporal horizon is an interval of reals or integers, defined by an initial time **Ts** and a final time **Te**, equal to $+\infty$ in case of unbounded horizon.

In the Petrobras problem, the temporal horizon is the interval of reals $[0; +\infty[$.

Events Let $\mathbf{Ne} = |\mathbf{E}|$ be the finite number of events to be considered. Each event e is defined by its name $\mathbf{nam}(e)$, its type $\mathbf{typ}(e)$, its presence $\mathbf{pres}(e)$ (events may be either present or absent), its position $\mathbf{pos}(e)$ in the event sequence (all events are totally ordered), its date $\mathbf{dat}(e)$, and the values of its parameters $\mathbf{par}(e)$. Name and type are constant, whereas presence, position, date, and parameter values are variables. Type $\mathbf{typ}(e)$ is an element of **Et**. Presence $\mathbf{pres}(e)$ is a boolean (1 in case of presence and 0 otherwise). Position $\mathbf{pos}(e)$ is an integer between 1 and **Ne**. Date $\mathbf{dat}(e)$ is an element of **H**. $\mathbf{par}(e)$ is a copy of $\mathbf{par}(\mathbf{typ}(e))$: for each parameter, same name, same type, and same domain of value. When an event is absent, its position, date, and parameters take default non significant values.

To model the Petrobras problem, we associate with each vessel $v \in \mathbf{Ve}$ an event $stTr_{v,0}$ of type **StTr** which represents the first transit. Moreover, with each vessel $v \in \mathbf{Ve}$ and each step $j \in [1..\mathbf{Ns}]$, we associate six events $stDo_{v,j}$, $stUl_{v,j}$, $stLo_{v,j}$, $stRf_{v,j}$, $stUd_{v,j}$, and $stTr_{v,j}$ of respective types **StDo**, **StUl**, **StLo**, **StRf**, **StUd**, and **StTr**, which represent respectively the starting of docking, unloading, loading, refueling, undocking, and transit.

Constraints on static variables A constraint on static variables is a constraint on any subset of **Vs**.

In the Petrobras problem, we have a set of constraints on static variables that express that, if a vessel must deliver items, it must visit at least one port or platform:

$$\forall v \in \mathbf{Ve} : \ (\sum_{i \in \mathbf{It}} (v_i = v) > 0) \leftrightarrow (n_v > 0) \tag{23}$$

Constraints on events A constraint on events is a constraint on any subset of **Vs** $\cup$ $(\cup_{e \in \mathbf{E}} (\{\mathbf{pres}(e), \mathbf{pos}(e), \mathbf{dat}(e)\} \cup \mathbf{par}(e)))$. In the Petrobras problem, we have the following set of constraints on events.

Case of the unused vessels; either they do not move, or they transit to another waiting area:

$$\begin{aligned} \forall v \in \mathbf{Ve} : \quad & (n_v = 0) \rightarrow \\ & ((\mathbf{pres}(stTr_{v,0}) = 0) \vee \\ & ((\mathbf{pres}(stTr_{v,0}) = 1) \wedge \\ & (\mathbf{dat}(stTr_{v,0}) = 0) \wedge (\mathbf{v}(stTr_{v,0}) = v) \wedge \\ & (\mathbf{l}(stTr_{v,0}) = \mathbf{Il}_v) \wedge (\mathbf{tl}(stTr_{v,0}) \in \mathbf{Wa}))) \end{aligned} \tag{24}$$

Case of the used vessels; they first transit to a port or platform:

$$\forall v \in \mathbf{Ve}: \quad (n_v > 0) \rightarrow \tag{25}$$
$$((\mathbf{pres}(stTr_{v,0}) = 1) \wedge$$
$$(\mathbf{dat}(stTr_{v,0}) = 0) \wedge (\mathbf{v}(stTr_{v,0}) = v) \wedge$$
$$(\mathbf{l}(stTr_{v,0}) = \mathbf{Il}_v) \wedge (\mathbf{tl}(stTr_{v,0}) \in \mathbf{Po} \cup \mathbf{Pf}))$$

For each vessel, absence of events beyond the number of steps:

$$\forall v \in \mathbf{Ve}, \tag{26}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: \quad (j > n_v) \rightarrow$$
$$((\mathbf{pres}(stDo_{v,j}) = 0) \wedge$$
$$(\mathbf{pres}(stUl_{v,j}) = 0) \wedge$$
$$(\mathbf{pres}(stLo_{v,j}) = 0) \wedge$$
$$(\mathbf{pres}(stRf_{v,j}) = 0) \wedge$$
$$(\mathbf{pres}(stUd_{v,j}) = 0) \wedge$$
$$(\mathbf{pres}(stTr_{v,j}) = 0))$$

For each vessel, presence of events of type docking, undocking, and transit until the number of steps; at each step, docking, undocking, and transit are mandatory, but unloading, loading, and refueling are not:

$$\forall v \in \mathbf{Ve}, \tag{27}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: \quad (j \leq n_v) \rightarrow$$
$$((\mathbf{pres}(stDo_{v,j}) = 1) \wedge$$
$$(\mathbf{pres}(stUd_{v,j}) = 1) \wedge$$
$$(\mathbf{pres}(stTr_{v,j}) = 1))$$

For each vessel, presence of events of type unloading, loading, and refueling until the number of steps; at least one operation between unloading, loading, and refueling is present (one does not visit a location to do nothing):

$$\forall v \in \mathbf{Ve}, \tag{28}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: (j \leq n_v) \rightarrow$$
$$(\mathbf{pres}(stUl_{v,j}) + \mathbf{pres}(stLo_{v,j}) + \mathbf{pres}(stRf_{v,j}) > 0)$$

For each vessel and each present event, event parameters (vessel and location):

$$\forall v \in \mathbf{Ve}, \tag{29}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: (j \leq n_v) \rightarrow$$
$$((\mathbf{v}(stDo_{v,j}) = v) \wedge$$
$$((\mathbf{pres}(stUl_{v,j}) = 1) \rightarrow (\mathbf{v}(stUl_{v,j}) = v)) \wedge$$
$$((\mathbf{pres}(stLo_{v,j}) = 1) \rightarrow (\mathbf{v}(stLo_{v,j}) = v)) \wedge$$
$$((\mathbf{pres}(stRf_{v,j}) = 1) \rightarrow (\mathbf{v}(stRf_{v,j}) = v)) \wedge$$
$$(\mathbf{v}(stUd_{v,j}) = v) \wedge$$
$$(\mathbf{v}(stTr_{v,j}) = v))$$

$$\forall v \in \mathbf{Ve}, \tag{30}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: (j \leq n_v) \rightarrow$$
$$((\mathbf{l}(stDo_{v,j}) = \mathbf{tl}(stTr_{v,j-1})) \wedge$$
$$((\mathbf{pres}(stUl_{v,j}) = 1) \rightarrow (\mathbf{l}(stUl_{v,j}) = \mathbf{tl}(stTr_{v,j-1})) \wedge$$
$$((\mathbf{pres}(stLo_{v,j}) = 1) \rightarrow (\mathbf{l}(stLo_{v,j}) = \mathbf{tl}(stTr_{v,j-1}))) \wedge$$
$$((\mathbf{pres}(stRf_{v,j}) = 1) \rightarrow (\mathbf{l}(stRf_{v,j}) = \mathbf{tl}(stTr_{v,j-1})) \wedge$$
$$(\mathbf{l}(stUd_{v,j}) = \mathbf{tl}(stTr_{v,j-1})) \wedge$$
$$(\mathbf{l}(stTr_{v,j}) = \mathbf{tl}(stTr_{v,j-1})))$$

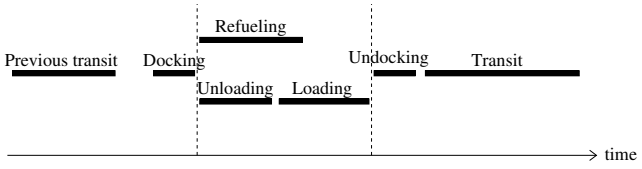

Figure 3: Temporal constraints between operations at any step.

Constraints on event dates: docking must follow transit from the previous location (we assume that one can wait at sea before docking); if present, unloading must immediately follow docking and, if present, loading must immediately follow unloading whereas, if present, refueling must immediately follow docking and be performed concurrently with unloading and loading; undocking is performed as soon as unloading, loading, and refueling are finished in order to limit docking costs; finally, transit to the next location must immediately follow undocking; see Figure 3 for a global view of temporal constraints; we present these constraints only for the starting dates of docking and unloading; the other ones can be written similarly.

Docking starting date, taking into account transit time:

$$\forall v \in \mathbf{Ve}, \tag{31}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: \quad (j \leq n_v) \rightarrow$$
$$(\mathbf{dat}(stDo_{v,j}) \geq \mathbf{dat}(stTr_{v,j-1}) +$$
$$\mathbf{Di}_{\mathbf{l}(stTr_{v,j-1}),\mathbf{tl}(stTr_{v,j-1})}/\mathbf{Sp}_v)$$

Unloading starting date, taking into account docking time:

$$\forall v \in \mathbf{Ve}, \tag{32}$$
$$\forall j \in [\![1; \mathbf{Ns}]\!]: \quad ((j \leq n_v) \wedge (\mathbf{pres}(stUl_{v,j}) = 1)) \rightarrow$$
$$(((\mathbf{l}(stDo_{v,j}) \in \mathbf{Po}) \rightarrow$$
$$(\mathbf{dat}(stUl_{v,j}) = \mathbf{dat}(stDo_{v,j}) + \mathbf{Dpo}_v)) \wedge$$
$$((\mathbf{l}(stDo_{v,j}) \in \mathbf{Pf}) \rightarrow$$
$$(\mathbf{dat}(stUl_{v,j}) = \mathbf{dat}(stDo_{v,j}) + \mathbf{Dpf}_v)))$$

Transit target locations; one transits to a waiting area only at the last step:

$$\forall v \in \mathbf{Ve}, \forall j \in [\![1; \mathbf{Ns}]\!]: \quad (j < n_v) \rightarrow \tag{33}$$
$$(\mathbf{tl}(stTr_{v,j}) \in \mathbf{Po} \cup \mathbf{Pf})$$
$$\forall v \in \mathbf{Ve}: \quad \mathbf{tl}(stTr_{v,n_v}) \in \mathbf{Wa} \tag{34}$$

At any time, no more than one vessel docked at a platform:

$$\forall v, \forall v' \in \mathbf{Ve} \,|\, v \neq v', \tag{35}$$
$$\forall j, \forall j' \in [\![1; \mathbf{Ns}]\!]: \quad ((j \leq n_v) \wedge (j' \leq n_{v'}) \wedge$$
$$(\mathbf{l}(stDo_{v,j}) \in \mathbf{Pf}) \wedge$$
$$(\mathbf{l}(stDo_{v',j'}) = \mathbf{l}(stDo_{v,j}))) \rightarrow$$
$$((\mathbf{dat}(stTr_{v,j}) \leq \mathbf{dat}(stDo_{v',j'})) \vee$$
$$(\mathbf{dat}(stTr_{v',j'}) \leq \mathbf{dat}(stDo_{v,j})))$$

At any time, no more than two vessels docked at a port:

$$\forall v \in \mathbf{Ve}, \quad (36)$$
$$\forall j \in [\![1; \mathbf{Ns}]\!] : ((j \leq n_v) \wedge (\mathbf{l}(stDo_{v,j}) \in \mathbf{Po})) \rightarrow$$
$$((\sum_{v' \in \mathbf{Ve}, j' \in [\![1;\mathbf{Ns}]\!] \mid v \neq v'} ((j' \leq n_{v'}) \wedge$$
$$(\mathbf{l}(stDo_{v',j'}) = \mathbf{l}(stDo_{v,j})) \wedge$$
$$(\mathbf{dat}(stDo_{v',j'}) \leq \mathbf{dat}(stDo_{v,j}) < \mathbf{dat}(stTr_{v',j'}))))$$
$$\leq 1)$$

Constraints on states Let **Ss** be the finite sequence of system states, function of the finite sequence of events (see the following section). For each dynamic variable $v \in \mathbf{Vd}$ and each state $s \in \mathbf{Ss}$, let $\mathbf{val}(v, s)$ be the value of v in s. A constraint on states is a constraint on any subset of $\mathbf{Vs} \cup \{\mathbf{val}(v, s) | s \in \mathbf{Ss}, v \in \mathbf{Vd}\}$. However, to model the Petrobras problem, we do not need to use any constraint on states.

Criterion The criterion **Cr** is a function of any subset of $\mathbf{Vs} \cup \{\mathbf{val}(v, s) | s \in \mathbf{Ss}, v \in \mathbf{Vd}\} \cup (\cup_{e \in \mathbf{E}}(\{\mathbf{pres}(e), \mathbf{pos}(e), \mathbf{dat}(e)\} \cup \mathbf{par}(e)))$ in a totally ordered set. In the Petrobras problem, the criterion (to be minimized) may be any combination of three criteria, which are all only function of events.

Consumed fuel quantity:

$$\sum_{v \in \mathbf{Ve}} \sum_{j=1}^{\mathbf{Ns}} \mathbf{pres}(stRf_{v,j}) \cdot \mathbf{f}(stRf_{v,j}) \quad (37)$$

Makespan (date of arrival of the last vessel at a waiting area):

$$\max_{v \in \mathbf{Ve}} \quad \mathbf{pres}(stTr_{v,n_v}) \cdot \quad (38)$$
$$(\mathbf{dat}(stTr_{v,n_v}) + \mathbf{Di}_{\mathbf{l}(stTr_{v,n_v}),\mathbf{tl}(stTr_{v,n_v})}/\mathbf{Sp}_v)$$

Docking cost:

$$\sum_{v \in \mathbf{Ve}} \sum_{j=1}^{\mathbf{Ns}} \quad \mathbf{Co} \cdot (j \leq n_v) \cdot (\mathbf{l}(stDo_{v,j}) \in \mathbf{Po}) \cdot \quad (39)$$
$$(\mathbf{dat}(stTr_{v,j}) - \mathbf{dat}(stDo_{v,j}))$$

Planning problem solution

In this section, we define the framework semantics, that is what an assignment, a solution, and an optimal solution of a planning problem are in TECK.

Assignment An assignment **A** of a planning problem $\mathbf{P} = \langle\langle \mathbf{Vs}, \mathbf{Vd}, \mathbf{De}, \mathbf{Et}\rangle, \mathbf{I}, \mathbf{H}, \mathbf{E}, \mathbf{Cs}, \mathbf{Ce}, \mathbf{Cd}, \mathbf{Cr}\rangle$ is a pair made of:

- an assignment **As** of the static variables in **Vs**;
- an assignment **Ae** of the events in **E**.

An assignment **As** of **Vs** assigns to every static variable $v \in \mathbf{Vs}$ a value from its domain. An assignment **Ae** of **E** assigns, for every event $e \in \mathbf{E}$, a value to $\mathbf{pres}(e)$ (0 or 1). If $\mathbf{pres}(e) = 1$ (present event), it assigns to $\mathbf{pos}(e)$, to $\mathbf{dat}(e)$, and to every parameter in $\mathbf{par}(e)$ a value from their respective domains.

Sequence of events We assume the following default constraints on event presences, positions, and dates:

Total order on present events:

$$\forall e \in \mathbf{E} : \quad 1 \leq \mathbf{pos}(e) \leq \sum_{e \in \mathbf{E}} \mathbf{pres}(e) \quad (40)$$
$$\forall e, e' \in \mathbf{E} \mid e \neq e' : \quad \mathbf{pos}(e) \neq \mathbf{pos}(e') \quad (41)$$

Consistency of dates with regard to positions:

$$\forall e, e' \in \mathbf{E} \mid e \neq e' : \ (\mathbf{pos}(e) < \mathbf{pos}(e')) \rightarrow (\mathbf{dat}(e) \leq \mathbf{dat}(e')) \quad (42)$$

As it is assumed in basic modeling frameworks for discrete event dynamic systems, such as automata or Petri nets, and differently from what is often assumed in classical planning, events may occur at exactly the same time, but are totally ordered.

As a consequence, an assignment **Ae** of **E** induces a finite sequence **Es** of present events, of the form $[e_1, \ldots, e_i, \ldots, e_{\mathbf{ne}}]$, where $\mathbf{ne} = \sum_{e \in \mathbf{E}} \mathbf{pres}(e)$ is the number of present events.

Sequence of states Due to initialization and event effects, an assignment **A** of a planning problem induces a finite sequence **Ss** of states, of the form $[s'_0, s_1, s'_1, \ldots, s_i, s'_i, \ldots, s_{\mathbf{ne}}, s'_{\mathbf{ne}}]$ if $\mathbf{Te} = +\infty$, and of the form $[s'_0, s_1, s'_1, \ldots, s_i, s'_i, \ldots, s_{\mathbf{ne}}, s'_{\mathbf{ne}}, s_{\mathbf{ne}+1}]$ otherwise. s'_0 is the state after initialization. For each $i \in [\![1; \mathbf{ne}]\!]$, s_i is the state just before event e_i and s'_i the state just after. If $\mathbf{Te} \neq +\infty$, $s_{\mathbf{ne}+1}$ is the state at the end of the temporal horizon. Distinguishing the state before and after an event is necessary in case of non piecewise constant evolutions of dynamic variables.

Solution An assignment $\mathbf{A} = \langle \mathbf{As}, \mathbf{Ae}\rangle$ of a planning problem $\mathbf{P} = \langle\langle \mathbf{Vs}, \mathbf{Vd}, \mathbf{De}, \mathbf{Et}\rangle, \mathbf{I}, \mathbf{H}, \mathbf{E}, \mathbf{Cs}, \mathbf{Ce}, \mathbf{Cd}, \mathbf{Cr}\rangle$ is solution if and only if:

- **As** satisfies all the constraints in **Cs**;
- **Ae** satisfies all the constraints in **Ce**, including the default constraints on event presences, positions, and dates;
- the sequence **Ss** of states induced by **A** satisfies all the constraints on domains of value of dynamic variables, all the event preconditions, and all the constraints in **Cd**.

Optimal solution A solution is optimal if and only if there is no other solution inducing a better value of **Cr**.

Complexity and subsumed frameworks

It can be shown that the decision problem associated with the TECK framework (existence or not of a solution plan) is NP-complete. It belongs to NP because checking that a given plan is solution is obviously polynomial. It is NP-complete because the CSP problem is NP-complete and any CSP p can be transformed into a TECK problem with a static part which is a copy of p and an empty dynamic part.

It can be shown that the TECK framework allows existing problems and frameworks to be modeled, such as for example:

- the RCPSP problem (*Resource Constrained Project Scheduling Problem* (Baptiste, Pape, and Nuijten 2001));
- the HTN framework (*Hierarchical Task Networks* (Nau et al. 2003)), if we assume that the tree of possible decompositions is finite;
- the STRIPS framework (Fikes and Nilsson 1971), provided that the number of actions in a plan be bounded (one does not require that the number of actions be fixed; we only need that a maximum number be defined).

Proofs of subsumption are omitted for space reasons.

Solving issues

Although this paper focuses on modeling issues, we cannot ignore that any modeling framework is the result of a difficult tradeoff between modeling issues (one wants to model as precisely as possible physical systems and user requirements) and solving issues (one wants to maintain the solving complexity as low as possible).

One can first observe that any TECK problem can be transformed into a CSP problem, more precisely into a form of dynamic CSP problem (Mittal and Falkenhainer 1990), because of the presence or absence of events. So, any constraint solver can be used to solve planning and scheduling problems expressed in the TECK framework.

This is an option, but not necessarily the best one, because it may not correctly exploit the particular structure of TECK problems. This why we are currently working on efficient local search algorithms which could be used to perform offline or online planning. Such algorithms, based on additions or removals of events and changes in event parameters, could be non chronological (choices not performed following a chronological order) and be consequently less blind and more heurisitically informed than the classical chronological planning algorithms.

Discussion and related frameworks

If we look at the TECK framework, we can first notice that it allows two kinds of constraints on plans to be expressed: (i) event preconditions and effects which are Markovian because they connect only the current state, the current event, and the next state, and (ii) constraints on events which may be non Markovian because they can connect any subset of events in the sequence of events. The first kind of constraints is usual in planning and, more generally, in the modeling of discrete event dynamic systems. The second one is less usual in planning, although it is very useful to express physical constraints, user requirements, or heuristics on plans. In the modeling of discrete event dynamic systems, temporal logics (Emerson 1990) is often used to model this kind of constraints and several works already proposed to introduce temporal logics in planning in order to express constraints on plans (Bacchus and Kabanza 2000; Kvarnström and Doherty 2001). In the TECK framework, this kind of constraints is expressed by using constraints on event parameters. Whereas event preconditions allow "what can be" to be expressed, constraints on event parameters also allow "what must be" to be expressed (for example events that must be present, possibly due to the presence of other events).

The introduction of static variables and constraints is another originality of the TECK framework. They are very useful when modeling planning problems, as it is clear in the Petrobras example. They are strangely unusual in planning, although the idea is already present in the discrete event dynamic systems community (Cimatti, Palopoli, and Ramadian 2008) with the notion of parametric timed automata.

On the other hand, it is clear that the need to associate a maximum number of events with each event type (we assume a finite number of events, present or not, each one with a fixed type) is the main limitation of the TECK framework. However, our experience in the modeling of many planning and scheduling problems in the aerospace domain and beyond allows us to claim that, in many real-world problems, it is possible to define reasonable bounds on the number of events.

With regard to classical scheduling (Baptiste, Pape, and Nuijten 2001) and to constraint programming tools that have been built to deal with scheduling problems (IBM ILOG), the TECK framework adds states, events, and state transitions. However, (Serra, Nishioka, and Marcellino 2012) uses IBM ILOG to model a Petrobras planning problem which is different from, but close to the 2012 ICKEPS competition challenge, used in this paper.

With regard to classical planning (Ghallab, Nau, and Traverso 2004) and usual planning languages (Fox and Long 2003), it adds static variables and constraints, and constraints on events. With regard to SAT or CSP-based planning approaches (Kautz and Selman 1992), it allows the presence and the position of events to be variable: for example, in the Petrobras problem, for each vessel v, at any step beyond the variable number n_v of steps, all events are absent and, at any step until n_v, docking, undocking, and transit events are present, but unloading, loading, and refueling events may be present or not (see Eqs. 26 and 27); moreover, the dates and thus the relative positions of events on different vessels are not predetermined. Finally, with regard to planning systems that are based on the common notion of timelines such as IxTeT, Europa, or APSI (Ghallab and Laruelle 1994; Frank and Jónsson 2003; Fratini, Pecora, and Cesta 2008), it offers simplicity and clear semantics.

The TECK framework inherits some features from frameworks we previously proposed. However, the CNT framework (*Constraint Networks on Timelines* (Verfaillie, Pralet, and Lemaître 2010)), based on the notion of dynamic constraint which connects a variable number of variables, was certainly too general. Moreover, it did not make any distinction between states and events. In the CTA framework (*Constraint Timed Automata* (Pralet and Verfaillie 2012)), the so-called activation constraints (events activating other events) were not clearly formalized.

Some choices may be arguable in the definition of the TECK framework, such as for example the choice of event rather than durative action as the elementary building block in the framework. We think that such a choice makes for the maximum flexibility (it is always preferable to use the most

basic elements as elementary building blocks) and that it does not prevent from building more complex blocks by using elementary ones (a durative action can be easily defined as a pair of events with the following constraints: presence (resp. absence) of action implies presence (resp. absence) of both events and presence implies a temporal distance between starting and ending events).

Finally, we think that the case where dynamic variables represent (numeric) resource levels, with events which add (subtract) quantities to (from) resource levels or slopes, represents a very important sub-framework, for which more efficient algorithms can be designed. This is what we started doing with (Pralet and Verfaillie 2013).

Conclusion and perspectives

In this paper we proposed a framework we think to be adequate for the modeling of the real-world planning and scheduling problems we encountered in the aerospace domain and beyond. Based on the notions of timelines, events, and constraints, it allows knowledge about events (event presence and parameters), time (event positions and dates), state (state variable evolutions), and resources (discrete or continuous resource evolutions) to be expressed. The next steps will consist in:

- verifying that many diverse planning and scheduling problems can effectively be modeled in the TECK framework;
- defining first useful constructs on top of the TECK basic framework, such as durative actions, temporal and resource constraints;
- exploring several solving alternatives, inspired from the background in combinatorial optimization, but exploiting the specific structure of TECK problems, with the objective to get efficient either optimal, or approximate anytime algorithms.

References

Bacchus, F., and Kabanza, F. 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence* 16:123–191.

Baptiste, P.; Pape, C. L.; and Nuijten, W. 2001. *Constraint-based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publishers.

Cassandras, C., and Lafortune, S. 2008. *Introduction to Discrete Event Systems*. Springer.

Cimatti, A.; Palopoli, L.; and Ramadian, Y. 2008. Symbolic Computation of Schedulability Regions using Parametric Timed Automata. In *Proc. of RTSS-08*, 80–89.

Emerson, E. 1990. Temporal and Modal Logic. In van Leeuwen, J., ed., *Handbook of Theoretical Computer Science, Volume B:Formal Models and Semantics*. Elsevier. 995–1072.

Fikes, R., and Nilsson, N. 1971. STRIPS: a New Approach to the Application of Theorem Proving. *Artificial Intelligence* 2:189–208.

Fox, M., and Long, D. 2003. PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20:61–124.

Frank, J., and Jónsson, A. 2003. Constraint-Based Attribute and Interval Planning. *Constraints* 8(4):339–364.

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):5–45.

Ghallab, M., and Laruelle, H. 1994. Representation and Control in IxTeT: a Temporal Planner. In *Proc. of AIPS-94*, 61–67.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

IBM ILOG. IBM ILOG CPLEX Optimization Studio. http://www-01.ibm.com/software/integration/optimization/cplex-optimization-studio/.

Kautz, H., and Selman, B. 1992. Planning as Satisfiability. In *Proc. of ECAI-92*, 359–363.

Kvarnström, J., and Doherty, P. 2001. TALplanner: A Temporal Logic Based Forward Chaining Planner. *Annals of Mathematics and Artificial Intelligence* 30:119–169.

Mittal, S., and Falkenhainer, B. 1990. Dynamic Constraint Satisfaction Problems. In *Proc. of AAAI-90*, 25–32.

Nau, D.; Au, T.; Ilghami, O.; Kuter, U.; Murdock, W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Pralet, C., and Verfaillie, G. 2013. Dynamic Online Planning and Scheduling using a Static Invariant-based Evaluation Model. In *Proc. of ICAPS-13*.

Pralet, C., and Verfaillle, G. 2012. Combining Static and Dynamic Models for Boosting Forward Planning. In *Proc. of CP-AI-OR-12*, 322–338.

Serra, T.; Nishioka, G.; and Marcellino, F. 2012. The Offshore Resources Scheduling Problem: Detailing a Constraint Programming Approach. In *Proc. of CP-12*, 823–839.

Toropila, D.; Dvoràk, F.; Trunda, O.; Hanes, M.; and Bartàk, R. 2012. Three Approaches to Solve the Petrobras Challenge. In *Proc. of ICTAI-12*, 191–198.

Vaquero, T.; Costa, G.; Tonidandel, F.; Igreja, H.; Silva, J.; and Beck, C. 2012. Planning and Scheduling Ship Operations on Petroleum Ports and Platforms. In *Proc. of the ICAPS-12 Workshop on "Scheduling and Planning Applications" (SPARK-12)*.

Verfaillie, G.; Pralet, C.; and Lemaître, M. 2010. How to Model Planning and Scheduling Problems using Timelines. *The Knowledge Engineering Review* 25(3):319–336.