A Unifying Framework for Hybrid Planning and Scheduling

Bernd Schattenberg and Susanne Biundo

Dept. of Artificial Intelligence
University of Ulm, Germany
`firstname.lastname@uni-ulm.de`

Abstract. Many real-world application domains that demand planning and scheduling support do not allow for a clear separation of these capabilities. Typically, an adequate mixture of both methodologies is required, since some aspects of the underlying planning problem imply consequences on the scheduling part and vice versa. Several integration efforts have been undertaken to couple planning and scheduling methods, most of them using separate planning and scheduling components which iteratively exchange partial solutions until both agree on a result.

This paper presents a framework that provides a uniform integration of hybrid planning –the combination of operator based partial order planning and abstraction based hierarchical task network planning– and a hierarchical scheduling approach. It is based on a proper formal account of refinement planning, which allows for the formal definition of hybrid planning, scheduling, and search strategies. In a first step, the scheduling functionality is used to produce plans that comply with time restrictions and resource bounds. We show how the resulting framework is thereby able to perform novel kinds of search strategies that opportunistically interleave what used to be separate planning and scheduling processes.

1 Introduction

Hybrid planning – the combination of hierarchical task network (HTN) planning with partial order causal link (POCL) techniques – turned out to be most appropriate for complex real-world planning applications [1] like crisis management support [2,3], etc. Here, the solution of planning problems often requires the integration of planning from first principles with the utilization of predefined plans to perform certain complex tasks. Beyond the challenge to produce complex courses of action, planning support in these domains has to consider all kinds of resources, ranging from limited time and material to power and supplies, which all define success and efficiency of the mission.

These two aspects of the planning task used to be regarded as two different kinds of problems, called *planning* and *scheduling*, one performed after the other. Newer approaches take into account, that the two problem solving phases interact to a huge extent, and neither can be reasonably carried out without knowledge and feedback about the progress of the other. This motivates to get more information out of resource analysis into the planning process, e.g. via resource profiles [4] or constraint-based techniques on shared memory [5]. It allows for some corrective steps on the "planner's side", respectively offers alternative plans to pursue. Another frequently used technique is to keep the plan and schedule generating processes completely separate, using the result of one as the input for the other. In this fashion, [6] proposes to perform classic plan generation on a relaxed problem without resource information and to give the result to a scheduler. [7] places scheduling in a pre-planning phase in order to determine necessary overlapping actions and minimal resource capacities. In all these approaches, the plan generation process is not guided by resource demands and vice versa. The IxTeT temporal planning system [8] integrates scheduling in an POCL planner by using temporally qualified expressions throughout the representation formalism, which represent state transitions and state persistences of the planning domain. The authors share our view of opportunistic scheduling as additional plan modification steps which can be interleaved with other planning steps: closing open or unachieved preconditions, resolving (resource) conflicts, and adding constraints to evade bottlenecks. An important feature is the dynamic construction of a resource hierarchy (not to be confused with hierarchical resources as introduced by [9]) based on condition analysis in the current partial plan [10]. The hierarchy represents a partial order on the "importance" of the resources for plan causality, and with that the order in which the different resources should be addressed by the reasoning process. This technique can be considered to be included as an additional strategic advice to our proposed approach.

Our aim is to provide a framework in which planning and scheduling functionality is uniformly integrated. Integration should not be limited to a planner delivering plans to be judged by a scheduler but it should be possible to generate and abandon plans schedule-driven and vice versa. We will show in a first step, what a system configuration under these requirements looks like for generating plans that are resource and time compliant.

To this end, we make use of the hybrid planning approach presented in [11]. It provides a formal framework, in which the plan generation process is functionally decomposed into well-defined flaw detecting and plan modification generating functions. We are going to show, how scheduling capabilities can be integrated and exploited in the plan generation process. *Flexible* planning and scheduling strategies operate opportunistically instead of following a fixed plan generation schema, thereby completely interleaving what used to be separate planning and scheduling processes.

The rest of the paper is organized as follows. In Section 2 we present the refinement-based planning framework on which our integrated hybrid approach relies; we define the necessary refinement operators and flaws for our purposes. Section 3 introduces the resulting system components and Section 4 describes strategies that are capable of guiding search in the integrated planner and scheduler. We conclude in Section 5 with a glimpse on future developments and some final remarks.

2 A Refinement-Based Framework

We employ a hybrid planning formalism that relies on a sorted logical language $\mathcal{L} = \{\mathcal{Z}, \mathcal{P}, \mathcal{C}, \mathcal{V}, \mathcal{O}, \mathcal{T}\}$. $\mathcal{Z}$ describes a hierarchy of sort symbols, $\mathcal{P}$ is a $\mathcal{Z}^*$-indexed set of predicate symbols, and $\mathcal{C}$ and $\mathcal{V}$ are non-empty finite $\mathcal{Z}$-indexed sets of constant symbols and variables. $\mathcal{O}$ denotes a non-empty finite set of *operator* symbols, $\mathcal{T}$ a finite set of *task* symbols; both symbol sets are $\mathcal{Z}^*$-indexed. All sets are assumed to be disjoint. $\mathcal{Z}$ contains two designated sort symbols: `Resource` and its sub-sort `Symbolic` for modeling symbolic resources. By specifying sub-sorts of `Symbolic`, we employ the notion of hierarchical resources in our approach. For presentation purposes we will focus in the following on (shareable) symbolic resources, e.g. `Vehicle` with sub-sorts `Truck` and `Jeep`. The difference between symbolic resources and other objects is subtle: the identity of a resource entity is explicitly not of interest. This allows for efficient reasoning mechanisms that analyze allocation profiles and identify bottlenecks, potential and necessary over-allocations, etc., rather than dealing with equations and un-equations in constraint sets. For a detailed discussion on hierarchical resource representations, also beyond subsumption (including numeric resources), see [9].

An *operator schema* $o(\bar{\tau}) = (\text{prec}(o(\bar{\tau})), \text{add}(o(\bar{\tau})), \text{del}(o(\bar{\tau})),\ d^{min}_{o(\bar{\tau})}, d^{max}_{o(\bar{\tau})})$ specifies the *preconditions* and the *positive* and *negative effects* of that operator ($o \in \mathcal{O}$, $\bar{\tau} = \tau_1, \ldots, \tau_n$ with $\tau_i \in \{\mathcal{C} \cup \mathcal{V}\}$ for $1 \leq i \leq n$ and n being the arity of o). Preconditions and effects are sets of literals and atoms over $\mathcal{P} \cup \mathcal{C} \cup \mathcal{V}$, respectively. Symbolic resources do not have to be allocated explicitly by specific predicates but are implicitly by their use in the condition atoms. $d^{min}_{o(\bar{\tau})}$ is the minimal duration of the operator, $d^{max}_{o(\bar{\tau})}$ the maximal.

A ground instance of an operator schema is called an *operation.* A *state* is a finite set of ground atoms and an operation $o(\bar{c})$ is *applicable* in a state s iff for the positive literals in the precondition of o: $\text{prec}^{\oplus}(o(\bar{c})) \subseteq s$ and for the negative literals: $\text{prec}^{\ominus}(o(\bar{c})) \cap s = \emptyset$. The result of applying $o(\bar{c})$ in state s is a state $s' = (s \cup \text{add}(o(\bar{c}))) \setminus \text{del}(o(\bar{c}))$. Operators are also called *primitive tasks.* Executability of sequences of operations is defined inductively.

Abstract actions are represented by *complex tasks*, which in hybrid planning show preconditions and effects like primitive ones. They are defined by *task schemata* $t(\bar{\tau}) = (\text{prec}(t(\bar{\tau})),\ \text{add}(t(\bar{\tau})),\ \text{del}(t(\bar{\tau})), d^{min}_{t(\bar{\tau})}, d^{max}_{t(\bar{\tau})})$ for $t \in \mathcal{T}$. A *decomposition method* $m = (t(\bar{\tau}), d)$ relates a complex task $t(\bar{\tau})$ to a *task network* d. This task network can be seen as a pre-defined *implementation* of a complex task and therefore the duration intervals of abstract actions have to be valid lower, respectively upper bounds for all their implementing networks. While HTN planning approaches like [12] deduce bounds for resources in abstract tasks from information about resource allocation by more primitive tasks, our notion of hybrid planning has to take into account, that implementing networks might not be complete and that the user's specification might therefore add some estimated extra time overhead, for example. Consequently, as it will be shown in the definition of the task expansion plan modification, the temporal artifacts of abstract task time bounds persist in the respective constraint sets of a plan.

For this presentation we omit axiomatic state refinements for modeling different abstraction levels on preconditions and effects for abstract task [2]. This restricts task networks occuring in methods to those in which all preconditions and effects of the abstract task have to occur explicitly.

A task network –or *partial plan*– over a language $\mathcal{L}$, a set of primitive and complex task schemata $\mathtt{T}$, and a set of decomposition methods $\mathtt{M}$, is a tuple $(TE, \prec, VC, CL, TC)$ with TE being a set of plan steps *task expressions* $te = l : t(\tau_1, \ldots, \tau_m)$, where l represents a unique label in TE and $t(\tau_1, \ldots, \tau_m)$ an instance of a task schema in $\mathtt{T}$, using variables unique in TE. $\prec$ is a set of *ordering constraints* imposing a partial order on the steps in TE. VC denotes a set of *variable constraints*, which *codesignate* and *non-codesignate* variables that occur in TE with each other or constants. It also contains sort restrictions: Constraints of the form $v \dot{\in} Z$ and $v \dot{\notin} Z$, $Z \in \mathcal{Z}$, include or exclude variables from being assigned to terms of the specified sort. CL is a set of *causal links* $te_i \xrightarrow{\phi} te_j$ with $te_i, te_j \in TE$ and ϕ being a literal with $\sigma(\phi) \in \sigma(\mathrm{prec}(te_j))$ and $\sigma(\phi) \in \sigma(\mathrm{add}(te_i))$, if ϕ is positive, and $\sigma(|\phi|) \in \sigma(\mathrm{del}(te_i))$, if ϕ is negative. σ is a VC-compatible variable substitution, i.e. a substitution that is consistent with the variable constraints. A causal link indicates that a task $(l : t_i(\bar{\tau}))$ *establishes* a precondition of a task $(l' : t_j(\bar{\tau}'))$ and is used in the usual sense as a book keeping entity.

Finally, TC represents the temporal information as a *simple temporal problem* [13]. TC is a constraint system (Z, D, C) with Z being a set of variables that represent time points and $D : Z \to R^+$ a function for assigning sets of real numbers (including the symbol ∞ for representing an infinite amount of time) to each variable in Z. The set of real numbers D_{x_i} that is assigned by D to a variable $x_i \in Z$ is called the *domain* of that variable. C is a set of unary and binary constraints. A binary constraint represents the temporal distance between two time point variables x_i and x_j by an interval $[d_{min}, d_{max}]$, which stands for the equation $d_{min} \leq x_j - x_i \leq d_{max}$. A unary temporal constraint specifies a time point x by an interval $[early, late]$, which means that $early \leq x \leq late$. The temporal network specifies for each task expression $te \in TE$ two time points that denote that beginning and the end of the action: $start_{te}$ in $[0, \infty)$ and end_{te} in $[start_{te} + d_{te}^{min}, start_{te} + d_{te}^{max}]$. For any two task expressions $te_i, te_j \in TE$ with $te_i \prec^* te_j$ (the transitive closure of $\prec$), the temporal relation $end_{te_i}^{max} \leq start_{te_j}$ holds, i.e. their temporal distance is given by the interval $[0, \infty)$. Conversely, for every two tasks with $end_{te_i}^{max} \leq start_{te_j}$, the transitive closure of the ordering relation $\prec^*$ contains $te_i \prec te_j$. A causal link $te_i \xrightarrow{\phi} te_j \in CL$ is reflected by a temporal relation $start_{te_i} \leq end_{te_j}^{max}$.

An *integrated planning and scheduling problem* $(d, \mathtt{T}, \mathtt{M}, s_{init}, s_{goal})$ consists of an initial task network d, a set of task and operator schemata $\mathtt{T}$, and a set $\mathtt{M}$ of decomposition methods for implementing the complex tasks in $\mathtt{T}$. The state s_{init} represents the initial world state, including resource capacities, and s_{goal} is a specification of the desired goal state and the overall makespan limit. Like it is common in partial order planning, we encode the initial and (optional) goal state as artificial task expressions te_{init} and te_{goal} in $TE(d)$ with respective effects and preconditions, artifacts in the temporal constraint system, and the obligation to order any other task between them.

Given a problem specification $(d, \mathtt{T}, \mathtt{M}, s_{init}, s_{goal})$, a hybrid planning and scheduling system transforms the initial task network d into a task network that is considered a *solution* to the problem. A partial plan $P = (TE, \prec, VC, CL, TC)$ is a solution to the problem, if and only if P is obtained from d by the application of plan modification steps, if TE includes only primitive task expressions, and if P is executable in s_{init} and generates s_{end}. P is called *executable* in a state s and generates a state s', if all ground linearizations of P, that means all linearizations of all ground instances of the task expressions in TE that are compatible with $\prec$ and VC, are executable in s and generate a state $s'' \supseteq s'$. No linearization may exceed the capacities specified in s_{init} in any intermediate state by its accumulated allocations, nor it may exceed the time limit specified in s_{goal} by its makespan in TC.

The presented framework makes violations of the solution criteria explicit by introducing *flaws*, data structures that literally "point" to deficiencies in the plan and allow for the problems' classification. This will allow us to guide the search process in particular to address specific problems at a specific time.

Definition 1 (Flaws). *For a given planning and scheduling problem specification and a plan P that is no solution to the problem, a flaw* $\mathtt{f}$ *is a pair $(flaw, E)$ with "$flaw$" denoting the flaw class and E being the set of components in P to which the flaw refers.*

The set of flaws is denoted by $\mathcal{F}$ with subsets $\mathcal{F}_{flaw}$ for given labels $flaw$. The set $\mathcal{F}$ of flaw classes in a partial plan $P = (TE, \prec, VC, CL, TC)$ for a given problem $(d, \mathtt{T}, \mathtt{M}, s_{init}, s_{goal})$ addresses the solution criteria by the following sub-sets (some of which being classical plan generation flaws, some related to a scheduling view):

1. $(\mathtt{AbstractTask}, \{te\})$ with $te = l : t(\bar{\tau}) \in TE, t \in \mathcal{T}$ being an abstract task expression. P is not yet primitive and this flaw class is typically associated with hybrid planning.
2. $(\mathtt{OrdInconsistency}, \{te_1, \ldots, te_k\})$ with $te_i \in TE, te_i \prec^* te_i, 1 \leq i \leq k$, i.e. if $\prec^*$ defines a cyclic partial order. There exists no defined linearization of P, which makes this flaw related to planning and scheduling likewise.
3. $(\mathtt{VarInconsistency}, \{v\})$ with $v \in \mathcal{V}$ being a variable for which $VC \models v \neq v$ holds. Since VC is inconsistent, no VC-compatible ground substitutions can be deduced to gain grounded operations from TE. This criterion is needed in both paradigms.
4. $(\mathtt{OpenVarBinding}, \{v\})$ with $v \in \mathcal{V}$ being a variable occurring in TE and there exists a constant $c \in \mathcal{C}$ with $VC \not\models v = c$ and $VC \not\models v \neq c$. The solution criterion requests all ground linearizations to be executable, therefore it has to be decided whether an operation is compatible with VC or not, for plan as well as for schedule generation.
5. $(\mathtt{OpenPrecondition}, \{te, \phi\})$ with $\phi \in \mathrm{prec}(te), te \in TE$, denotes a not fully supported task, i.e., for the subset of te-supporting causal links $\{te_i \xrightarrow{\phi_i} te | 1 \leq i \leq k\} \subseteq CL$ we find $\bigcup_{1 \leq i \leq k} \phi_i \subset \mathrm{prec}(te)$. If not all necessary

precondition establishers have yet been identified in P, some, if not all ground linearizations might not be executable. This is a typical planning-only flaw.

6. $(\texttt{OrdInconsistency}, \{te_1, te_2\})$ with $te_1, te_2 \in TE$ being causally linked task expressions, say $te_1 \xrightarrow{\phi} te_2 \in CL$, for which $te_2 \prec^* te_1$ does hold. If an identified establisher for a given precondition is ordered after the respective consumer, no linearization will be executable. Like the previous flaw, this one belongs to the area of classical planning.
7. $(\texttt{Threat}, \{te_i \xrightarrow{\phi} te_j, te_k\})$ with $te_k \not\prec^* te_i$ or $te_j \not\prec^* te_k$ and there exists a VC-compatible substitution σ such that $\sigma(\phi) \in \sigma(\text{del}(te_k))$ for positive literals ϕ and $\sigma(|\phi|) \in \sigma(\text{add}(te_k))$ for negative literals. A classical planning flaw, because te_k will corrupt executability of at least some ground linearizations.
8. Temporal constraint inconsistencies belong to scheduling and occur if intervals for time variables collapse or temporal distances become negative. The flaw structure is either $(\texttt{TempInconsistency}, \{te\})$ with $te \in TE$ being the task expression for which $start_{te}$ or end_{te} time point intervals have collapsed, or it is $(\texttt{TempInconsistency}, \{te_1, te_2\})$ with $te_1, te_2 \in TE$ being two task expressions for which the temporal distance between associated variables became negative. An overrun of the specified maximum makespan is covered by the interval for $start_{te_{goal}}$.
9. $(\texttt{SymbolicOverAllocation}, \{v_1, \ldots, v_n\})$ with $v_1, \ldots, v_n \in \mathcal{V}$ being variables for which $VC \not\models v_i = v_j$, $1 \leq i < j \leq n$ and n exceeding the capacity of any common (sub-) sort of the $v_1, \ldots, v_n$ (cf. *potential* allocations and resource profiles discussed in [9]). It is a classical scheduling aspect of the problem, that too many objects of one kind might be required at one point in time.

It can be shown that the above flaw definitions are complete in the sense, that for any given planning problem $(d, \mathtt{T}, \mathtt{M}, s_{init}, s_{goal})$ and plan P that is not flawed, P is a solution to the problem.

We now define the refinement operators for the integrated system, some of them origin in hybrid planning, some in scheduling.

Definition 2 (Plan Modifications). *For a given partial plan P over a language $\mathcal{L}$, a set of primitive and complex task schemata $\mathtt{T}$, and a set of decomposition methods $\mathtt{M}$, a plan modification $\mathtt{m}$ is a pair $(mod, E^{\oplus} \cup E_{\ominus})$. "mod" denotes the modification class, $E^{\oplus}$ and $E_{\ominus}$ are sets of elementary additions and deletions of plan components over $\mathcal{L}$, P, $\mathtt{T}$, and $\mathtt{M}$. These two sets are assumed to be disjoint and $E^{\oplus} \cup E_{\ominus} \neq \emptyset$.*

The set of all plan modifications is denoted by $\mathcal{M}$ and grouped into subsets $\mathcal{M}_{mod}$ for given classes *mod*. The application of a plan modification is characterized by the generic plan transformation function $app : \mathcal{M} \times \mathtt{P} \rightarrow \mathtt{P}$, which takes a plan modification $\mathtt{m} = (mod, E^{\oplus} \cup E_{\ominus})$ and a plan P, and returns a plan P' in which all elements of $E^{\oplus}$ have been added to and that of $E_{\ominus}$ have been removed from P.

In the integrated hybrid planning and scheduling approach, the following classes of correct plan modifications are defined for manipulating a given partial plan $P = (TE, \prec, VC, CL, TC)$ over a given language $\mathcal{L}$ and sets of task schemata $\mathtt{T}$ and expansion methods $\mathtt{M}$:

1. $(\mathtt{InsertTask}, \{\oplus te, \oplus(te \xrightarrow{\phi} te'), \oplus(v_1 = \tau_1), \ldots, \oplus(v_k = \tau_k)\})$ with $te = l : t(\bar{\tau}) \notin TE, t \in \mathtt{T}$, being a new task expression to be added, $te' \in TE$, and $\sigma'(\phi) \in \sigma'(\mathrm{add}(te))$ for positive literals ϕ, $\sigma'(|\phi|) \in \sigma'(\mathrm{del}(te))$ for negative literals, and $\sigma'(\phi) \in \sigma'(\mathrm{prec}(te'))$ with σ' being a VC'-compatible substitution for $VC' = VC \cup \{v_i = \tau_i | 1 \leq i \leq k\}$.

   We focus on symbolic resources, which cannot be "produced". Adding a new task is therefore only done for planning purposes.
2. $(\mathtt{AddOrdConstraint}, \{\oplus(te_i \prec te_j)\})$ for $te_i, te_j \in TE$. This lies in the domain of planning as well as scheduling.
3. $(\mathtt{AddVarConstraint}, \{\oplus(v = \tau)\})$ for codesignating variables $v \in \mathcal{V}$ with terms $\tau \in \mathcal{V} \cup \mathcal{C}$ and $(\mathtt{AddVarConstraint}, \{\oplus(v \neq \tau)\})$ for a corresponding non-codesignation. Cotyping and non-cotyping constraints are added by $(\mathtt{AddVarConstraint}, \{\oplus(v \dot{\in} Z)\})$ and $(\mathtt{AddVarConstraint}, \{\oplus(v \dot{\notin} Z)\})$ for $Z \in \mathcal{Z}$. Variable constraints are in the focus of plan generation as well as scheduling methods.
4. $(\mathtt{AddCausalLink}, \{\oplus(te_i \xrightarrow{\phi} te_j), \oplus(v_1 = \tau_1), \ldots, \oplus(v_k = \tau_k)\})$, with $te_i, te_j \in TE$. The codesignations represent necessary variable substitutions, such that after the modification execution, they induce a VC'-compatible substitution σ' for $VC' = VC \cup \{(v_1 = \tau_1), \ldots, (v_k = \tau_k)\}$ for which $\sigma'(\phi) \in \sigma'(\mathrm{add}(te_i))$ for positive literals ϕ, $\sigma'(|\phi|) \in \sigma'(\mathrm{del}(te_i))$ for negative literals, and $\sigma'(\phi) \in \sigma'(\mathrm{prec}(te_j))$. This treatment of causal links is originated in classical partial order planning.
5. Given an abstract task expression $te = l : t(\bar{\tau})$ in $TE, t \in \mathcal{T}$ and an expansion method $m = (t, (TE_m, \prec_m, VE_m, CL_m, TC_m))$ in $\mathtt{M}$, the expansion of te is defined as:

$$
\begin{aligned}
&(\mathtt{Expansion}, \{\ominus te\} \cup \{\oplus te_m | te_m \in TE_m\} \cup \\
&\quad \{\oplus(te_{m1} \prec te_{m2}) | (te_{m1} \prec_m te_{m2})\} \cup \\
&\quad \{\oplus(te_{m1} \xrightarrow{\phi} te_{m2}) | (te_{m1} \xrightarrow{\phi} te_{m2}) \in CL_m\} \\
&\quad \{\oplus(v = \tau) | (v = \tau) \in VC_m\} \cup \{\oplus(v \neq \tau) | (v \neq \tau) \in VC_m\} \cup \\
&\quad \{\ominus(te' \prec te), \oplus(te' \prec te_m) | (te' \prec te), te_m \in TE_m\} \cup \\
&\quad \{\ominus(te \prec te'), \oplus(te_m \prec te') | (te \prec te'), te_m \in TE_m\} \cup \\
&\quad \{\ominus(te \xrightarrow{\phi} te'), \oplus(te_m \xrightarrow{\phi} te') | (te \xrightarrow{\phi} te') \in CL, \\
&\qquad\quad te_m \in TE_m, |\phi| \in \mathrm{add}(te_m) \cup \mathrm{del}(te_m)\} \cup \\
&\quad \{\ominus(te' \xrightarrow{\phi} te), \oplus(te' \xrightarrow{\phi} te_m) | (te' \xrightarrow{\phi} te) \in CL, \\
&\qquad\quad te_m \in TE_m, \phi \in \mathrm{prec}(te_m)\} \cup \\
&\quad \{\oplus(d_{min} \leq x_j - x_i \leq d_{max}) | (d_{min} \leq x_j - x_i \leq d_{max}) \in TC_m\}
\end{aligned}
$$

   During an expansion the abstract task is replaced by the decomposition network with all its sub-tasks being ordered between the predecessors and successors of the abstract task and with all the causalities re-distributed among the appropriate sub-tasks. If the causal links cannot be re-distributed unambiguously, that means if there is more than one task in the expansion network that carries the respective precondition, one expansion modification has to be generated for each such permutation. This modification is clearly associated with (hierarchical) planning.

6. ($\texttt{AddTempConstraint}, \{\oplus(d_{min} \leq x_j - x_i \leq d_{max})\}$ adds the specified binary distance constraint $[d_{min}, d_{max}]$ between two temporal variables x_i and x_j in TC. The handling of temporal constraints is in the focus of scheduling. It is a refinement that is used to narrow the temporal distance interval between two time point variables x_i and x_j in TC. The variant ($\texttt{AddTempConstraint}, \{\oplus(d'_{min} \leq x \leq d'_{max})\}$ defines, respectively contracts, the interval for a time point x.

These plan modifications are the canonical plan transformation generators in a refinement-based planner: starting from an initial task network, the current plan can be checked against the solution criterion, and while that is not met, all refinements are applied. If no applicable modification exists, backtracking is performed. In order to make the search more systematic and efficient, the algorithm should focus on those modification steps which are appropriate to overcome the deficiencies in the current plan. Based on the formal notions of plan modifications and flaws, a generic algorithm and planning strategies can be defined. A strategy specifies *how* and *which* flaws in a partial plan are eliminated through appropriate plan modification steps. We therefore need to define the conditions under which a plan modification can *in principle* eliminate a given flaw.

Definition 3 (Appropriate Modifications). *A class of plan modifications* $\mathcal{M}_m \subseteq \mathcal{M}$ *is* appropriate *for a class of flaws* $\mathcal{F}_f \subseteq \mathcal{F}$ *iff there exist partial plans* P*, which contain flaws* $\texttt{f} \in \mathcal{F}_f$*, and modifications* $\texttt{m} \in \mathcal{M}_m$ *such that the refined plans* $P' = app(\texttt{m}, P)$ *do not contain* $\texttt{f}$*.*

The defined plan modifications perform a strict refinement, i.e., a subsequent application of them does never result in the same plan twice; the plan development is inherently a-cyclic. Given that, the same flaw cannot be re-introduced once it has been eliminated. This qualifies the appropriateness relation as a valid strategic advice for plan and schedule generation and motivates its use as the following trigger function for plan modifications:

Definition 4 (Modification Triggering Function). *Flaws in a partial plan can be removed by triggering the application of suitable plan modification steps according to the following function:* $\alpha(\mathcal{F}_x) =$

$$\begin{array}{ll}
\mathcal{M}_{\texttt{Expansion}} & \textit{if } \mathcal{F}_x = \mathcal{F}_{\texttt{AbstractTask}} \\
\mathcal{M}_{\texttt{AddVarConstraint}} & \textit{if } \mathcal{F}_x = \mathcal{F}_{\texttt{OpenVarBinding}} \\
\mathcal{M}_{\texttt{AddCausalLink}} \cup \mathcal{M}_{\texttt{InsertTask}} \cup \mathcal{M}_{\texttt{Expansion}} & \textit{if } \mathcal{F}_x = \mathcal{F}_{\texttt{OpenPrecondition}} \\
\mathcal{M}_{\texttt{Expansion}} \cup \mathcal{M}_{\texttt{AddOrdConstraint}} \cup \mathcal{M}_{\texttt{AddVarConstraint}} & \textit{if } \mathcal{F}_x = \mathcal{F}_{\texttt{Threat}} \\
\mathcal{M}_{\texttt{Expansion}} \cup \mathcal{M}_{\texttt{AddVarConstraint}} & \textit{if } \mathcal{F}_x = \mathcal{F}_{\texttt{SymbolicOverAllocation}} \\
\emptyset & \textit{else}
\end{array}$$

Modification class 7 is missing intentionally in this line-up: the manipulation of temporal constraints, $\mathcal{M}_{\texttt{AddTempConstraint}}$, is not used “actively” or as a “point of choice" for this presentation, since temporal reasoning is only performed in order to check for schedule executability. Their utilization for additional plan inferences will be shown later.

We have to omit the appropriateness proofs for the above function due to a lack of space, we would however like to sketch the arguments for two particular relationship expressed in α which displays the outstanding flexibility of the unified hybrid planning and scheduling: a) Threats of causal links can not only be addressed as usual by relocating the threatening task outside the scope of the causal link or by decoupling variable constraints. If an abstract task is involved in the threat situation, hybrid planning can alternatively make use of task expansion for producing "overlapping" task networks that may offer less strict promotion or demotion opportunities, since the causalities in the expansion network are typically linked from and to several of the introduced sub-tasks. As a side effect, the variable constraints of such a network may also rule out the threat. b) The flaws and associated modifications reflect the interplay between planning and scheduling aspects. E.g., in a plan with an abstract task, there are n different symbolic resources allocated of a given sort $Z \in \mathcal{Z}$. This implies a potential need of n such objects for every sub-sort of Z – which are not available. An expansion now concretizes a specific resource demand by co-typing constraints in its implementing task network, thereby assigning some of the allocations to one of the sub-sorts of Z, which lowers the need in other sub-sorts.

Ordering cycles $\mathcal{F}_{\texttt{OrdInconsistency}}$ and variable $\mathcal{F}_{\texttt{VarInconsistency}}$ and temporal inconsistencies $\mathcal{F}_{\texttt{TempInconsistency}}$ obviously cannot be resolved by our modifications and do therefore not trigger any modification.

3 Integrated Hybrid Planning and Scheduling

It is an important property of this approach, that the trigger function allows to completely separate the computation of flaws from that of modifications, and in turn both computations to be independent from search issues. The system architecture relies on this separation and exploits it in two ways: module invocation and interplay are specified through α while reasoning about search can be performed on the basis of flaws and modifications without taking their actual computation (or even their origin in the planning or scheduling field) into account. The issued flaws can *only* be addressed by the assigned modification generators; if none can solve the flaw, the system has to backtrack. Hence, we can map flaw and modification classes directly onto groups of modules which are responsible for their computation.

Definition 5 (Detection Modules). *A detection module x is a function that, given a partial plan P, returns all flaws of type x in P:*

$$f_x^{det} : \texttt{P} \rightarrow 2^{\mathcal{F}_x}$$

It may rank the flaws according to local priorities. E.g., $f_{\texttt{OpenPrecondition}}^{det}$ prioritizes its detections according to the number of literals in the tasks' preconditions.

Definition 6 (Modification Modules). *A modification module y is a function which computes all plan modifications of type y that are appropriate for given flaws:*

$$f_y^{mod} : \mathtt{P} \times 2^{\mathcal{F}_x} \rightarrow 2^{\mathcal{M}_y} \text{ for } \mathcal{M}_y \subseteq \alpha(\mathcal{F}_x)$$

These modules also prioritize their answers by local preferences. Priorities for modifications in $\mathcal{M}_{\mathtt{InsertTask}}$ correlate with the number of available task schemata and implied variable constraints, for example. Scheduling related modules can quantify their expected gain in plan quality or simply use their local cost estimates, e.g. preferring variable co-typing modifications in the least allocated sort.

Definition 7 (Strategy Modules). *A strategy module z is a function that selects plan modifications for their application to the current plan, possibly taking into account the detected flaws. It is defined by the projection*

$$f_z^{strat} : \mathtt{P} \times 2^{\mathcal{F}} \times 2^{\mathcal{M}} \rightarrow \mathcal{M} \cup \epsilon$$

Strategies discard a current plan P if any flaw remains un-addressed by the associated modification modules, i.e., if for any f_x^{det} and $f_{y_1}^{mod}, \ldots, f_{y_n}^{mod}$ with $\mathcal{M}_{y_1} \cup \ldots \cup \mathcal{M}_{y_n} = \alpha(\mathcal{F}_x)$:

$$\bigcup_{1 \leq i \leq n} f_{y_i}^{mod}(P, f_x^{det}(P)) = \emptyset$$

A very important consequence of the last definition is, that planning and scheduling flaws can force a backtracking at any time, in contrast to approaches where a plan has to be fully developed before it can be checked by the scheduler, etc.

In order to keep the design as flexible as possible, it is necessary to provide additional inference capabilities to the system, which may be shared by the participating modules. For all inference tasks on the plan which are not subject to choice, we define inference rules in the following way:

Definition 8 (Inference Modules). *An inference module ρ is a function that computes plan modifications of type ρ which represent necessary changes on the plan to uncover implicit information:*

$$f_\rho^{inf} : \mathtt{P} \rightarrow 2^{\mathcal{M}_\rho}$$

These inferences are used in hybrid planning to add ordering constraints between causally linked primitive tasks (cf. modification classes `InsertTask` and `AddCausalLink`, which both do not add ordering constraints in order to maintain flexibility in the case of a later overlapping of abstract task expansions). In the presented integrated hybrid planning and scheduling system, an AC-3 based constraint engine keeps the TCSP arc-B-consistent (cf. [14]), includes the implicit constraints respectively narrows down intervals, and synchronizes temporal and ordering constraints. This is done by the two specific inference modules $f_{\mathtt{AddTempConstraint}}^{inf}$ and $f_{\mathtt{AddOrdConstraint}}^{inf}$.

The following generic algorithm implements a stepwise refinement of partial plans by applying plan modifications according to detected deficiencies, i.e., flaws. It is used as the core component of any integrated planning and scheduling system that is to be implemented within our architecture:

```
**plan**($P$, $\mathtt{T}$, $\mathtt{M}$):
$F \leftarrow \emptyset$
for all $f_x^{det}$ do
  $F \leftarrow F \cup f_x^{det}(P)$
if $F = \emptyset$ then
  return $P$
$M \leftarrow \emptyset$
for all $F_x = F \cap \mathcal{F}_x$ with $F_x \neq \emptyset$ do
  answered $\leftarrow$ false
  for all $f_y^{mod}$ with $\mathcal{M}_y \subseteq \alpha(\mathcal{F}_x)$ do
    $M' \leftarrow f_y^{mod}(P, F_x)$
    if $M' \neq \emptyset$ then
      $M \leftarrow M \cup M'$
      answered $\leftarrow$ true
  if answered = false then
    return fail
return plan(infer(apply($P, f_z^{strat}(P, F, M)$)), $\mathtt{T}$, $\mathtt{M}$)
```

The procedure `infer` recursively calls all provided inference modules and applies their modifications on the plan, until no further inferences are issued.

Please note, that the algorithm is formulated independently from the deployed modules, since the options to address existing flaws by appropriate plan modifications is defined via α. The call of the strategy function z is of course the backtracking point of the system.

4 Search Strategies

The translation of existing search strategies for hybrid planning revealed that practically all of them are fixed in the sense, that they represent a preference schema on the flaw type they want to get eliminated primarily and then select appropriate modification methods. For example, it is very common to care for the plan to become primitive first and then to deal with causal interactions. A similar situation can be observed in integrated planning and scheduling systems, where the typical process is first to generate a plan and then to verify whether it can be scheduled or not. We propose the use of *flexible* strategies [11], which are capable of operating on a more general level by exploiting flaw/modification information: they are neither *flaw-dependent* as they do not primarily rely on a flaw type preference schema, nor *modification-dependent* as their do not have to be biased in favor of specific modification types. An example is the following strategy in the least commitment fashion that has proven to be very effective in the context of hybrid planning alone.

Definition 9 (Least Committing First). *Let* $\mathtt{f} \in \mathcal{F}$ *be a flaw and* $\mathtt{m}_1, \ldots, \mathtt{m}_n \in \mathcal{M}$ *a set of modifications that has been issued for the current plan. The* committing level l_c *of such a flaw is defined as follows:*

$$l_c(\mathtt{f}, \{\mathtt{m}_1, \ldots, \mathtt{m}_n\}) = \begin{cases} 0 & \text{for } n = 0 \\ 1 + l_c(\mathtt{f}, \{\mathtt{m}_1, \ldots, \mathtt{m}_{n-1}\}) & \text{for } \mathtt{m}_n \text{ answering } \mathtt{f} \\ l_c(\mathtt{f}, \{\mathtt{m}_1, \ldots, \mathtt{m}_{n-1}\}) & \text{otherwise} \end{cases}$$

The Least Committing First *strategy selects from the set of modifications those, which deal with flaws that have a minimal* l_c *value.*

$$f_{LCF}^{strat}(P, F, M) = \mathtt{m} \in \{\mathtt{m}_\mathtt{f} | \mathtt{f} \in \min(l_c(\mathtt{f}, M))$$

It can easily be seen, that this is a *flexible* strategy, since it does not depend on the actual types of issued flaws and modifications: it just compares answer set sizes in order to keep the branching in the search space low.

More importantly, it is also opportunistic with respect to planning and scheduling, since it selects whatever modification has the lowest commitment level; planning and scheduling flaws and modifications will be addressed, respectively solved, in an interleaving manner. In this way, a planning process guides the scheduling and vice versa. And if one of the two formerly separate processes finds a reason to discard the current plan, the system performs backtracking immediately.

5 Conclusions and Future Developments

We have presented a novel unifying framework and architecture for integrated planning and scheduling systems. It relies on a formal account of hybrid planning and scheduling, which allows to decouple flaw detection, modification computation, and search control. Problem solving capabilities – in this case HTN, POCL, and scheduling – can easily be combined by orchestrating respective elementary modules via an appropriate strategy module. In particular it can be configured as a classical partial order planner, an HTN planner, a resource scheduler, or any hybrid of these methods. The implemented system can be employed as a platform to implement and evaluate various planning methods and strategies. It can be easily extended to additional functionality, e.g. probabilistic reasoning [15,16], without implying changes to the deployed modules – in particular flexible strategy modules – and without jeopardizing system consistency through interfering activity.

Future work includes experimental evaluation of search strategies, e.g., the flexible HotSpot technology [11] as well as providing modification modules with local optimization techniques.

References

1. Estlin, T.A., Chien, S.A., Wang, X.: An argument for a hybrid HTN/operator-based approach to planning. In Steel, S., Alami, R., eds.: Proceedings of the 4th European Conference on Planning. Volume 1348 of LNAI., Springer (1997) 182–194

2. Biundo, S., Schattenberg, B.: From abstract crisis to concrete relief – A preliminary report on combining state abstraction and HTN planning. In Cesta, A., Borrajo, D., eds.: Proceedings of the 6th European Conference on Planning. LNCS, Springer (2001) 157–168
3. Castillo, L., Fdez-Olivares, J., González, A.: On the adequacy of hierarchical planning characteristics for real-world problem solving. In: Proceedings of the 6th European Conference on Planning. (2001)
4. Drabble, B., Tate, A.: The use of optimistic and pessimistic resource profiles to inform search in an activity based planner. In Hammond, K., ed.: Proceedings of the 2nd International Conference on Artificial Intelligence Planning Systems, AAAI (1994) 243–248
5. Garrido, A., Salido, M.A., Barber, F.: Scheduling in a planning environment. In Sauer, J., Köhler, J., eds.: Proceedings of the 14th European Conference on Artificial Intelligence Workshop on New Results in Planning, Scheduling and Design. (2000) 36–43
6. Srivastava, B., Kambhampati, S.: Scaling up planning by teasing out resource scheduling. In Biundo, S., Fox, M., eds.: Proceedings of the 5th European Conference on Planning. Volume 1809 of LNCS., Springer (2000) 172–186
7. El-Kholy, A., Richards, B.: Temporal and resource reasoning in planning: The parcPLAN approach. In Wahlster, W., ed.: Proceedings of the 12th European Conference on Artificial Intelligence, John Wiley & Sons (1996) 614–618
8. Laborie, P., Ghallab, M.: Planning with sharable resource constraints. In Mellish, C.S., ed.: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1995) 1643–1651
9. Schattenberg, B., Biundo, S.: On the identification and use of hierarchical resources in planning and scheduling. In Ghallab, M., Hertzberg, J., Traverso, P., eds.: Proceedings of the 6th International Conference on Artificial Intelligence Planning Systems, AAAI (2002) 263–272
10. Garcia, F., Laborie, P.: Hierarchisation of the seach space in temporal planning. In Ghallab, M., Milani, A., eds.: New Directions in AI Planning, Proceedings of the 3rd European Workshop on AI Planning. Volume 31 of Frontiers in Artificial Intelligence., IOS Press (1996) 217–232
11. Schattenberg, B., Weigl, A., Biundo, S.: Hybrid planning using flexible strategies. In Furbach, U., ed.: Proceedings of the 28th German Conference on Artificial Intelligence. Volume 3698 of LNAI., Springer (2005) 258–272
12. Clement, B.J., Barrett, A.C., Rabideau, G.R., Durfee, E.H.: Using abstraction in planning and scheduling. In Cesta, A., Borrajo, D., eds.: Proceedings of the 6th European Conference on Planning. LNCS, Springer (2001) 145–156
13. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial Intelligence **49** (1991) 61–91
14. Lhomme, O.: Consistency techniques for numeric CSPs. In Bajcsy, R., ed.: Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann (1993) 232–238
15. Biundo, S., Holzer, R., Schattenberg, B.: Dealing with continuous resources in AI planning. In: Proceedings of the 4th International Workshop on Planning and Scheduling for Space (IWPSS'04), European Space Agency Publications Division (2004) 213–218
16. Biundo, S., Holzer, R., Schattenberg, B.: Project planning under temporal uncertainty. In Castillo, L., Borrajo, D., Salido, M.A., Oddi, A., eds.: Planning, Scheduling, and Constraint Satisfaction: From Theory to Practice. Volume 117 of Frontiers in Artificial Intelligence and Applications. IOS Press (2005) 189–198