On the Construction and Evaluation of Flexible Plan-Refinement Strategies

Bernd Schattenberg, Julien Bidot*, and Susanne Biundo

Institute for Artificial Intelligence
Ulm University, Germany
firstname.lastname@uni-ulm.de

Abstract. This paper describes a system for the systematic construction and evaluation of planning strategies. It is based on a proper formal account of refinement planning and allows to decouple plan-deficiency detection, refinement computation, and search control. In adopting this methodology, planning strategies can be explicitly described and easily deployed in various system configurations.
We introduce novel domain-independent planning strategies that are applicable to a wide range of planning capabilities and methods. These so-called *HotSpot* strategies are guided by information about current plan defects and solution options. The results of a first empirical performance evaluation are presented in the context of hybrid planning.

1 Introduction

In hybrid planning – the combination of hierarchical-task-network (HTN) planning with partial-order causal link (POCL) techniques – the solution of planning problems requires the integration of plan synthesis based on primitive operations with the use of predefined plans that implement abstract actions. As a consequence, causal reasoning and task expansion interleave during the plan-development process.

So far, only few research has been devoted to search strategies in hybrid planning; nor have existing strategies ever been systematically compared and assessed in this context. For most of today's planning systems, this is due to the fact that search strategies are only implicitly defined and indirectly controlled via parameters. In these frameworks, search strategies and planning algorithms are strongly interdependent and changing one of them inevitably affects the other.

Furthermore, all these search strategies are basically fixed; i.e., in each plan-generation cycle, they follow a pre-defined schema indicating at which kind of deficiency to look first, and which changes to apply to the plan preferably. In POCL planning, Joslin & Pollack proposed Least-Cost Flaw Repair, a method that repairs first the flaws associated with the minimal number of modifications [1]. Peot & Smith presented strategies that consist of delaying the treatment

* This work has been supported by a grant from the Ministry of Science, Research and the Arts of Baden-Württemberg (Az: 23-7532.24-14-19)

of causal threats for which multiple solutions exist [2]. Their results have been picked up by Schubert & Gerevini who proposed simple but effective plan metrics for an A^* heuristic [3]. In HTN planning, Tsuneto et al. described the *fewest-alternatives first* heuristic for selecting task expansions in the UMCP system [4]. McCluskey introduced the *Expand then Make Sound* method [5] in which the expansion of an abstract task is followed by repairing the plan's causal structure.

For studying search strategies, we use the hybrid planning approach presented in Schattenberg et al. [6]. It provides a formal framework in which the plan generation process is decomposed into well-defined functions for flaw detection and the generation of plan modifications, respectively. Furthermore, the approach decouples search strategies from planning algorithms. The strategies are quite flexible and can be applied to large varieties of system configurations. Recently, they have been integrated into a planning-and-scheduling system [7].

2 Refinement-based Planning

We employ a hybrid planning formalism based on sorted first-order logic [6].

An operator or primitive *task schema* $t(\overline{\tau}) = (\mathrm{prec}(t(\overline{\tau})), \mathrm{add}(t(\overline{\tau})), \mathrm{del}(t(\overline{\tau})))$ specifies the *preconditions* and the *positive* and *negative effects* of the task. Preconditions and effects are sets of literals, $\bar{\tau} = \tau_1, \ldots, \tau_n$ are the task parameters. A ground instance of a task schema is called an *operation*. A *state* is a finite set of ground atoms, and an operation $t(\bar{c})$ is called *applicable* in a state s, if the positive literals of $\mathrm{prec}(t(\bar{c}))$ are in s and the negative are not. The result of applying the operation in a state s is a state $s' = (s \cup \mathrm{add}(t(\bar{c}))) \setminus \mathrm{del}(t(\bar{c}))$. The applicability of sequences of operations is defined inductively over state sequences as usual.

Abstract actions are represented by *complex tasks*. In hybrid planning these show preconditions and effects exactly like the primitive tasks. The associated state transition semantics is based on axiomatic state refinements that relate task preconditions and effects across various abstraction levels [8].

A *partial plan* or *task network* is a tuple $P = (TE, \prec, VC, CL)$ with the following sets of *plan components*: TE is a set of plan steps or *task expressions* $te = l : t(\overline{\tau})$ where l is unique label and $t(\overline{\tau})$ is a (partially) instantiated task. $\prec$ is a set of *ordering constraints* that impose a partial order on the plan steps in TE. VC is a set of *variable constraints* $v \doteq \tau$ and $v \dot{\neq} \tau$, which *codesignate* and *non-codesignate* variables occurring in TE with variables or constants. Furthermore, it contains *sort restrictions* of the form $v \dot{\in} Z$ and $v \dot{\notin} Z$, where Z is a sort symbol, that restrict further codesignations. CL is a set of *causal links* $\langle te_i, \phi, te_j \rangle$, indicating that a task te_i *establishes* a precondition of task te_j. Causal links are used in the usual sense as a book-keeping means for maintaining the causal structure.

Task networks are also used as pre-defined *implementations* of complex tasks. An *expansion method* $em = (t(\overline{\tau}), (TE_{em}, \prec_{em}, VC_{em}, CL_{em}))$ relates such a complex task schema $t(\overline{\tau})$ to a task network.

A domain model for hybrid planning is specified by $D = (\mathtt{T}, \mathtt{EM})$, that is a set of task schemata `T` and a set `EM` of expansion methods for implementing

the complex tasks in $\mathtt{T}$. Please note that there are in general multiple methods provided for each complex task schema.

A *hybrid planning problem* is the structure $\pi = (D, s_{init}, s_{goal}, P_{init})$. It consists of a domain model D and an initial task network P_{init}. A partial plan $P = (TE, \prec, VC, CL)$ is a solution to a problem specification, if and only if TE consists of primitive task expressions only, and P is executable in s_{init} and generates a state s_{end} such that $s_{goal} \subseteq s_{end}$ holds. Plan P is thereby 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 $s'' \supseteq s'$.

In our framework, violations of the solution criteria are made explicit by so-called *flaws*, data structures that represent critical components of a partial plan, i.e. those that prevent the plan from being a solution. The flaws serve to classify sub-problems and to guide the search for a solution.

Definition 1 (Flaw). *Given a problem specification π and a partial plan P that is not a solution to π, a* flaw $\mathtt{f} = \{\alpha_1, \ldots, \alpha_n\}$ *consists of a set of plan components α_i of P. It represents a defect in which these components are involved.*

The set of all flaws is denoted by $\mathtt{F}$ and subsets $\mathtt{F_x}$ represent classes of flaws. For a partial plan $P = (TE, \prec, VC, CL)$ the violation of the solution criteria is mirrored by flaw classes such as the following. The class $\mathtt{F_{CausalThreat}}$, e.g., contains flaws $\mathtt{f} = \{\langle te_i, \phi, te_j \rangle, te_k\}$ describing causal threats, i.e., indicating that a task te_k is possibly being ordered between plan steps i and j ($te_k \not\prec^* te_i$ and $te_j \not\prec^* te_k$) and there exists a variable substitution σ that is consistent with the equations and in-equations imposed by the variable constraints in VC such that $\sigma(\phi) \in \sigma(\mathrm{del}(te_k))$ for positive literals ϕ and $\sigma(|\phi|) \in \sigma(\mathrm{add}(te_k))$ for negative literals. This means, the presence of te_k in P as it stands will possibly corrupt the executability of at least some ground linearizations of P.

Flaw classes also cover the presence of abstract actions in the plan, ordering and variable constraint inconsistencies, unsupported preconditions of actions, etc. The complete class definitions can be found in [6]. It can be shown that these flaw definitions are complete in the sense that for any given planning problem π and plan P that is not flawed, P is a solution to π.

For the definition of the refinement operators, we make use of an explicit representation of change to the plan data structure: plan modifications. This representation has two major impacts on the hybrid planning framework: First, it makes changes to the plan analyzable and with that the available options for a search strategy become comparable qualitatively and quantitatively. Second, new functionality can instantly be integrated as soon as it is translated into the plan modification structure.

Definition 2 (Plan Modification). *For a given partial plan P and domain model D, a* plan modification *is defined as the structure $\mathtt{m} = (E^{\oplus}, E^{\ominus})$. $E^{\oplus}$ and $E^{\ominus}$ are sets of elementary additions and deletions of plan components over P and D.*

$E^{\oplus}$ and $E^{\ominus}$ are assumed to be disjoint and $E^{\oplus} \cup E^{\ominus} \neq \emptyset$. Furthermore, we require plan modifications to be consistent in the sense that all elements in the deletion set are components of plan P and all elements in the addition set are new components.

The set of all plan modifications is denoted by $\mathtt{M}$ and grouped into modification classes $\mathtt{M_y}$. The application of plan modifications is characterized by the generic plan transformation function $app : \mathtt{M} \times \mathtt{P} \rightarrow \mathtt{P}$, which takes a plan modification $\mathtt{m} = (E^{\oplus}, E^{\ominus})$ and a plan P, and returns a plan P' that is obtained from P by adding all elements of $E^{\oplus}$ and removing those in $E^{\ominus}$. As an example, the class $\mathtt{M_{AddCL}}$ contains plan modifications $\mathtt{m} = (\{\langle te_i, \phi, te_j \rangle, v_1 \doteq \tau_1, \ldots, v_k \doteq \tau_k\}^{\oplus}, \{\}^{\ominus})$ for manipulating a given partial plan $P = (TE, \prec, VC, CL)$ by adding causal links. The plan steps te_i and te_j are in such a modification in TE and the codesignations represent variable substitutions. They induce a VC'-compatible substitution σ' with $VC' = VC \cup \{v_1 \doteq \tau_1, \ldots, v_k \doteq \tau_k\}$ such that $\sigma'(\phi) \in \sigma'(\mathrm{add}(te_i))$ for positive literals ϕ, $\sigma'(|\phi|) \in \sigma'(\mathrm{del}(te_i))$ for negative literals, and $\sigma'(\phi) \in \sigma'(\mathrm{prec}(te_j))$.

The complete collection of refinement operations for hybrid planning is introduced in [6]. This also covers the expansion of abstract plan steps and the insertion of new plan steps, ordering constraints, and variable (in-) equations. These plan modifications are the canonical plan transformation generators in a refinement-based hybrid planner: starting from an initial task network, the current plan is checked against the solution criteria, and while these are not met, refinements are applied. If no applicable modification exists, backtracking is performed. In order to make the search 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.

A class of plan modifications $\mathtt{M_y} \subseteq \mathtt{M}$ is called *appropriate* for a class of flaws $\mathtt{F_x} \subseteq \mathtt{F}$, if and only if there exist partial plans P which contain flaws $\mathtt{f} \in \mathtt{F_x}$ and modifications $\mathtt{m} \in \mathtt{M_y}$, such that the refined plans $P' = app(\mathtt{m}, P)$ do not contain these flaws anymore.

It is easy to see that the plan modifications perform a strict refinement, that means, a subsequent application of any modification instances cannot result in the same plan twice; the plan development is inherently acyclic. Given that, any flaw instance cannot be re-introduced once it has been eliminated. This qualifies the appropriateness relation as a valid strategic advice for the plan generation process and motivates its use as the trigger function for plan modifications: the α modification triggering function relates flaw classes with their potentially solving modification classes. As an example, causal threat flaws can be solved by expanding abstract actions which are involved in the threat (by overlapping task implementations), by promotion or demotion, or by separating variables through inequality constraints [8]: $\alpha(\mathtt{F_{CausalThreat}}) = \mathtt{M_{ExpTask}} \cup \mathtt{M_{AddOrdCstr}} \cup \mathtt{M_{AddVarCstr}}$.

Please note, that α states nothing about the relationship between the actual flaw and modification instances.

The modification triggering function allows for a simple plan generation schema: based on the flaws that are found in the current plan, the calculation of plan modifications is motivated. A strategy component can then freely choose among all applicable plan modifications in order to try to solve one of the particular problems that have been identified. If any flaw remains unanswered by triggered modification generating entities, no appropriate refinement candidate exists and the plan can be discarded. There are even flaw classes that do not trigger any modification. Flaw instances that represent ordering cycles and variable inconsistencies obviously cannot be resolved by our refinement operators and do therefore not trigger any modification: $\alpha(\mathtt{F}_{\mathtt{OrdInconst}}) = \alpha(\mathtt{F}_{\mathtt{VarInconst}}) = \emptyset$.

The above mentioned triggering function completely separates the computation of flaws from the computation of modifications, and in turn both computations are independent from search-related considerations. The system architecture relies on this separation and exploits it in two ways: module invocation and interplay are specified through the α-trigger, while reasoning about search can be performed on the basis of flaws and modifications without taking their actual computation into account. Hence, we map flaw and modification classes directly onto groups of modules which are responsible for their computation.

Definition 3 (Detection Modules). *A detection module x is a function that, given a partial plan P, a domain model D, and a problem specification π, returns all flaws of type x that are present in the plan:* $f_x^{det} : \mathtt{P} \times \mathtt{D} \times \Pi \rightarrow 2^{\mathtt{F_x}}$.

Definition 4 (Modification Modules). *A modification module y is a function which computes all plan modifications of type y that are applicable to a given plan P and that are appropriate for given flaws with respect to a given domain model:* $f_y^{mod} : \mathtt{P} \times 2^{\mathtt{F_x}} \times \mathtt{D} \rightarrow 2^{\mathtt{M_y}}$ *for* $\mathtt{M_y} \subseteq \alpha(\mathtt{F_x})$.

Please note that plan modifications carry a reference to the flaw instance they address, i.e., any plan modification is unambiguously linked with its triggering flaw.

While the plan deficiency detectors and the refinement generators provide the basic plan generation functionality, strategy functions can be designed for reasoning about which paths in the refinement space to pursue. To this end, we split up reasoning about search into two compartments: The first component is an option evaluation that is performed in the local view of the currently processed plan; it reasons about the detected flaws and proposed refinements in the current plan and assesses the modifications. The second component is responsible for the global view on the refinement space and evaluates the alternative search options.

We begin with the definition of a strategic function that selects all plan modifications that are considered to be worthwhile options, thereby determining the ordered set of successors for the current plan in the plan refinement space. In doing so, the following function also determines the branching behavior of the upcoming refinement-based planning algorithm.

Definition 5 (Modification-Selection Module). *Given a plan, a set of flaws, and a set of plan modifications, a* modification-selection module *is a function* $f^{modSel} : \mathtt{P} \times 2^{\mathtt{F}} \times 2^{\mathtt{M}} \rightarrow 2^{\mathtt{M} \times \mathtt{M}}$ *that selects some (or all) of the plan modifications and returns them in a partial order for application to the passed plan.*

Strategies discard a plan P, if any flaw remains unaddressed by the associated modification modules. That means, we reject any plan P, over any domain model D and for any planning problem π, if for any f_x^{det} and $f_{y_1}^{mod}, \ldots, f_{y_n}^{mod}$ with $\mathtt{M}_{\mathtt{y}_1} \cup \ldots \cup \mathtt{M}_{\mathtt{y}_n} = \alpha(\mathtt{F}_{\mathtt{x}})$ the following holds: $\bigcup_{1 \leq i \leq n} f_{y_i}^{mod}(P, D, f_x^{det}(P, D, \pi)) = \emptyset$.

The second aspect of search control concerns the selection of those plans that are to be processed next by the detection and modification modules. These unassessed partial plans, the leaves of the search tree, are usually called the *fringe*. In other words, concrete implementations of the following module are responsible for the general search schema, ranging from uninformed procedures such as depth-first, breadth-first, etc., to informed, heuristic schemata.

Definition 6 (Plan-Selection Module). *A* plan-selection module *is a function that returns a partial order of plans for a given sequence of plans. It is described as* $f^{planSel} : \mathtt{P}^* \rightarrow 2^{\mathtt{P} \times \mathtt{P}}$.

Building on the components defined so far we can now assemble a hybrid planning system. A software artifact that implements the generic refinement algorithm (Alg. 1) is making the flaw detection and modification generating modules operational by stepwise collecting plan deficiencies, collecting appropriate modifications, selecting worthwhile modifications, and finally selecting the next plan in the fringe of the search tree. 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.

3 Search Strategies

A large number of search strategies can be realized in the proposed refinement-planning framework by sequencing the respective selection modules and using the returned partially ordered sets of modifications, respectively plans, to modulate preceding decisions: if the primary strategy does not prefer one option over the other, the secondary strategy is followed, and so on, until finally a random preference is assumed. The following single-objective heuristics constitute the building blocks for more sophisticated strategy compositions.

Unflexible Strategies: Previous work has presented the translation of some existing search strategies into our hybrid planning framework and showed that most of them are unflexible in the sense that they represent a fixed 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.

Algorithm 1 The generic refinement planning algorithm.

Require: Sets of modules $\mathfrak{Det} = \{f_1^{det}, \ldots, f_d^{det}\}$ and $\mathfrak{Mod} = \{f_1^{mod}, \ldots, f_m^{mod}\}$
Require: Selection modules f^{modSel} and $f^{planSel}$

plan$(P_1 \ldots P_n, D, \pi)$**:** {P_1 is the plan that is worked on}
if $n = 0$ **then**
  return failure
$P \leftarrow P_1$; Fringe $\leftarrow P_2 \ldots P_n$
$F \leftarrow \emptyset$
for all $f_x^{det} \in \mathfrak{Det}$ **do** {Flaw detection}
  $F \leftarrow F \cup f_x^{det}(P, D, \pi)$
if $F = \emptyset$ **then**
  return P
$M \leftarrow \emptyset$
for $x = 1$ to d **do** {Modification generation}
  $F_x = F \cap \mathtt{F_x}$ {Process flaws class-wise as returned by corresponding f_x^{det}}
  for all $\mathtt{f} \in F_x$ **do**
    for all $f_y^{mod} \in \mathfrak{Mod}$ with $\mathtt{M_y} \subseteq \alpha(\mathtt{F_x})$ **do**
      $M \leftarrow M \cup f_y^{mod}(P, \mathtt{f}, D)$
    if $\mathtt{f}$ was un-addressed **then**
      $P_{\text{next}} \leftarrow f^{planSel}$(Fringe)
      return plan$(P_{\text{next}} \circ (\text{Fringe} \setminus P_{\text{next}}), D, \pi)$
for all $\mathtt{m}$ in *linearize*$(f^{modSel}(P, F, M))$ **do** {Strategic choices}
  Fringe $\leftarrow$ *app*$(\mathtt{m}, P) \circ$ Fringe
$P_{\text{next}} \leftarrow$ *first*(*linearize*$(f^{planSel}$(Fringe)))
return plan$(P_{\text{next}} \circ (\text{Fringe} \setminus P_{\text{next}}), D, \pi)$

We will not recapitulate the results obtained in [6], but rather give a short, systematic overview over the possible instances of unflexible selection schemas for modification and plan selection.

A traditional form of modification selection is either to prefer or to disfavor categorically specific classes of plan modifications; e.g., we prefer the expansion of tasks to task insertions or we try to avoid an assignment of variables to constants "as long as possible." Systems with built-in strategies of this kind are typically provided with hand-tailored heuristics such that certain plan properties modulate the preference schema to some extent, but for now we focus on purely modification-based forms of decisions.

In the presented framework, the preference of a modification class is encoded as follows: Let $\mathtt{M_{Pref}}$ be the preferred modification class by modification-selection module f_{PrefMC}^{modSel}, then the corresponding function is defined as

$$\mathtt{m}_i < \mathtt{m}_j \in f_{PrefMC}^{modSel}(P, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, \{\mathtt{m}_1, \ldots, \mathtt{m}_n\}) \text{ if } \mathtt{m}_i \in \mathtt{M_{Pref}} \text{ and } \mathtt{m}_j \notin \mathtt{M_{Pref}}$$

for all plans $P \in \mathtt{P}$, sets of flaws $\mathtt{f}_1, \ldots, \mathtt{f}_m \in \mathtt{F}$, and sets of plan modifications $\mathtt{m}_1, \ldots, \mathtt{m}_n \in \mathtt{M}$. Avoiding the selection (and consequently the application) of instances of particular modification classes is realized by an analogue selection module that is returning the inverse of the above definition: f_{DisfMC}^{modSel}.

Plan selection can also be based upon the availability of refinement options for a plan. For simplification, the presented algorithm performs three consecutive sections: one loop for the flaw detection, one loop for the modification generation, and finally the strategic choices. In the little more complicated implementation of the framework, after applying the selected plan modifications to the current plan (line 20), for every new plan in the fringe, flaws and modifications are calculated in advance, so that the following two plan-selection modules can be defined for the preference of a modification class:

$$P_i < P_j \in f^{planSel}_{PrefMC}(\{P_1, \ldots, P_n\}) \text{ if } |\, mods(P_i) \cap \mathtt{M_{Pref}}| > |\, mods(P_j) \cap \mathtt{M_{Pref}}|$$

The function $mods : \mathtt{P} \rightarrow 2^{\mathtt{M}}$ thereby returns all modifications that have been published for a given plan. This plan selection can be formulated in relation to the plan size; e.g., it is related to the number of plan steps:

$$P_i < P_j \in f^{planSel}_{PrefMRatio}(\{P_1, \ldots, P_n\}) \text{ if } \frac{|\, mods(P_i) \cap \mathtt{M_{Pref}}|}{|TE_i|} > \frac{|\, mods(P_j) \cap \mathtt{M_{Pref}}|}{|TE_j|}$$

The plan selection can also be defined inversely, thereby evading partial plans for which undesired refinement options are available.

Strategies that concentrate on the flaw situation rather than on the available refinement options are more "problem-oriented" and are typical for agenda-based planning algorithms, which collect the plan defects and then decide which one to tackle first. Let $\mathtt{F_{Pref}}$ be the preferred class of flaws and let $modFor : \mathtt{F} \times \mathtt{P} \rightarrow 2^{\mathtt{M}}$ be the function that returns all modifications that have been answered to a given flaw in a given plan. A modification strategy module for preferring $\mathtt{F_{Pref}}$ is then defined as:

$$\begin{aligned} &\mathtt{m}_i < \mathtt{m}_j \in f^{modSel}_{AddrFC}(P, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, \{\mathtt{m}_1, \ldots, \mathtt{m}_n\}) \\ &\text{if } 1 \leq x_i, x_j \leq m,\ 1 \leq i, j \leq n,\ \mathtt{m}_i \in modFor(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in modFor(\mathtt{f}_{x_j}, P) \\ &\quad \text{and } \mathtt{f}_{x_i} \in \mathtt{F_{Pref}},\ \mathtt{f}_{x_j} \notin \mathtt{F_{Pref}} \end{aligned}$$

Like for the modification-dependent strategies, a flaw avoiding modification-selection module can be set up, and also corresponding absolute and relative plan selections are available.

Flexible Modification-Selection Strategies: *Flexible* strategies 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 they do not have to be biased in favor of specific modification types. A representative is the modification-selection strategy *Least Committing First (LCF)*. It works according to the least-commitment principle and has proven to be very successful in the context of hybrid planning [6]. It is a generalized variant of *Least-Cost Flaw Repair* [1] and selects those modifications that address flaws for which the smallest number of

alternative modifications has been found:

$$\begin{aligned}&\mathtt{m}_i < \mathtt{m}_j \in f_{LCF}^{modSel}(P, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, \{\mathtt{m}_1, \ldots, \mathtt{m}_n\})\\ &\text{if } 1 \leq x_i, x_j \leq m,\ 1 \leq i, j \leq n,\ \mathtt{m}_i \in \mathit{modFor}(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in \mathit{modFor}(\mathtt{f}_{x_j}, P)\\ &\quad \text{and } |\mathit{modFor}(\mathtt{f}_{x_i}, P)| < |\mathit{modFor}(\mathtt{f}_{x_j}, P)|\end{aligned}$$

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.

The so-called *HotSpot-based modification-selection* strategies have their origin in the observation that the least-commitment principle is in practice a very efficient search schema, although it does not take the actual structure of the compared options into account. It is, for example, not very intuitive to say that choosing one of two large task expansions represents less commitment than choosing one of three variable codesignations. The expansion obviously makes considerably more changes to the plan structure, and we expect that the "amount of change" has side effects on the overall flaw and modification availability in the future. Lifting the least-commitment schema on the plan structure level led to the development of the HotSpot notion, that is plan components that are referred to by multiple flaws and modifications [6]. We are now going to extend this strategy family by new modification-selection modules and will later introduce plan selection that is based on HotSpot calculations.

We begin with flaw-based modification selections, that means plan modifications are chosen based on the number of cross-references the addressed flaw has with other flaws. The simplest form of a HotSpot is the number of references to shared plan components. The following module avoids modifications that address flaws that contain frequently referenced components:

$$\begin{aligned}&\mathtt{m}_i < \mathtt{m}_j \in f_{DirUniHS}^{modSel}(P, F, M)\\ &\text{if } \mathtt{f}_{x_i}, \mathtt{f}_{x_j} \in F,\ \mathtt{m}_i \in M \cap \mathit{modFor}(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in M \cap \mathit{modFor}(\mathtt{f}_{x_j}, P)\\ &\quad \text{and } \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_i}} |\mathtt{f} \cap \mathtt{f}_{x_i}| < \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_j}} |\mathtt{f} \cap \mathtt{f}_{x_j}|\end{aligned}$$

Direct HotSpot calculations offer a first insight into inter-flaw relationships but do not capture relatively trivial indirect dependencies. We therefore introduce the notion of plan *sub-components* which are the structures that constitute plan components. E.g., a task expression is a component of a plan, but may also be a sub-component of an ordering constraint. In order to represent HotSpot elements on the sub-component level, the modification selection has to be adapted as follows: Given a function *comp* that returns all sub-components of a set of plan components, we define the indirect uniform HotSpot selection as the function:

$$\begin{aligned}&\mathtt{m}_i < \mathtt{m}_j \in f_{IndUniHS}^{modSel}(P, F, M)\\ &\text{if } \mathtt{f}_{x_i}, \mathtt{f}_{x_j} \in F,\ \mathtt{m}_i \in M \cap \mathit{modFor}(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in M \cap \mathit{modFor}(\mathtt{f}_{x_j}, P)\\ &\quad \text{and } \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_i}} |\mathit{comp}(\mathtt{f}) \cap \mathit{comp}(\mathtt{f}_{x_i})| < \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_j}} |\mathit{comp}(\mathtt{f}) \cap \mathit{comp}(\mathtt{f}_{x_j})|\end{aligned}$$

The extension of the HotSpot focus on sub-components is that of finding "transitive relationships" between HotSpot components on the component and sub-component level: HotZones. The idea is to build an initial map of all HotSpots in a plan, and then to re-evaluate each individual HotSpot according to the HotSpots in its neighborhood:

$$\begin{aligned}&\mathtt{m}_i < \mathtt{m}_j \in f^{modSel}_{HZone}(P, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, \{\mathtt{m}_1, \ldots, \mathtt{m}_n\})\\ &\text{if } 1 \leq x_i, x_j \leq m,\ 1 \leq i, j \leq n,\ \mathtt{m}_i \in \mathit{modFor}(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in \mathit{modFor}(\mathtt{f}_{x_j}, P)\\ &\quad \text{and } h(\mathtt{f}_{x_i}, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, 0) < h(\mathtt{f}_{x_j}, \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}, 0)\end{aligned}$$

The recursive evaluation of the HotSpot value with respect to the HotSpot values of flaws that share components is defined as follows (constant parameter θ modulates the slope of the decay in the influence neighbors' values with increasing distance d):

$$h(\mathtt{f}, F, d) = \begin{cases} 0 & \text{for } F = \emptyset \\ \displaystyle\sum_{\mathtt{f}_i \in F \setminus \mathtt{f}} |\mathtt{f} \cap \mathtt{f}_i| \cdot e^{-\frac{d}{\theta}} + h(\mathtt{f}_i, F \setminus \{\mathtt{f}, \mathtt{f}_i\}, d+1) & \text{else} \end{cases}$$

The HotSpot strategies defined so far solely focused on the flaws' commonalities. As already suggested in [6], overlappings in plan modifications can also be analyzed for commitment estimates. A direct HotSpot calculation is of course possible, however not very promising, since elementary additions never share components directly (all of them are new) and deletion overlappings are seldom (only occur in alternative expansions). We therefore define the modification-selection module $f^{modSel}_{ModBasedHS}$ analogously to the indirect uniform HotSpot function such that it treats the respective domain model entities as sub-components of the referenced components (by doing so, two task insertion modifications overlap if they share the same task schema). A second element of the heuristic is that the influence of transitive references decays (like in the HotZone calculation) because of the strong interdependencies of the domain model components. We refer to the experimental results, however omit this module due to lack of space.

In first experiments with learning strategies, we found in adaptive HotSpot calculation a straightforward enhancement of the uniform calculations. The adaptive variant tries to predict the conflict potential between two flaw classes in the current refinement space exploration and modulates the calculated HotSpot values with that estimate.

$$\begin{aligned}&\mathtt{m}_i < \mathtt{m}_j \in f^{modSel}_{DirAdaptHS}(P, F, M)\\ &\text{if } \mathtt{f}_{x_i}, \mathtt{f}_{x_j} \in F,\ \mathtt{m}_i \in M \cap \mathit{modFor}(\mathtt{f}_{x_i}, P),\ \mathtt{m}_j \in M \cap \mathit{modFor}(\mathtt{f}_{x_j}, P)\\ &\quad \text{and } \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_i}} |\mathtt{f} \cap \mathtt{f}_{x_i}| \cdot g(\mathtt{f}, \mathtt{f}_{x_i}) < \sum_{\mathtt{f} \in F \setminus \mathtt{f}_{x_j}} |\mathtt{f} \cap \mathtt{f}_{x_j}| \cdot g(\mathtt{f}, \mathtt{f}_{x_j})\end{aligned}$$

Modification-selection module $f^{modSel}_{DirAdaptHS}$ works exactly like the uniform module except every summand is multiplied with the estimate function $g : \mathtt{F} \times \mathtt{F} \rightarrow \mathbb{R}$

which is defined as follows:

$$g(\mathtt{f}_a, \mathtt{f}_b) = \frac{count(\mathtt{F_a}, \mathtt{F_b})}{\max_{\mathtt{f}_x, \mathtt{f}_y \in \mathtt{F}}(count(\mathtt{f}_x, \mathtt{f}_y))} \cdot g_{user}(\mathtt{f}_a, \mathtt{f}_b) \text{ for } \mathtt{f}_a \in \mathtt{F_a}, \mathtt{f}_b \in \mathtt{F_b}$$

Function *count* records the accumulated amount of HotSpot overlappings between the corresponding flaw classes according to the results of the $f^{modSel}_{DirUniHS}$ computations that have been obtained in the current plan generation and normalizes the value with the current maximum of all count records. The second function g_{user} is a user-defined modulation table in which certain conflict weights can be set. This mechanism allows us to mark, e.g., a HotSpot relationship between an open precondition flaw and an abstract task flaw to be less critical than the one between an open precondition and a causal threat.

Like for the uniform selections, the direct adaptive modification selection can also be formulated as an indirect working module with calls to the respective indirect uniform HotSpot calculations.

Flexible Plan-Selection Strategies: The simplest way of selecting plans is choosing the first or last plan in the fringe, which are the conventional uninformed plan-selection modules $f^{planSel}_{First}$ for a depth-first and $f^{planSel}_{Last}$ for a breadth-first search scheme. If a "depth-first-flavor" is to be added in a combination of strategies, the $f^{planSel}_{LongerHistoryFirst}$ selection prefers plans that have been obtained by a longer sequence of modification applications (the inverse strategy favors leaves of shorter refinement paths). Other simple plan metrics that can be used to estimate the stage of development for a given plan are based on the size of the constraint sets and in particular on the number of plan steps.

An effective plan-selection module that is based on plan metrics is the following:

$$P_i < P_j \in f^{planSel}_{ConstrPlans}(P_1, \ldots, P_n) \text{ if } \frac{(|\prec_i| + |VC_i| + |CL_i|) \cdot |TE_j|}{|TE_i| \cdot (|\prec_j| + |VC_j| + |CL_j|)} > 1$$

The idea behind this selection function is to focus on plans that are more constrained than others and that are therefore more likely to either get completed or turn out a failure. Since processed plans tend to get processed again, it is more of a depth-first search but it is less vulnerable to getting trapped in useless task-insertion paths (for solely inserting tasks reduces the heuristic value).

In the fixed-strategy section, we already mentioned flaw- and modification-based plan metrics. More general forms of that schema compare the relative sizes of detected flaw sets or those of proposed modification sets:

$$P_i < P_j \in f^{planSel}_{LessFlawsPerTask}(P_1, \ldots, P_n) \text{ if } \frac{|\mathit{flaws}(P_i)|}{|TE_i|} < \frac{|\mathit{flaws}(P_j)|}{|TE_j|}$$

As our experiments have shown, the heuristic to develop plans that have less flaws per plan step is a very effective estimate (although not admissible in general). Moreover, plan metrics such as $\mathit{flaws} : \mathtt{P} \rightarrow \mathbb{N}$ or the size of the plan component sets are computationally extremely cheap.

The last selection functions is related to an A^*-heuristic proposed in [3], which is used as a UCPOP plan-selection. In our framework, such a strategy is modelled as follows:

$$P_i < P_j \in f^{planSel}_{S+OC}(P_1, \ldots, P_n) \text{ if } \frac{|TE_i| + |\mathit{flaws}(P_i) \cap \mathtt{F_{OpenPrec}}|}{|TE_j| + |\mathit{flaws}(P_j) \cap \mathtt{F_{OpenPrec}}|} < 1$$

For plan selection, a set of HotSpot-based metrics can be derived from the presented modification-selection functions. We will briefly describe the ideas because their arithmetics are presented above. The first two strategies consider the flaw-based HotSpot situation in the plans. Plan selections $f^{planSel}_{DirUniHS}$ and $f^{planSel}_{IndUniHS}$ accumulate the direct, respectively indirect HotSpot values for each plan and relate these values to the total number of detected flaws. The result is in both cases an average of the direct and indirect HotSpot values for flaws and gives a good estimation of the degree of connectivity between defects in plans.

The HotZone concept can also be transfered to plan-selection task, namely in two ways: the simpler form is identifying the plan with the smallest maximal HotZone value. The $f^{planSel}_{LeastHZone}$ module performs for each plan the calculations of f^{modSel}_{HZone} for each flaw, keeps the maximum HotZone value for each plan, and then prefers the plans accordingly. Similarly to the other HotSpot heuristics, this strategy has also the option to prefer strong or weak interactions in the plan.

The second HotZone derivate is $f^{planSel}_{FewerHZones}$, a module that prefers plans with fewer HotZone *clusters*. Such clusters are identified by performing the f^{modSel}_{HZone} computation: When following the transitive overlappings of components, the set of flaws can be partitioned accordingly into independent sub-sets. The number of identified sub-sets is the number of HotZones.

Following the modification-based modification selection, the examination of plan modifications can contribute to the plan selection as well: Corresponding to the direct uniform HotSpot plan-selection, $f^{planSel}_{FewerModHotSpotRatio}$ is a strategy module that accumulates the computed modification overlappings as provided by $f^{modSel}_{ModBasedHS}$ and relates these values to the total number of plan modifications issued for a particular plan. The average overlapping of plan modifications is intended to estimate the amount of interdependencies between the available flaw resolution proposals on a particular plan, and these interdependencies in turn indicate potentially conflicting refinement options in the future.

4 Evaluation

Most of the selection modules are not suitable for being used as singular strategies, because they cover only particular aspects of information about the search space. Consequently, we combine strategies into sequences of selection modules, as described above. This technique covers a wide area of search control in planning: e.g., performing modification selection, modulated by the available alternative modifications, as it is done in the *fewest alternatives first* heuristic for task-expansion selection of the UMCP system [4], is achieved by employing $f^{modSel}_{PrefMC}(\mathtt{M_{ExpTask}})$ as the primary modification selection, and f^{modSel}_{LCF} as the secondary strategy.

The well-known "expand then make sound" (EMS) strategy can be emulated by modification selections (1) f^{modSel}_{AddrFC} with argument $\mathtt{F}_{\mathtt{CausalThreat}}$, (2) f^{modSel}_{AddrFC} with argument $\mathtt{F}_{\mathtt{OpenPrec}}$, and (3) f^{modSel}_{PrefMC} with argument $\mathtt{M}_{\mathtt{ExpTask}}$: the strategy defers task expansion until the current plan is sound. The reverse combination correlates with the UMCP-style of HTN planning, which expands until the very concrete level and then fixes the causal structure.

Although our framework provides the means, a systematic experimental evaluation of all strategy combinations is not feasible. About 40 modification and 60 plan-selection modules – combined to triples – are far too many configurations to test; even though this includes many trivially fruitless combinations, such as fixed strategies that avoid, respectively prefer the same flaw classes. It also became apparent that all HotSpot variants are only performing well in the "avoidance" mode, and the same holds for plan selections that prefer less constrained or developed plans. Seeking HotSpots implies an early commitment and going after un-constrained plans tends to degrade into breadth-first search.

Our experiment domains were the UMTranslog domain from [6], a hierarchical variant of the Satellite competition domain, and an artificial domain "Criss-Cross". Its name is derived from the fact that the causal structure includes many interleaving causal dependencies, and plans in that domain have typically many HotSpots. The solution space for CC-problems is relatively sparse, while the larger problems in the Satellite and UMTranslog domains can be solved in multiple ways. UMTranslog and CC share a deep task expansion hierarchy and many (typing-) constraints in their methods. This makes both domains behave exactly the same way during our experiments: every strategy for UMTranslog is performing well in CC and vice versa. Satellite problems are a little bit less constrained and hierarchically structured. This may be the reason why they need different strategies and may also explain, why larger problems, which offer a larger selection of modification options, are sometimes easier to solve for the HotSpot strategies than smaller ones.

For this presentation, we conducted a series of experiments with only binary combinations of modification selection and a plan selection with the modules $f^{planSel}_{LessFlawsPerTask}$ and $f^{planSel}_{FewerModsFirst}$, which is a good compromise between a prediction of the future efforts and the branching factor for the newly explored node. Table 1 shows the results in terms of visited plans for finding a solution. The table contains no pure fixed strategy components, because their performance was between one and two orders of magnitude worse and also afflicted with a high variance. The first row displays our reference strategy from [6].

The Satellite domain turned out to be particularly vulnerable to variance problems and this is reflected in many runs of the smaller problems performing worse than the larger ones. Also because of high sample variances, we used the established LCF as a primary selection and used other flexible strategies for modulation. Two interesting observations could be made: using LCF for modulation did not work very well in most cases because the primary HotSpot strategies favor independent (in terms of plan structures) flaw situations, and those have in turn more modifications available on average due to a higher de-

gree of freedom. I.e., a secondary *LCF* strategy mostly contradicts the primary HotSpot and cannot stabilize its decisions. The second interesting outcome is the *HZone/LCF* combination. It is a good example for the dependency of domain characteristics and strategy quality: The HotZone is better informed than *LCF* in a domain with many interdependencies, while *LCF* performs better if that information is scarce. Furthermore, it was the only stable strategy in the experiments for the larger Satellite problems.

f^{modSel} primary – secondary	Sat1	Sat2	Sat3	Sat4	CC1	CC2	CC3	CC4	CC5
LCF – PrefMC($\mathtt{M}_{\mathtt{ExpTask}}$)	62	50	957	886	21	33	73	95	190
LCF – DirUniHS	43	48	771	854	21	34	75	92	185
LCF – DirAdaptHS	31	42	596	418	21	31	71	95	184
LCF – IndUniHS	24	37	475	367	21	30	69	95	186
LCF – IndAdaptHS	33	40	468	482	22	29	71	87	209
LCF – HZone	24	40	424	375	21	30	71	98	188
HZone – LCF	36	68	399	718	22	30	70	99	161
LCF – ModBasedHS	36	128	945	702	22	33	70	93	204

Table 1. Modification-selections for the Satellite and CrissCross domains.

Table 2 documents the performance of different single plan selections for a given least-commitment modification selection. It has to be noted that only the first strategy (our reference plan selection) and the "fewer HotZones" produce relatively stable results. High sample variances with encouraging low minima indicate that further experimentation with modulating secondary plan selections have to be conducted.

$f^{planSel}$	CC1	CC2	CC3	CC4	CC5
LessFlawsPerTask	21	33	73	95	190
DirUniHS	22	42	119	159	220
IndUniHS	22	38	125	152	210
LeastHZone	20	35	124	145	215
FewerHZones	21	34	72	94	205
FewerModBHS	22	35	189	365	420

Table 2. Plan-selections for the crisscross domain with $f^{modSel}_{LCF} - f^{modSel}_{PrefMC}(\mathtt{M}_{\mathtt{ExpTask}})$.

Our empirical analysis confirms previous results that flexible strategies are suitable domain-independent search procedures. It also raised a number of interesting questions, e.g., why every combination including the FewerHZones plan selection performs very well for CrissCross problems but only poor for Satellite problems, while LeastHZone produces the opposite behaviour. The differences between those two heuristics are very subtle and it has to be clarified, which property of a domain or problem they specifically address.

5 Conclusions and Future Developments

We presented a formal framework for refinement-based planning in which the functionality for generation plans is decomposed into plan-deficiency detection, plan-modification computation, and (tactical) search reasoning. In particular, we focused on search strategies: how to systematically construct strategic guidance and how to evaluate its performance. Strategies from the literature have been integrated into this framework and novel ones have been developed.

The newly developed search strategies are procedures that are capable of reasoning about the interaction of flaws and plan modifications *in general*, thereby becoming unlimited in their applicability and expandability: flexible strategies can be deployed in any system configuration, ranging from hybrid planning to resource planning, etc.

However, more experimental evidence has to be collected in order to get more insight into the relation between modification and plan selections and their performance in specific domains. Future work also includes the investigation of dynamic strategies, and a more thorough coverage of the available strategy spectrum. In particular, we will deal with strategy performance in an integrated planning-and-scheduling system.

Acknowledgement The authors would like to thank Andreas Lanz who contributed to the strategy implementation and the experimental evaluation.

References

1. Joslin, D., Pollack, M.: Least-cost flaw repair: A plan refinement strategy for partial-order planning. In Hayes-Roth, B., Korf, R., eds.: Proc. of the 12th National Conference on AI, AAAI (1994) 1004–1009
2. Peot, M.A., Smith, D.: Threat removal strategies for partial-order planning. In: Proc. of the 11th National Conference on AI. (1993) 492–499
3. Schubert, L.K., Gerevini, A.: Accelerating partial order planners by improving plan and goal choices. In: Proc. of the 7th IEEE International Conference on Tools with AI, IEEE Computer Society Press (1995) 442–450
4. Tsuneto, R., Nau, D., Hendler, J.: Plan-refinement strategies and search-space size. In Steel, S., Alami, R., eds.: Proc. of the 4th European Conference on Planning. Volume 1348 of LNCS., Springer (1997) 414–426
5. McCluskey, T.L.: Object transition sequences: A new form of abstraction for HTN planners. In Chien, S., Kambhampathi, R., Knoblock, C., eds.: Proc. of the 5th International Conference on AI Planning Systems, AAAI (2000) 216–225
6. Schattenberg, B., Weigl, A., Biundo, S.: Hybrid planning using flexible strategies. In Furbach, U., ed.: Proc. of the 28th German Conference on AI. Volume 3698 of LNAI., Springer (2005) 258–272
7. Schattenberg, B., Biundo, S.: A unifying framework for hybrid planning and scheduling. In Freksa, C., Kohlhase, M., Schill, eds.: Proc. of the 29th German Conference on AI. Volume 4314 of LNAI., Springer (2007) 361–373
8. 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.: Proc. of the 6th European Conference on Planning. (2001)