Hybrid Planning Using Flexible Strategies

Bernd Schattenberg, Andreas Weigl, and Susanne Biundo

Dept. of Artificial Intelligence
University of Ulm

Motivation

- Crisis Management Support: THW missions at River Oder (1997) and Danube (2002/2005)
- Size and complexity demand for system support, e.g. for training
- Application properies:
  - Procedural knowledge available
  - ... however fragmentary

Introduction

Utilizing procedural knowledge

real-world domains can/should be
ybrid planning, a combination of

- Hierarchical Task Network (HTN) Planning
  - Pre-defined plans implement abstract actions (tasks)
- Partial Order Causal Link (POCL) Planning
  - Synthesizing partially ordered plans by causal reasoning

Handling underspecified procedures

Aims

- Proper theoretical integration of POCL and HTN techniques
- Deriving the design of a hybrid planning system from the formal frame
- Providing an experimental platform for evaluation of planning functionality and search procedures
- Development of efficient strategies for hybrid planning systems
- Systematic experimental evaluation of search strategies

States & Tasks

- States as finite sets of ground atoms
  - Language over rigid and flexible symbols
  - Sort axioms reflect hierarchies on sorts
- Uniform representation of primitive and complex actions: task schemata $t(\overline{\tau}) = (\mathrm{prec}(t(\overline{\tau})), \mathrm{add}(t(\overline{\tau})), \mathrm{del}(t(\overline{\tau})))$
  - Preconditions as sets of literals, positive and negative effects as sets of atoms
- Instances of primitive schemata: operations
  - Applicable in a state s, if $\mathrm{prec}^{\oplus}(o(\bar{c})) \subseteq s \;\wedge\; \mathrm{prec}^{\ominus}(o(\bar{c})) \cap s = \emptyset$
  - State transformation $s' = (s \cup \mathrm{add}(o(\bar{c}))) \setminus \mathrm{del}(o(\bar{c}))$

Plans & Task Networks

- Plans / task networks are tuples $(TE, \prec, VC, CL)$
  - TE: set of plan steps (task expressions) $te = l : t(\overline{\tau})$
  - <: partial ordering on TE
  - VC: set of variable constraints $v \doteq \tau \quad v \dot{\neq} \tau$
  - CL: set of causal links $te_i \xrightarrow{\phi} te_j$
- Decomposition methods relate abstract tasks with their implementing networks $m = (t(\bar{\tau}), d)$

Problems & Solutions

- Planning problem (d, T, M)
- Solution criteria for plan $P = (TE, \prec, VC, CL)$
    - Plan steps in TE are primitive
    - < is acyclic
    - VC is consistent
    - Plan steps in TE are fully instantiated
    - CL is compatible with <
    - All preconditions are supported
    - No causal support is threatened

Plan Modifications

- Explicit representation of changes to the plan
  - $(mod, E^{\oplus} \cup E_{\ominus})$
  - Elementary additions and deletions of plan components
- Plan modifications for hybrid planning
  - Adding causal links, inserting new plan steps with causal links
  - Adding ordering / variable constraints
  - Expanding abstract task expressions according to method definitions

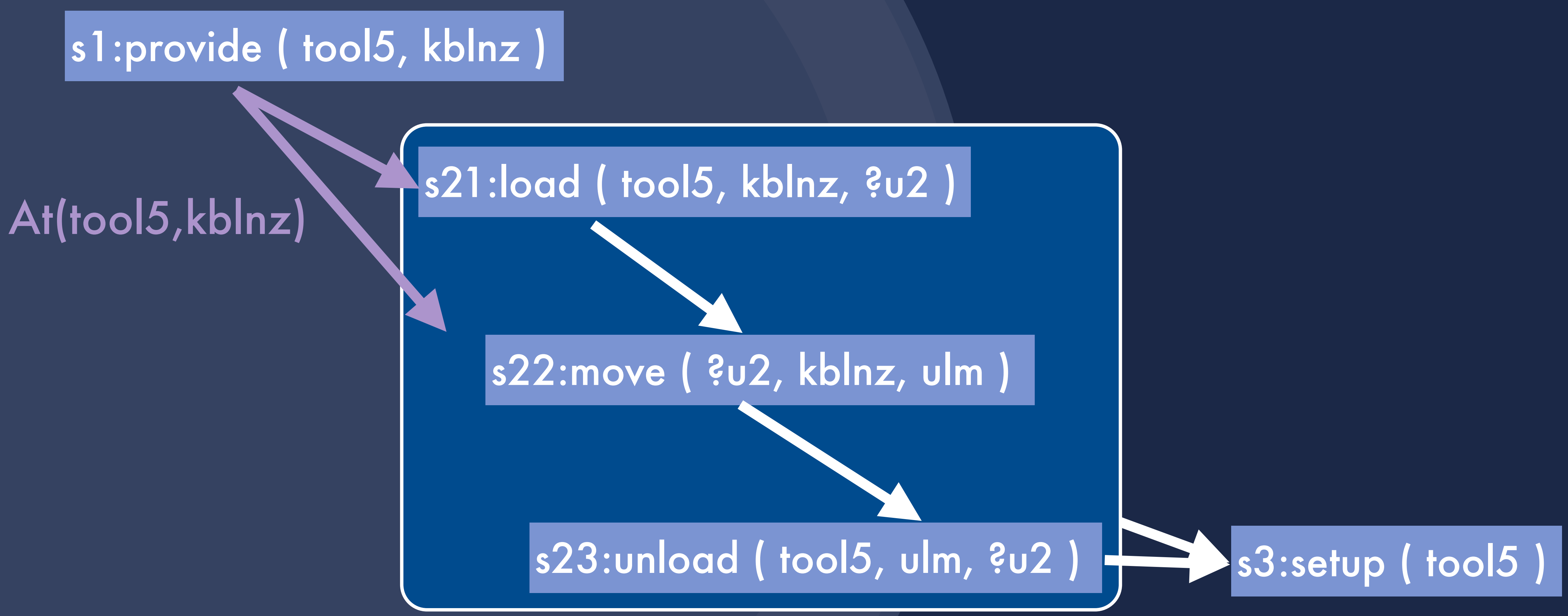
Example: Expansion

- Given abstract task expression $te = l : t(\overline{\tau})$ in plan $P = (TE, \prec, VC, CL)$ with method $m = (t, (TE_m, \prec_m, VE_m, CL_m))$

- $$\begin{aligned}(\texttt{ExpandTask}, &\{\ominus te\} \cup \{\oplus te_m | te_m \in TE_m\} \cup \\ &\{\oplus(te_{m1} \prec te_{m2}) | (te_{m1} \prec_m te_{m2})\} \cup \\ &\{\oplus(te_{m1} \xrightarrow{\phi} te_{m2}) | (te_{m1} \xrightarrow{\phi} te_{m2}) \in CL_m\} \\ &\{\oplus(v = \tau) | (v = \tau) \in VC_m\} \cup \{\oplus(v \neq \tau) | (v \neq \tau) \in VC_m\} \cup \\ &\{\ominus(te' \prec te), \oplus(te' \prec te_m) | (te' \prec te), te_m \in TE_m\} \cup \\ &\{\ominus(te \prec te'), \oplus(te_m \prec te') | (te \prec te'), te_m \in TE_m\} \cup \\ &\{\ominus(te \xrightarrow{\phi} te'), \oplus(te_m \xrightarrow{\phi} te') | (te \xrightarrow{\phi} te') \in CL, \\ &\qquad te_m \in TE_m, |\phi| \in add(te_m) \cup del(te_m)\} \cup \\ &\{\ominus(te' \xrightarrow{\phi} te), \oplus(te' \xrightarrow{\phi} te_m) | (te' \xrightarrow{\phi} te) \in CL, \\ &\qquad te_m \in TE_m, \phi \in prec(te_m)\}\end{aligned}$$

Example: Expansion

- Given abstract task expression $te = l : t(\overline{\tau})$ in plan $P = (TE, \prec, VC, CL)$ with method $m = (t, (TE_m, \prec_m, VE_m, CL_m))$



Flaws

- Explicit representation of solution criteria violations
  - $(flaw, E)$
  - References to involved plan components
- Flaws for hybrid planning
  - Presence of abstract tasks
  - Orderings cyclic or incompatible with causal links
  - Variable constraints inconsistent
  - Open variable bindings
  - Open preconditions
  - Causal Threats

Example: Causal Threat

- In the following situation...
  - Causal link $te_i \xrightarrow{\phi} te_j$
  - A plan step te_k for which
    if literal is positive $\sigma_{VC}(\phi) \in \sigma_{VC}(\mathrm{del}(te_k))$
    if literal is negative $\sigma_{VC}(|\phi|) \in \sigma_{VC}(\mathrm{add}(te_k))$
  - Neither $te_k \not\prec^* te_i$ nor $te_j \not\prec^* te_k$
- ... the following flaw is issued
  $(\mathtt{Threat}, \{te_i \xrightarrow{\phi} te_j, te_k\})$

Triggering Modifications

- Certain classes of modifications can *in principle* solve certain classes of flaws
  - Example: Causal threat solvable by separation, promotion/demotion, or expansion (overlapping)
- Make this relationship explicit by defining a function for identifying suitable modification classes $\alpha : 2^{\mathcal{F}} \rightarrow 2^{\mathcal{M}}$
- $$\alpha(\mathcal{F}_x) = \begin{cases} \mathcal{M}_{\texttt{ExpandTask}} & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{AbstrTask}} \\ \mathcal{M}_{\texttt{AddCLink}} \cup \mathcal{M}_{\texttt{InsertTask}} \cup \mathcal{M}_{\texttt{ExpandTask}} & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{OpenPrec}} \\ \mathcal{M}_{\texttt{ExpandTask}} \cup \mathcal{M}_{\texttt{AddOrdConstr}} \cup \mathcal{M}_{\texttt{AddVarConstr}} & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{Threat}} \\ \mathcal{M}_{\texttt{AddVarConstr}} & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{OpenVarBind}} \\ \emptyset & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{OrdIncons}} \\ \emptyset & \text{if } \mathcal{F}_x = \mathcal{F}_{\texttt{VarIncons}} \end{cases}$$

A Modular Design

- Flaw and modification generating modules
  - $f_x^{det} : \mathcal{P} \rightarrow 2^{\mathcal{F}_x}$
  - $f_y^{mod} : \mathcal{P} \times 2^{\mathcal{F}_x} \rightarrow 2^{\mathcal{M}_y}$ for $\mathcal{M}_y \subseteq \alpha(\mathcal{F}_x)$
- Strategy modules
  - $f_z^{strat} : \mathcal{P} \times 2^{\mathcal{F}} \times 2^{\mathcal{M}} \rightarrow \mathcal{M} \cup \epsilon$
  - Plans with un-addressed flaws can be discarded, i.e.

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

A Generic Algorithm

plan(P, T, 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(apply(P, f_z^{strat}(P, F, M)), T, M)

Detection Module D_1

$(label, E)$

α

Modification Module M_1

$(label, E_{\oplus} \cup E_{\ominus})$

Strategy

$P = (TE, \prec, VC, CL)$

Fixed Strategies

- UMCP [Erol et al., 1995]: Pure HTN planner, prefers decomposition, causal conflicts etc. on primitive level

$$f^{strat}_{UMCP}(P, F, M) = m \in \begin{cases} select(M \cap \mathcal{M}_{\texttt{ExpandTask}}) \text{ if } F \cap \mathcal{F}_{\texttt{AbstrTask}} \neq \emptyset \\ M \text{ otherwise} \end{cases}$$

- EMS [McCluskey, 2000]: “Expand then make sound”

$$f^{strat}_{EMS}(P, F, M) = m \in \begin{cases} select(M \cap \mathcal{M}_{\texttt{ExpandTask}}) \text{ if } F \cap (\mathcal{F} \setminus \mathcal{F}_{\texttt{AbstrTask}}) = \emptyset \\ M \text{ otherwise} \end{cases}$$

- Shop [Nau et al., 1999]: Expansion in execution order

$$f^{strat}_{SHOP}(P, F, M) = m \in \begin{cases} select(M \cap \mathcal{M}_{\texttt{ExpandTask}}) \\ \quad \text{if } F \cap (\mathcal{F} \setminus \mathcal{F}_{\texttt{AbstrTask}}) = \emptyset \\ \quad \text{for } t_f \text{ in } \texttt{f}, \texttt{f} \in F, t_f \in T_{before} \text{ and} \\ \quad\quad t_m \text{ in } \texttt{m}, \texttt{m} \in M, t_m \in T_{left} \\ M \text{ otherwise} \end{cases}$$

Flexible Strategies

- Least Committing First: Modification from smallest answer set

$$f(f,\{m_1,\ldots,m_n\}) = \begin{cases} 0 & \text{for } n = 0 \\ 1 + f(f,\{m_1,\ldots,m_{n-1}\}) & \text{for } m_n \text{ answering } f \\ f(f,\{m_1,\ldots,m_{n-1}\}) & \text{otherwise} \end{cases}$$

$$f^{strat}_{LCF}(P, F, M) = m \in \{m_f \mid f \in \min(f(f, M))\}$$

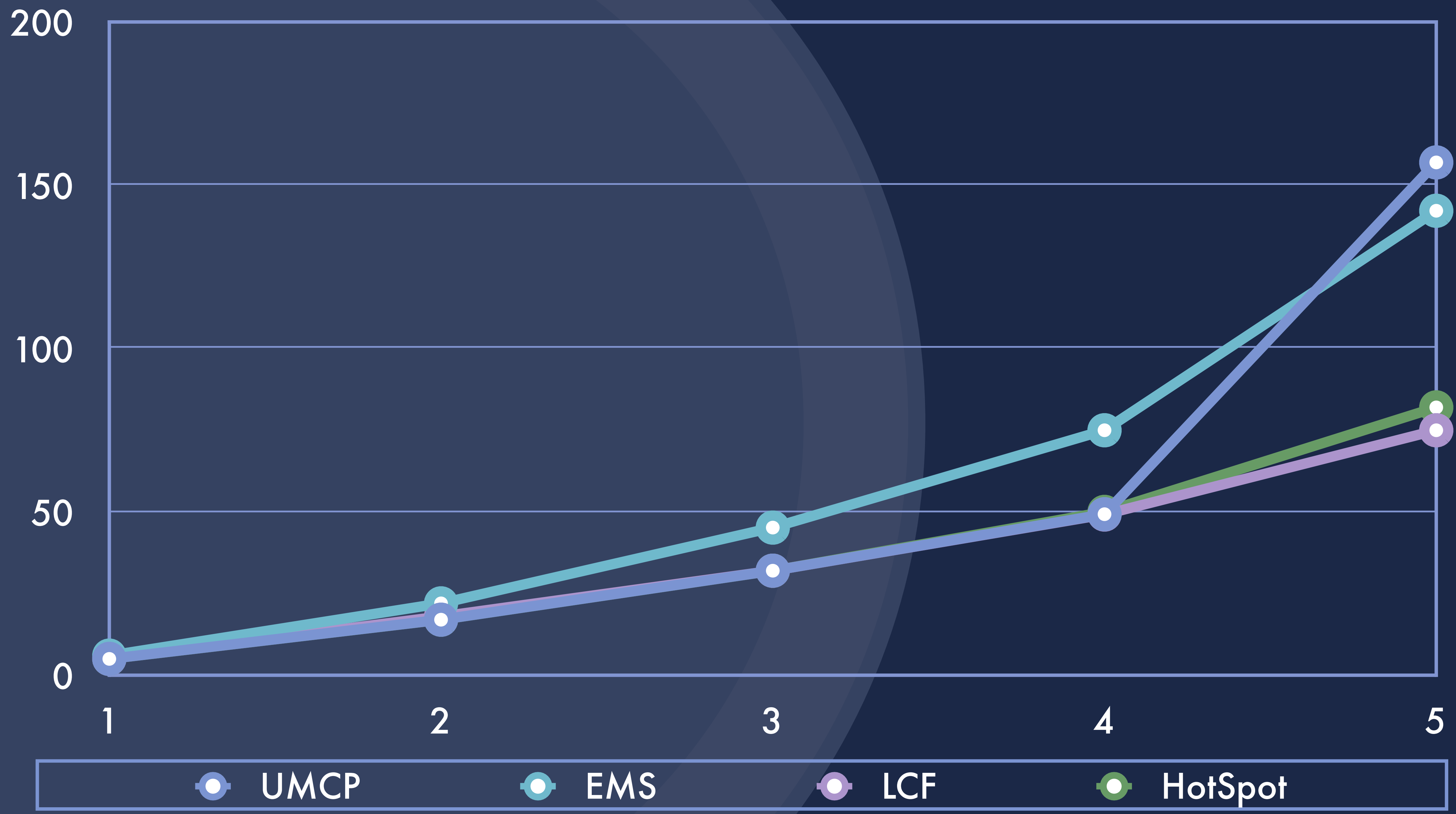
- HotSpot: Avoiding flaws with structural commonalities → independent modifications

$$h(f,\{f_1,\ldots,f_m\}) = \begin{cases} 0 & \text{if } m = 0 \quad \text{if } m = 0 \\ h(f,\{f_1,\ldots,f_{m-1}\}) & \text{if } f = f_m \quad \text{if } f = f_m \\ h(f,\{f_1,\ldots,f_{m-1}\}) + |\{e_1,\ldots,e_n\} \cap \{e_{m_1},\ldots,e_{m_v}\}| & \text{otherwise} \quad \text{otherwise} \end{cases}$$

$$f^{strat}_{HotSpot}(P, F, M) = select_g(\{m_f \mid f \in \min(h(f, F))\})$$

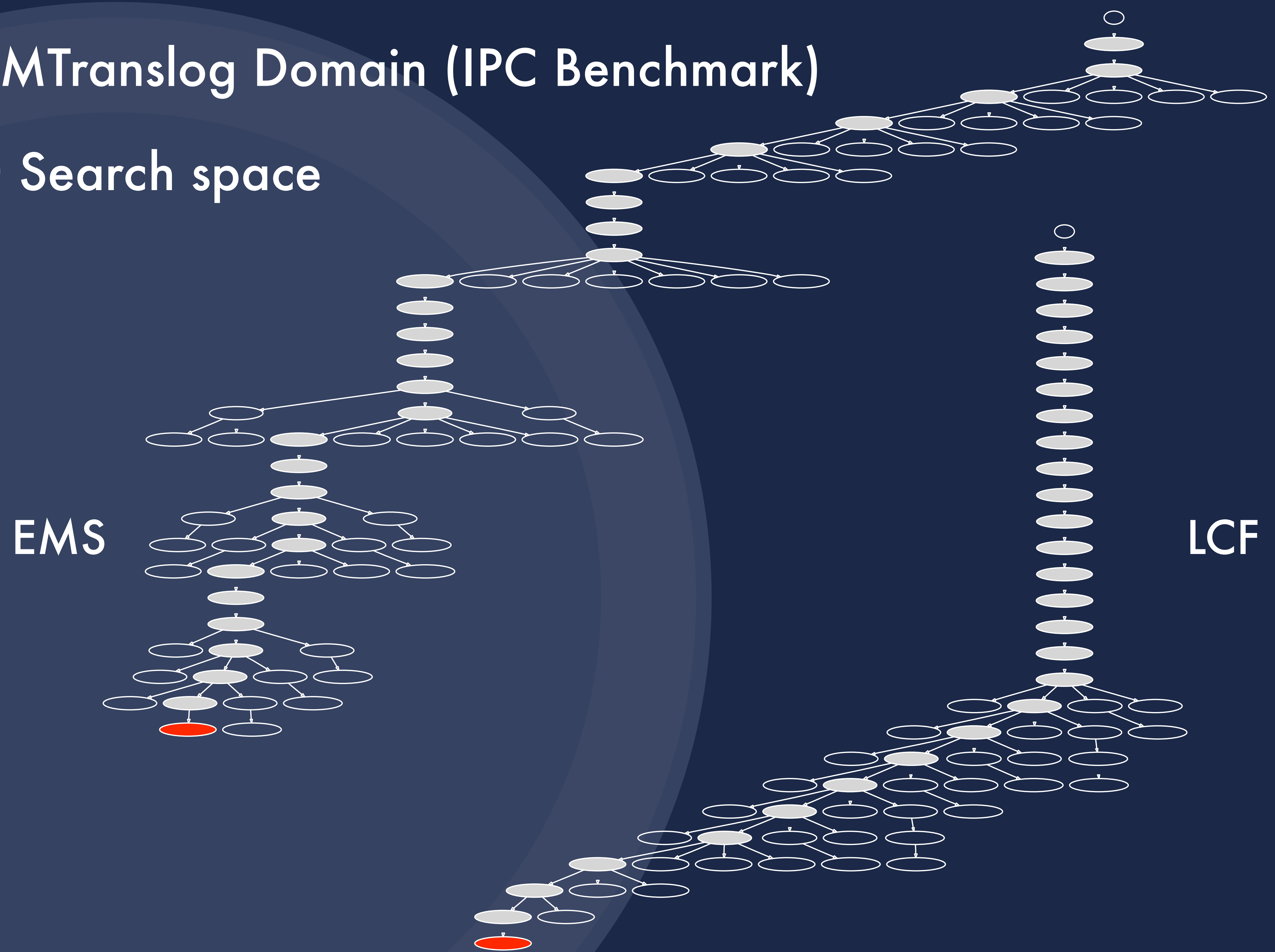
Experimental Results

- UMTranslog Domain (IPC Benchmark)
  - Size of search space



Experimental Results

- UMTranslog Domain (IPC Benchmark)
  - Search space



Conclusion

- Formal frame for the integration of HTN and POCL techniques
- Flexible architecture which decouples flaw detection, modification computation, and search control
  - Platform for implementation and evaluation of planning systems
  - Systematic approach allows for novel *flexible* strategies
- Systematic evaluation of search strategies
  - ... *to be continued*