Plan Failure Analysis and Plan Adaptation for Multi-Level Campaign Planning

Jens Happe
MDA Systems Ltd.
Richmond, B.C.

Mohamad Allouche
Micheline Bélanger
DRDC-Valcartier
Outline

• **The Military Planning Process**
• **Plan Representation Model**
  • Classical Planning
  • Hierarchical Task Networks
  • Scheduling Plans
  • Hierarchical Goal Analysis
• **Prototype System**
• **Conclusion**
Military Plan Management Needs

• Distributed
  • Different commanders have different responsibilities
  • Interference between plans and between levels of command

• Dynamic
  • Highly variable and uncertain situation
  • Plans must be adapted on the fly

• Intuitive
  • Users are not logicians or knowledge experts
  • Map/visually oriented
  • No information overload!
Plan Management Aspects

• Plan Representation
  • gives a complete, unambiguous description of a plan at the appropriate level(s) of abstraction
  • specifies all aspects of a plan, including resources, assumptions, constraints, and objectives
  • includes visualization

• Plan Analysis
  • assesses the feasibility and quality of a plan
  • describes links (dependencies) between plan elements
  • identifies aspects with possible positive/negative impact
  • identifies and tracks possible causes of failure
Plan Management Aspects

- **Plan Forecasting**
  - projects the current state of plan execution forward
  - makes expectations of future plan outcomes
  - predicts the impact of plan updates

- **Plan Monitoring**
  - observes the plan execution in real time
  - identifies and tracks changes in the environment
  - compares the current and predicted plan state
  - determines changes required to an active plan
Outline

• The Military Planning Process
• Plan Representation Model
  • Classical Planning
  • Hierarchical Task Networks
  • Scheduling Plans
  • Hierarchical Goal Analysis
• Prototype System
• Conclusion
Plan Representation Model

• Given: A human-readable representation of a military campaign plan.
• Goal: Find a computational representation of this plan, which:
  • captures all goals, assumptions, constraints
  • can be easily validated
  • allows identifying the source of failures
  • allows plan monitoring and forecasting
  • can be visually presented at different levels of abstraction
  • can be collaboratively edited.
Classical AI Planning(1)

• Given:
  • Original state $S_0$
  • State transition actions
  • Set of goal states $G$

• Planning problem
  • Find a plan (=sequence of actions) that transforms $S_0$ into a goal state in $G$. 

![Diagram showing state transitions from $S_0$ to $S_1$, $S_2$, and so on, with a question mark indicating the goal state in $G$.]
Classical AI Planning(2)

• Suitable for a homogeneous search space
  (no hierarchical plan structure)

• Disadvantages:
  • High computational complexity
  • Poor guidance towards a solution
  • Not intuitive:
    • Nobody plans a campaign as a sequence of atomic steps.
Hierarchical Task Networks(1)

- More intuitive:
  - Plans decompose into subplans, sub-subplans etc.
  - Actions are the elementary, executable steps (leaves of the decomposition tree)
  - Note: the decomposition tree is NOT the HTN.
Hierarchical Task Networks (1)

• More intuitive:
  • Plans decompose into subplans, sub-subplans etc.
  • Actions are the elementary, executable steps (leaves of the decomposition tree)
  • Note: the decomposition tree is NOT the HTN.

• Tasks
  • Templates for plans, subplans or actions
  • Expressions with free variables (parameters)
  • Lowest-level tasks are called *primitive*. 
Hierarchical Task Networks(2)

- Decomposition Methods:
  - Head task (non-primitive)
  - Set of Subtasks (primitive or non-primitive)
  - Constraint relations between tasks.
Hierarchical Task Networks (2)

- Decomposition Methods:
  - Head task (non-primitive)
  - Set of Subtasks (primitive or non-primitive)
  - Constraint relations between tasks.

- Operators:
  - Primitive task
  - Preconditions
  - Effects
Hierarchical Task Networks (3)

- Example:
  - **Operator** `fly(res, from, to)`:  
    - Task: `move(res, from, to)`
    - Precondition: `is_at(res, from)`
    - Effect: `is_at(res, to)`

  - **Operator** `pick_up(res, person, at)`:  
    - Task: `attach(res, person)`
    - Precondition: `is_at(res, at)`
    - Precondition: `is_at(person, at)`
    - Effect: `is_attached(res, person)`
Hierarchical Task Networks (4)

- **Method** `airlift(res, personnel, start, loc, end)`:
  - **Task**: `rescue(personnel, loc)`
  - **Subtasks**:
    - 1. `move(res, start, loc)`
    - 2. `attach(res, personnel)`
    - 3. `move(res, loc, end)`
  - **Precedences**:
    - 1. before 2.
    - 2. before 3.

```
move(res, start, loc)
attach(res, personnel)
moves(res, loc, end)
```
Hierarchical Task Networks (5)

- Given the task instance
  \[ \text{rescue}(\text{stranded\_crew, hornby\_island}) \]

- A valid plan (decomposition)
  (executable if resource \text{heli17} is currently at \text{YVR}):
Hierarchical Task Networks (6)

• Advantages:
  • Abstracts from plan instances to plan templates
  • Encodes domain knowledge within the methods
  • Allows formal verification of plans
  • Considerably reduces the state space
Hierarchical Task Networks (6)

- Advantages:
  - Abstracts from plan instances to plan templates
  - Encodes domain knowledge within the methods
  - Allows formal verification of plans
  - Considerably reduces the state space

- Disadvantages:
  - HTN need to be designed by a domain expert up front
  - Represents precedences between tasks, but not time
  - No notion of goals
Scheduling HTN plans

• HTN plans already define a partial execution order
• Scheduling the plans is easy.
  • Assume: elementary actions have known durations
  • Manually: Can verify whether the schedule is feasible
  • Automatically: schedule all actions as early as possible
    • Users can define a later start time, insert pauses etc.
  • The plans' execution times are aggregated from those of the actions.
Scheduling HTN plans: Example

• Start time: 11h15
• Actions' execution times:

<table>
<thead>
<tr>
<th>Action</th>
<th>Execution Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>fly(heli17, YVR, hornby_island)</td>
<td>40 min</td>
</tr>
<tr>
<td>pick_up(heli17, stranded_crew, hornby_island)</td>
<td>20 min</td>
</tr>
<tr>
<td>fly(heli17, hornby_island, VGH)</td>
<td>45 min</td>
</tr>
</tbody>
</table>

airlift(heli, stranded_crew, YVR, hornby_island, VGH))
Hierarchical Goal Analysis

- Often used in military planning
- Similar to plan decomposition:
  - A given goal is decomposed into subgoals.
  - Units are tasked with lowest-level subgoals, executing a plan to accomplish their goal
- Forces commanders to think differently:
  - *Not*: what does the plan tell us to do?
  - *But*: what need we do to achieve success?
Combining Plan and Goal Hierarchies

- **Goals**
- **Plans**
- **Goal Decomposition**
- **Plan Decomposition**
- **Plan Assignment**

Diagram: Hierarchical structure of plans and goals, with arrows indicating decomposition and assignment relationships.
Outline

• The Military Planning Process
• Plan Representation Model
  • Classical Planning
  • Hierarchical Task Networks
  • Scheduling Plans
  • Hierarchical Goal Analysis
• Prototype System
• Conclusion
Prototype System

- Distributed multi-user system
  - Filtered by user's area of responsibility
  - Tailored to user's level of abstraction
  - Drill-down access to underlying elements
- Always shows:
  - Goal and Plan Decomposition Hierarchy
  - Time slider
  - Predicted execution status of plans and goals (colour)
- Client-Server architecture
Prototype System: Views

• The following screenshots show a representative extract from the North Atlantis Combat Search and Rescue vignette:

<table>
<thead>
<tr>
<th>Plan</th>
<th>Goal</th>
<th>Subplans</th>
<th>Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>CSAR extraction</td>
<td>f-, f, f+</td>
<td>CH-53 (rescue) 2xCF-18 (echo 1/2) stranded heli crew</td>
</tr>
<tr>
<td>P2</td>
<td>Air superiority</td>
<td>c, c+</td>
<td>1xCF-18</td>
</tr>
<tr>
<td></td>
<td>Resource recovery</td>
<td>(f+,b+,c+)</td>
<td>all</td>
</tr>
</tbody>
</table>
Prototype System: Map View
Prototype System: Schedule View
Prototype System: Browser View

![Prototype System Image]

- **Plan:** \( \text{F}^* \)
- **Start:** 11:00:00
- **End:** 11:15:00
- **Duration:** 00:15:00
- **Assigned to:** None
- **Implements:** escorted_fly(Crash Site, Wahiab, rescue heli, echo 1)
- **Method:** escorted_fly_decomposition(Crash Site, Wahiab, rescue heli, echo 1)
- **Parent Plan(s):** Rescue Crew
- **Sub-plans:**
  - \( \text{f}^1\) [fly]
  - \( \text{f}^2\) [fly]
- **Attains Goal(s):** Platform Recovery
- **Status:** NOT_STARTED
- **Predicted at end:** SUCCESS
- **Resources used:**
  - echo 1
  - rescue heli
- **ROI:**
  - 56:50:00.0001N to 58:00:00.001N
  - 27:40:00.006W to 23:50:00.001W
- **Preconditions:** None
- **Precedences:**
  - \( \text{f}^1\) starts at least 03:00:00 after and at most 20:00:00 after \( \text{f}^2\) starts
- **Effects:** isycled(escort)
Prototype System: Adaptation Capability
Outline

• The Military Planning Process
• Plan Representation Model
  • Classical Planning
  • Hierarchical Task Networks
  • Scheduling Plans
  • Hierarchical Goal Analysis
• Prototype System
• Conclusion
Conclusion

• Prototype of a plan management system that address the needs of military planners
  • distributed multi-level multi-plan management
  • insight into the plan status, without information overload
  • ability to manage dependencies between plans
  • ability to trace conflicts and plan failures to their source

• Solid underlying representation model
  • Combines hierarchical task networks, scheduling, and hierarchical goal analysis
  • Encodes domain knowledge
  • Allows formal validation of plans
Future Work

• Develop an operational system, or integrate the concepts into an existing system
  • Support other plan elements such as centres of gravity, decisive points, risk (e.g. CF OPP)
• More visual concepts to support planning
• Heuristics that propose suitable plans for a given situation (e.g. case-based)