HTN plánování

Radek Mařík

CVUT FEL, K13133

18. dubna 2011

Obsah

Obsah

Hierarchické plánování [SV10]

angl. Hierarchical Task Net (HTN) Planning

Základní myšlenka

- složité plány mají často zřejmou strukturu,
- struktura se dá zachytit ve formě hierarchie abstraktních plánů.
- podplány jsou často (skoro) na sobě nezávislé.)

Příklad

- "Abychom se dostali na konferenci v ?x, dojeď na letiště, nastup do letadla do ?x, potom jeď to hotelu konference."
- "Abychom se dostali na letiště, buď řiď auto nebo si vezmi taxi."
- "Jestliže máš dostatek peněz a potřebuješ si vzít taxi do ?y, buď zavolej dopředu nebo zamávej rukou, potom nasedni do taxíku, řekni "Chci je to ?y", počkej dokud nedojedete do ?y, zaplať jízdné a poté vystup z taxíku."

Hierarchické plánování [SV10]

angl. Hierarchical Task Net (HTN) Planning

Základní myšlenka

- složité plány mají často zřejmou strukturu,
- struktura se dá zachytit ve formě hierarchie abstraktních plánů.
- podplány jsou často (skoro) na sobě nezávislé.)

Příklad

- "Abychom se dostali na konferenci v ?x, dojeď na letiště, nastup do letadla do ?x, potom jeď to hotelu konference."
- "Abychom se dostali na letiště, buď řiď auto nebo si vezmi taxi."
- "Jestliže máš dostatek peněz a potřebuješ si vzít taxi do ?y, buď zavolej dopředu nebo zamávej rukou, potom nasedni do taxíku, řekni "Chci je to ?y", počkej dokud nedojedete do ?y, zaplať jízdné a poté vystup z taxíku."

HTN plány [SV10]

- Plan = (Tasks, Constraints)

Úlohy

- Úlohy
  - primitivní
    - standard forma STRIPS operátorů
    - "proveditelné"
  - složené
    - vstupní podmínky a efekty
    - metody na kompozici operátory do podplánů

HTN plány [SV10]

- Plan = (Tasks, Constraints)

Úlohy

- Úlohy
  - primitivní
    - standard forma STRIPS operátorů
    - "proveditelné"
  - složené
    - vstupní podmínky a efekty
    - metody na kompozici operátory do podplánů

HTN plány [SV10]

- ***Plan = (Tasks, Constraints)***
- ***Tasks***
  - ***Primitive***
    - Standard STRIPS-style operators
    - "Executable"
  - ***Compound***
    - Preconditions and Effects
    - ***Methods*** for decomposing operator into more detailed subplans
      - Details internal structure
      - Parameterized
      - Similar to *macros* or *subroutines*

HTN plány [SV10]

- ***Constraints***
  - Precedence (*before, after, between)*
  - Metric temporal
  - Resource
  - Constraints at one level apply to all pairs of tasks at lower level

$$\mathbf{U = \{u_1, u_2\};\ V = \{v_1, v_2\}}$$

$$\mathbf{U < V \Rightarrow u_1 < v_1;\ u_2 < v_1;\ u_1 < v_2;\ u_2 < v_2}$$

- An ***Abstract Plan*** contains compound operators
- An ***Instantiated Plan*** has only primitive operators
- A ***Fully Instantiated Plan*** is a totally-ordered instantiated plan with all variables bound

Vyhledávací prostor [SV10]

- Problem Reduction Search
  - "Goal state" is an abstract ***task*** to be achieved (***not*** state of the world)

- Search space operators
  - Decompose task into subtask(s)
  - Parameterize task
  - Solve for conflicts

Abstraktní HTN algoritmus [SV10]

- HTN-Plan (*tasks, constraints, methods*)
  - If (*tasks, constraints*) has no solution, return {}
  - If (*tasks, constraints*) is an instantiated plan
    - **Select** a fully instantiated plan
    - If such a plan exists, return it; otherwise return {}
  - **Select** an abstract task $t \in tasks$
  - **Choose** an applicable $m \in methods$
    - where u is the task-list of m and c is the constraints of m
    - $tasks = tasks - \{t\} \cup u$
    - $constraints = constraints \cup c$
  - (*tasks, constraints*) = **apply-critics**(*tasks, constraints*)
  - Return HTN-Plan(*tasks, constraints, methods*)

Proč je tak dobré [SV10]

- Methods encode domain knowledge
- Methods encode problem solving knowledge
  - "how" rather than "what"
- Abstractions encapsulate patterns of interaction

\+ May be easier to specify domain

\- Have to specify all possible goals (and how to achieve them)

Caveats

- HTN planning, in worst case, is still NP-complete
- May not terminate (recursive method expansions – may be hard to detect infinite loops)
- May have to completely expand before finding plan is illegal

Simple Hierarchical Order Planner (SHOP) [SV10]

- SHOP Algorithm:
  - Forward search, linear planner
    - Plans in same order as execution
    - Essentially depth-first search
  - Primitive operators have no preconditions
  - No concurrent actions
  - Highly expressive operator representation (numeric calculations)
  - Efficient (but rather inflexible) planning algorithm

Příklad SHOP domény [SV10]

(:operator (!putdown ?block)
((holding ?block)) ← Delete list
((ontable ?block) (handempty))) ← Add list

(:method (make-clear ?y)
((clear ?y)) ← Applicability condition
nil) ← Task list

(:method (make-clear ?y)
((on ?x ?y)) ← Applicability condition
((make-clear ?x)
(!unstack ?x ?y) (!putdown ?x)) ← Task list

Rozšíření jednoduchého HTN plánování [SV10]

1. Threat detection
   - Deal with interacting goals
   - Find ways of reusing operators
2. Methods can include preconditions and effects
   - Find threats earlier in the planning process
3. Methods can indicate resource usage
   - Do some types of scheduling
4. Operators/Methods can include open conditions ("external preconditions")
   - Need to use action-based (refinement) planning
   - Enables planner to find "novel" solutions

Příklad: stavba domu (O-Plan) [SV10]

Method Build (?house)

Precondition:	(and (own land) (have money))
Effects:	(built ?house)
Applicability:	(single-family-home ?house)
Expansion:	S1: Build-Foundation(?house)
	S2: Build-Frame(?house)
	S3: Build-Roof(?house)
	S4: Build-Walls(?house)
	S5: Build-Interior(?house)
	S6: Decorate(?house)
Orderings:	S1<S2, S2<S3, S2<S4, S3<S5, S5<S6
Links:	S1 causes (foundations laid) for S2
	S2 causes (frame erected) for S3 and S4
	S3 causes (roof built) for S5
	S4 causes (walls built) for S5
	S5 causes (interior done) for S6
TimeWindow:	start between 11:30 and 14:30 at S3
Resources:	bricklayers = between 1 and 2 men at S4

Interakce dílčích plánů [SV10]

- Types of Interactions
  - Deleted condition (threat)
  - Resource (e.g., “existing object” bindings)
  - Redundant steps

- HTN Planners Often Use ***Critics*** to Detect Certain Types of Illegal Plans and/or Synergies
  - Time window bounds
  - Resource bounds
  - Interaction of effects between compound tasks
  - Operators that can be ***merged***
    - In contrast, POP planners try to ***share*** operators
    - Merging is more efficient, but may not be complete

Literatura I

Reid Simmons and Manuela Veloso.
Planning, execution, and learning, lecture notes.
http://www.cs.cmu.edu/ mmv/planning/, September 2010.