Flexible FOND HTN Planning

Dillon Chen, Pascal Bercher

School of Computing
The Australian National University

ICAPS 2022

HTN planning

- *"planning or decision making with restrictions on actions"*
- a task network is a **partially ordered collection/directed acyclic graph** of tasks
- a HTN problem has the form $P = \langle F, N_p, N_c, \delta, M, s_I, \mathrm{tn}_I \rangle$
  - F is a set of facts, of which a subset is a state
  - N_p is a set of primitive task names
  - N_c is a set of compound task names
  - δ maps primitive task names to actions
  - M maps compound task names to task networks
  - $s_I \subseteq F$ is an initial state
  - tn_I is an initial task network
- a solution consists of a **sequence** of decomposition methods in M applied on tn_I followed by a **sequence** of executable tasks on the decomposed network

FOND$^{\text{MP}}$ HTN planning

- a FOND$^{\text{MP}}$ HTN problem has the form $P = \langle F, N_p, N_c, \delta, M, s_I, \text{tn}_I \rangle$
- δ now maps primitive task names to *nondeterministic* actions
- a solution consists of a **policy of task selection on task network-state tuples** $\sigma_\alpha = (\text{tn}_\alpha, s_\alpha)$ which either
  1. executes a first primitive task $t \in \text{tn}_\alpha$ applicable to s_α, *or*
  2. decomposes a first compound task $t \in \text{tn}_\alpha$
- note: input networks for a policy are quotiented out by their **(task network) isomorphism class**
- contrast to previous work (FOND$^{\text{FM}}$ HTN planning): a **sequence** of methods in M applied on tn_I followed by a **policy** on the decomposed network
- can extend to stochastic case by adding probabilities to actions

Isn't graph isomorphism hard?

- TN/DAG isomorphism is GI-complete [Behnke, Höller, and Biundo 2015]
  - create a new node for each original node
  - create a new node for each original edge
  - create a new directed edge from a new node-node to new edge-node corresponding to whether the original node was an endpoint of the original edge
- TN isomorphism practically also easy [Höller and Behnke 2021]
  - idea: hashing on layers of tasks in a task network
- almost all graphs easy to solve: `nauty` package [McKay and Piperno 2014]
  - idea: individualisation and (colour) refinement
- *hard* graphs are regular but almost never the case for TNs
  - colour refinement sufficient for almost all graphs [Babai, Erdös, and Selkow 1980]

Simple algorithms

Can compile a FOND$^{\text{MP}}$ HTN problem into a simple nondet. search problem:

- each search node consists of a **task network-state tuple** $\sigma_\alpha = (\mathrm{tn}_\alpha, s_\alpha)$
- a search node can be viewed as an FOND$^{\text{MP}}$ HTN **subproblem**
- transitions between search nodes correspond to choice of decomposition or primitive task transitions:
  - if a first task t in tn is primitive, define a nondet. transition

$$a = (\sigma_\alpha, \{\sigma_i = (\mathrm{tn}_\alpha \setminus \{t\}, s_i) \mid s_i \in \tau(t, s_\alpha)\})$$

  - else for each method applicable to t, define a det. transition

$$a = (\sigma_\alpha, \{\sigma_\beta = (\mathrm{tn}_\beta, s_\alpha)\}), \quad \text{s.t.} \quad \mathrm{tn}_\alpha \rightarrow^t_m \mathrm{tn}_\beta$$

Then solve with backwards search [Cimatti et al. 2003] or AND-OR search.

Complexity: HTN subclasses

- general HTN planning semidecidable, so clearly FOND$^{\mathrm{MP}}$ HTN at least as hard
- divide HTN planning problems into subclasses based on
  1. order of task networks: total or partial
  2. hierarchy classes of task networks:
     - primitive: no compound tasks
     - acyclic: no compound task can reach itself with decomposition
     - regular: at most one compound task in each network and is the last task
     - tail-recursive: $\sim$ acyclic $+$ regular

Complexity: membership proof ideas

- use simple algorithms described earlier
    1. compile into a nondet. state transition model
    2. solve with AND-OR or backwards search
- find upper complexity bounds

Complexity: hardness proof ideas

- reduce from alternating Turing machines (ATMs)
  - $\text{ASPACE}(f(n)) = \text{DTIME}(2^{O(f(n))}), \quad f(n) \geq \log(n)$
  - $\text{ATIME}(g(n)) = \text{DSPACE}(g(n)), \quad g(n) \geq \log(n)$
- use some tricks with some HTN classes (acyclic, regular, tail-recursive) in order to compactly encode ATMs for reduction
  - acyclic problems can compactly encode an exponential number of tasks
  - regular problems can model nondet. planning; or just reduce directly from polynomially bounded ATMs w.r.t. space
  - tail-recursive proof extends proof of deterministic version which uses a scheduling style reduction [Alford, Bercher, and Aha 2015]

Results

Table: Complexity results for FOND^{MP} HTN planning. The first column lists known special cases by restricting the hierarchy. Classes marked $*$ are not complete where only membership is known. Weak = deterministic for almost all subclasses.

Hierarchy	Order	Det.	Weak	Strong	Strong cyclic
primitive	total	P	NP	P*	P*
	partial	NP	NP	PSPACE	PSPACE
acyclic	total	PSPACE	PSPACE	EXPTIME	EXPTIME
	partial	NEXPTIME	NEXPTIME	EXPSPACE	EXPSPACE
regular	total	PSPACE	PSPACE	EXPTIME	EXPTIME
	partial	PSPACE	PSPACE	EXPTIME	EXPTIME
tail-recursive	total	PSPACE	PSPACE	EXPTIME	EXPTIME
	partial	EXPSPACE	EXPSPACE	2-EXPTIME	2-EXPTIME*

Conclusion

Takeaway:

- FOND^{MP} HTN:
    - nondet. HTN planning with *decomposition* selection as part of the solution
- almost all problem classes to be one class harder in the complexity heirarchy

Possible future work:

- benchmarks for nondet. and stochastic HTNs
- less naive algorithms and implementations of solvers