Hierarchical Problem Networks for Knowledge-Based Planning

Pat Langley
Institute for the Study of Learning and Expertise

Howard E. Shrobe
Massachusetts Institute of Technology

Ninth Annual Conference on
Advances in Cognitive Systems
August 15–18, 2021

Introductory Remarks

Domain expertise plays an essential role in both human and machine intelligence.

This holds across all cognition, but we will focus on sequential goal-directed activities.

- People regularly use such knowledge to carry out routine but complex procedures to achieve their goals.
- Miller, Galanter, and Pribram (1960) referred to these mental structures as ***plans***, although we now use other terms.

In this talk, we offer a new approach – ***hierarchical problem networks*** – to representing and using procedural expertise.

Target Abilities

We desire a theory of procedural expertise that explains major features of human problem solving, including the ability to:

- Generate novel action sequences that achieve sets of goals
- Consider both goals and situations when selecting actions
- Decompose complex activities into simpler ones when needed
- Use knowledge to guide or constrain search when available
- Carry out search through a problem space when necessary

Existing frameworks for knowledge-based planning support these abilities, but they also have limitations.

Hierarchical Task Networks

Hierarchical task networks (*HTNs*) exhibit these abilities; they encode knowledge as sets of *methods*, each with:

- *A task name and arguments*
- *State conditions under which the method applies*
- *A sequence of component subtasks*

Such knowledge structures can encode conditional, disjunctive, and even recursive procedures.

They are useful for generating, executing, and understanding complex goal-directed activities.

Hierarchical Goal Networks

Hierarchical goal networks (*HGNs*) are similar but differ in that each method specifies:

- *A general goal that it achieves*
- *State conditions under which the method applies*
- *A sequence of component subgoals*

HGNs cannot encode arbitrary procedures like HTNs, but they support goal-directed processing more directly.

Both frameworks combine ideas from classical programming languages with modular, relational notations.

Limits of HTNs and HGNs

Despite their strengths, classic HTNs and HGNs cannot state general procedures for problems with ***goal interactions***.

- *E.g., an HTN for building towers in the Blocks World needs a separate method for each number of target blocks:*

  (build-tower ?X ?Y), (build-tower ?X ?Y ?Z), and so forth

- *We can specify more general HTN programs, but they would then require search to find successful plans.*

HGNs also need one method for each type of tower, as they cannot encode knowledge about goal orderings.

We have developed a new framework – *hierarchical problem networks* – that overcomes this limitation.

Representational Postulates

The theory of hierarchical problem networks (HPNs) makes a number of familiar representational assumptions:

- A ***state*** is a set of relational facts that describe a situation.
- A ***goal*** is a desired relation and a ***problem*** is a set of goals.
- A ***hierarchical plan*** is a recursive decomposition of a problem into subproblems.
- Procedural knowledge is a set of ***methods*** that specify how to decompose problem goals into subproblems.
- Each method specifies ***state-related*** conditions under which a decomposition is acceptable.

The HPN framework shares these postulates with hierarchical goal networks, its closest relative.

Representational Postulates

However, the HPN theory also makes some more distinctive representational claims:

- Each method specifies how to decompose ***an entire problem*** (a set of goals) into an ordered set of subproblems.
- Each method specifies ***goal-related*** conditions under which a decomposition is ***not*** acceptable.
- Each method's ***head*** is some effect of an operator O, some of O's conditions as a ***first subproblem***, and O as a ***second subproblem***.

These structural constraints give HPNs a very different feel from classic HTNs or even HGNs.

They come closer to generalized partial-order plans that can handle tasks with an arbitrary number of objects.

HPN Methods for Blocks World

This HPN knowledge base for the Blocks World has six methods.

Each specifies a *head*, a set of *state conditions*, an ordered set of *subproblems*, and a set of goal conditions.

The latter serve as constraints on the order of goal processing.

```
((on ?X ?Y)
 :conditions   ((block ?X)(block ?Y))
 :subproblems  (((clear ?Y)(holding ?X))((stack ?X ?Y)))
 :unless-goals ((on ?Y ?ANY)(ontable ?Y)))
((holding ?X)
 :conditions   ((block ?X)(ontable ?X))
 :subproblems  (((clear ?X)(hand-empty))((pickup ?X)))
 :unless-goals ((clear ?ANY)))
((holding ?X)
 :conditions   ((block ?Y)(block ?X)(on ?X ?Y))
 :subproblems  (((clear ?X)(hand-empty))((unstack ?X ?Y)))
 :unless-goals ((clear ?ANY)))
((clear ?Y)
 :conditions   ((block ?Y)(block ?X)(on ?X ?Y))
 :subproblems  (((clear ?X)(hand-empty))((unstack ?X ?Y))))
((hand-empty)
 :conditions   ((block ?X)(holding ?X))
 :subproblems  (((putdown ?X)))
 :unless-goals ((clear ?ANY)))
((ontable ?X)
 :conditions   ((block ?X))
 :subproblems  (((holding ?X))((putdown ?X))))
```

HPN Operators for Blocks World

Here are six operators for the Blocks World that describe the effects of actions under certain conditions.

```
((stack ?X ?Y)
 :conditions   ((block ?X)(block ?Y)(holding ?X)(clear ?Y))
 :effects      ((on ?X ?Y)(hand-empty)(not(clear ?Y))(not(holding ?X))))
((pickup ?X)
 :conditions   ((block ?X)(ontable ?X)(clear ?X)(hand-empty))
 :effects      ((holding ?X)(not(hand-empty))(not(ontable ?X))))
((unstack ?X ?Y)
 :conditions   ((block ?X)(block ?Y)(on ?X ?Y)(clear ?X)(hand-empty))
 :effects      ((clear ?Y)(holding ?X)(not(on ?X ?Y))(not(hand-empty))))
((putdown ?X)
 :conditions   ((block ?X)(holding ?X))
 :effects      ((ontable ?X)(hand-empty)(not(holding ?X))))
```

These are equivalent to operators in STRIPS or PDDL notation, but they are stated in slightly different HPN syntax.

A Hierarchical Plan for Blocks World

```
(a) Initial state:
    ((block A)(block B)(block C)(block D)
     (ontable A)(ontable B)(ontable C)(ontable D)
     (clear A)(clear B)(clear C)(clear D)(hand-empty))
(b) Goal description:
    ((on A B)(on B C)(ontable C))
```

Here is an HPN plan for the initial state and problem goals that are shown above.

Each nonterminal problem has three subproblems, with the second being an *operator*.

This plan is right branching, but left-branching and mixed structures can also occur.

```
((on A B)(on B C)(ontable C))
    ((clear C)(holding B))
        ((clear B)(hand-empty))
        ((pickup B))
        ((clear C))
    ((stack B C))
    ((ontable C)(on A B))
        ((clear B)(holding A))
            ((clear A)(hand-EMPTY))
            ((pickup A))
            ((clear B))
        ((stack A B))
        ((ontable C))
```

Processing Postulates

The theory of hierarchical problem networks also incorporates postulates about processing:

- Planning ***recursively decomposes*** a problem into subproblems to find an operator sequence that achieves the problem's goals.
- Planning ***iteratively*** examines the topmost element on the ***problem stack*** and places new subproblems above it.
- Problem decomposition relies on three main subprocesses: method ***matching***, method ***selection***, and method ***expansion***.
- Planning involves ***search through a space of decompositions*** defined by methods, problem goals, and initial state.

However, it shares these assumptions with HTNs and HGNs; the key difference lies in their *representations*.

The HPD Interpreter

The HPD planner implements these processing postulates with a control cycle that covers four situations:

- *If the topmost problem P matches the current state S (all goals are satisfied), then pop P from the problem stack.*
- *If topmost problem P is to apply operator O and O's conditions match state S, then use O's effects to update S and remove P.*
- *If a method M applies to problem P in state S, then push M's subproblems onto stack and add them as P's children in plan.*
- *If the plan is too long or no method applies to problem P, then pop P from stack and remove P and its siblings from the plan.*

Given appropriate knowledge, HPD mimics a deterministic procedure, but it falls back on search when needed.

Matching an HPN Method

Consider the HPN method for getting one block on top of another:

```
((on ?X ?Y)
  :conditions      ((block ?X) (block ?Y))
  :subproblems     (((clear ?Y) (holding ?X)) ((stack ?X ?Y)))
  :unless-goals    ((on ?Y ?ANY) (ontable ?Y)))
```

Suppose the top problem is *((on A B) (on B C))* and the state is:

((block A) (block B) (block C) (ontable A) (ontable B) (ontable C) (clear A) (clear B) (clear C) (hand-empty))

Here the head (on ?X ?Y) matches in two different ways:

- (on A B) matches with bindings $?X \rightarrow A$ and $?Y \rightarrow B$
- (on B C), matches with bindings $?X \rightarrow B$ and $?Y \rightarrow C$

But the first match fails because the *:unless-goals* condition, (on ?Y ?ANY), matches a goal, (on B C), that is *not* satisfied.

Demonstrations of HPD's Abilities

To demonstrate HPD's abilities, we developed HPN knowledge bases for three planning domains:

- Blocks World – *Changing an initial layout to a target layout*
  - ***Six methods***, four operators, six predicates, twenty problems
- Logistics – *Transporting packages from initial to target locations*
  - ***Seven methods***, six operators, nine relations, ten problems
- Depots – *Moving crates between pallets and stacking them*
  - ***Eight methods***, five operators, twelve predicates, ten problems

For every problem in each domain, it found a hierarchical plan (from 4 to 18 steps) without backtracking.

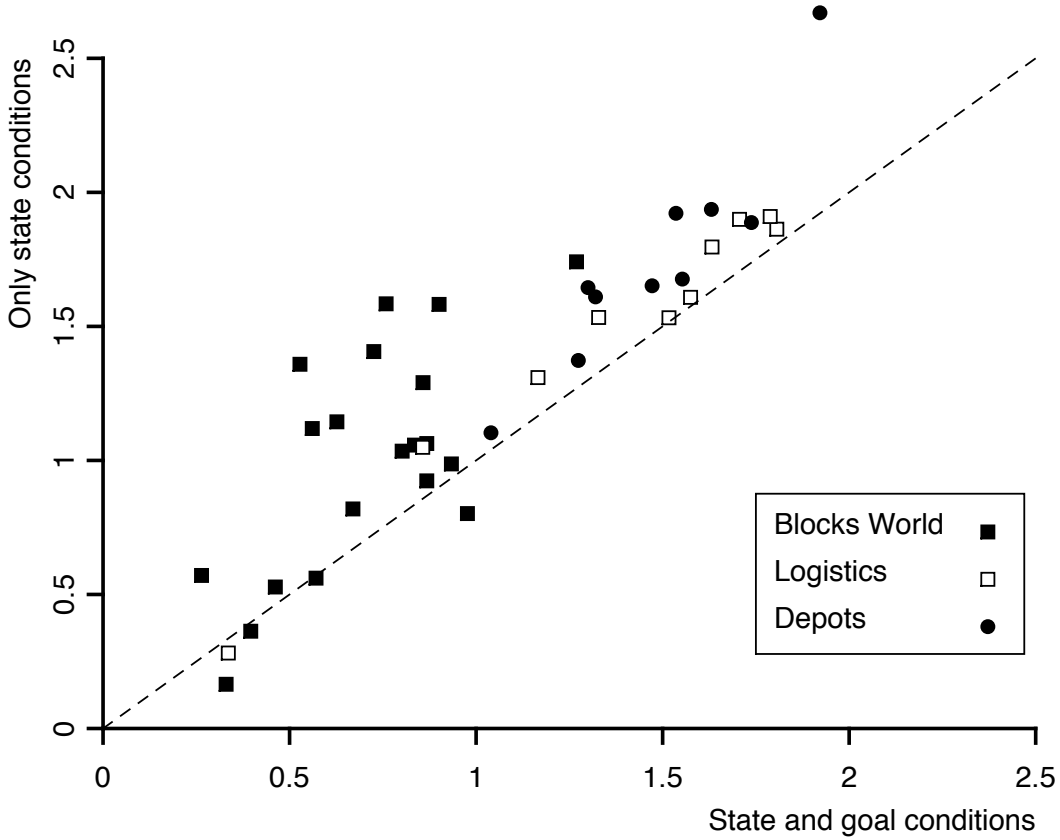
This shows HPNs can represent general procedures effectively.

Lesion Studies of HPD Planning

Scatter plot that compares effort by HPD programs with *both* state and goal conditions to one with *only* state conditions.

Each axis shows CPU time on a logarithmic scale.

Each point is an average of 30 separate runs.

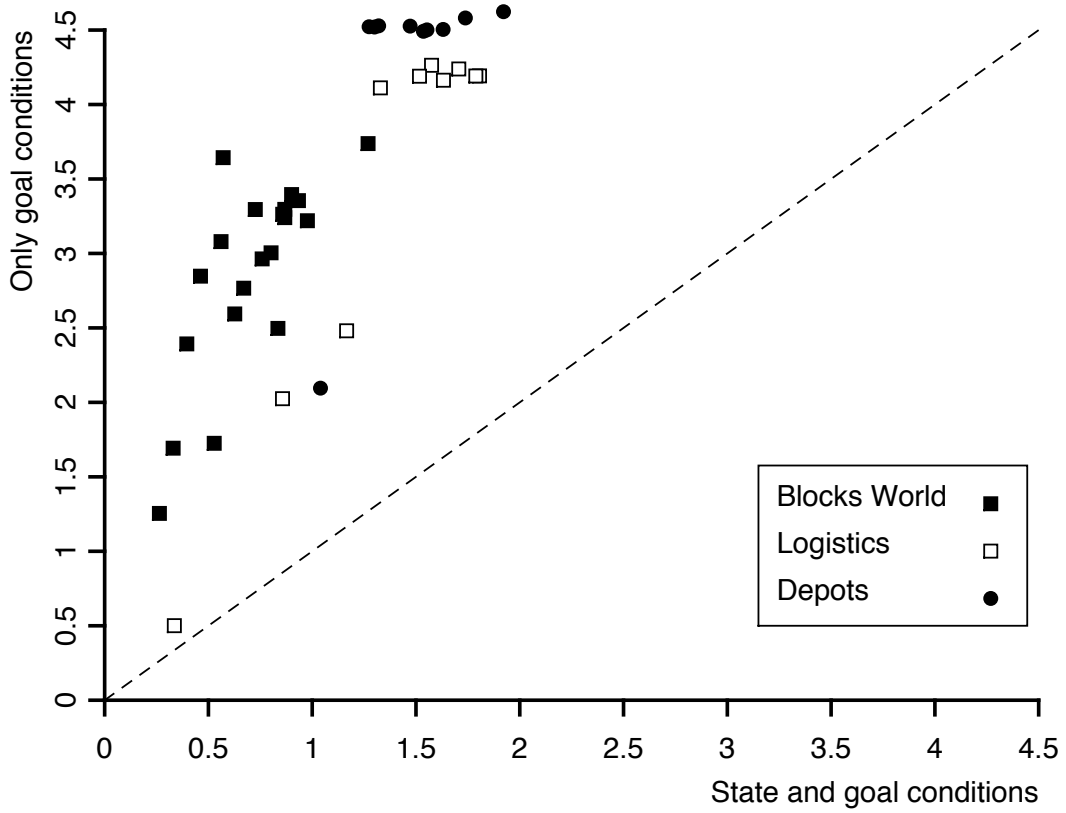


Lesion Studies of HPD Planning

Scatter plot that compares effort by HPD programs with *both* state and goal conditions to one with *only* goal conditions.

Each axis shows CPU time on a logarithmic scale.

Each point is an average of 30 separate runs.

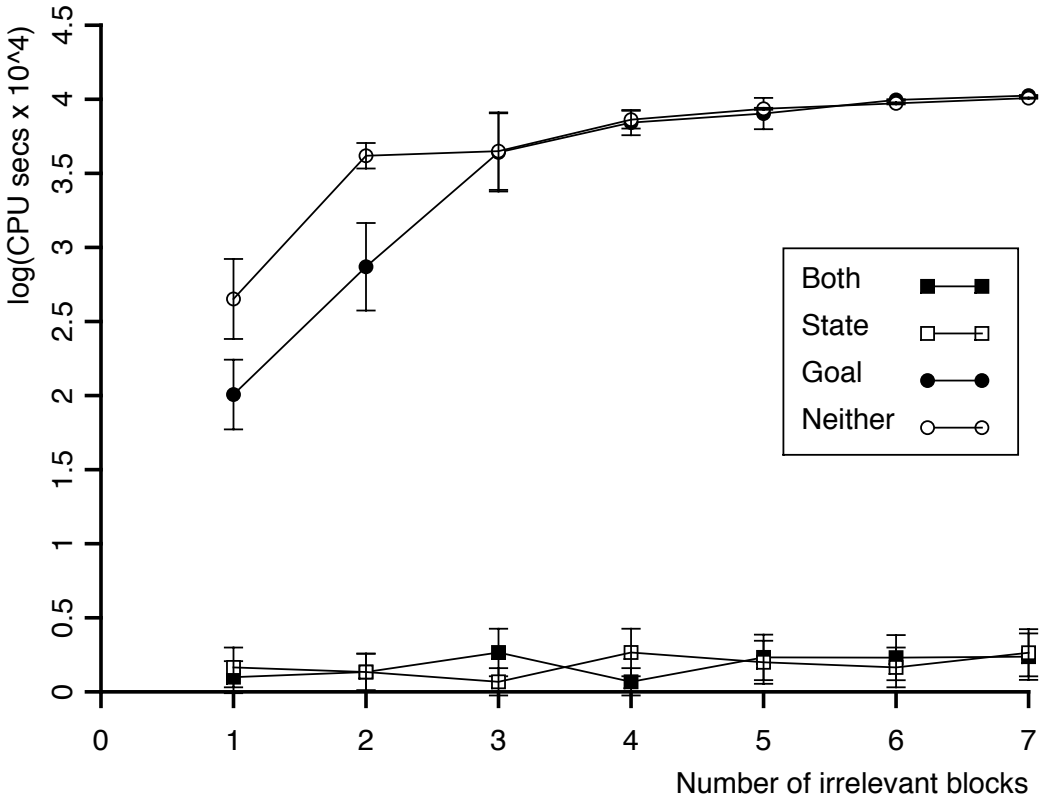


Scaling Studies of HPD Planning

Scaling curves that show time to solve Blocks World problems for different HPD programs as function of number of *irrelevant blocks*.

Each curve shows CPU time on a logarithmic scale.

Each point is an average of 30 separate runs.

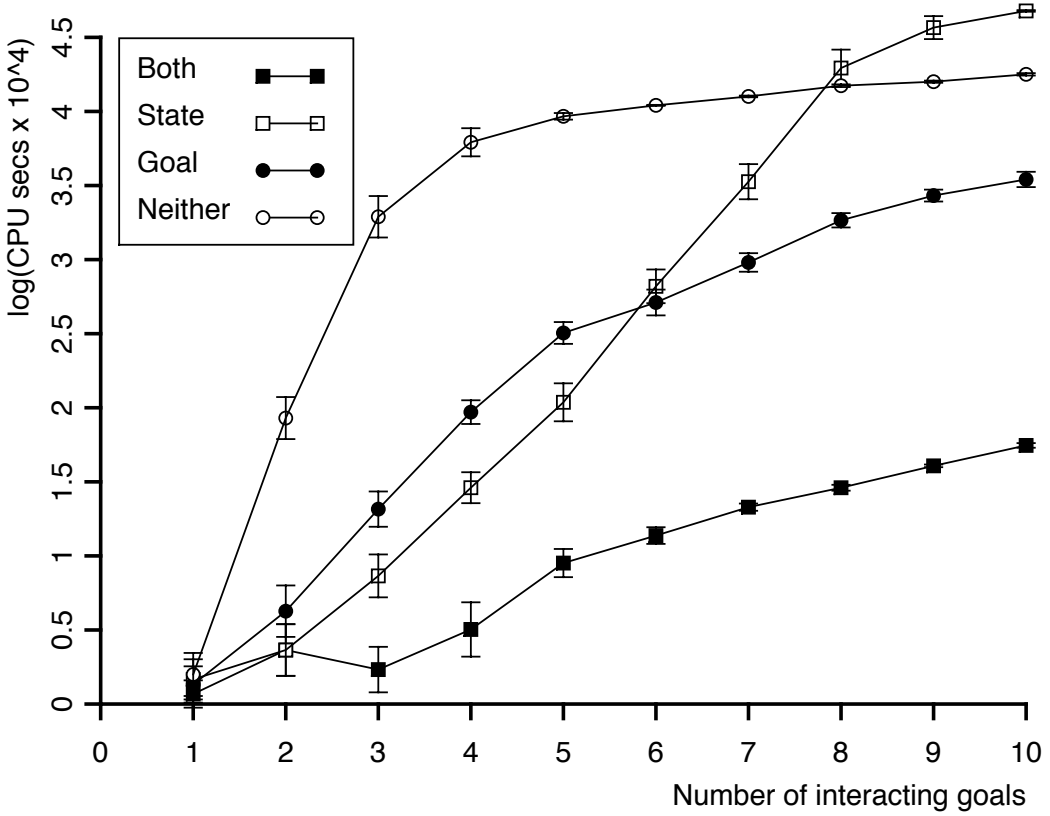


Scaling Studies of HPD Planning

Scaling curves that show time to solve Blocks World problems for different HPD programs as function of number of *interacting goals*.

Each curve shows CPU time on a logarithmic scale.

Each point is an average of 30 separate runs.



Relation to Earlier Research

The theory of hierarchical problem networks incorporates key ideas from the previous literature:

- Procedural knowledge is organized into modular, hierarchical methods (Slagle, 1963; Lloyd, 1984; Nau et al., 2003)
  - *But HPNs state how to decompose problems, not tasks or goals*
- Knowledge can specify what goal orderings are acceptable (Laird, 2012; Minton, 1988; Goldman & Kuter, 2019)
  - *But HPNs incorporate goal constraints into methods themselves*
- Problem solving decomposes problems into subproblems (Newell et al., 1960; Jones & Langley, 2005; Marsella, 1993)
  - *But HPNs emphasize procedures rather than heuristic search*

Thus, the theory offers a new and promising alternative for knowledge-based planning.

Comparison to Other Frameworks

Hierarchical problem networks have similarities to earlier planning frameworks, but they also introduce important differences.

Representational and Processing Assumptions	*Classic Planners*	*HTN Planners*	*HGN Planners*	*HPN Planners*
Generate sequential plans that achieve goals	●	●	●	●
Decompose complex activities hierarchically	○	●	●	●
Methods require that relations hold in state	○	●	●	●
Methods indexed by goals they achieve	○	○	●	●
Decompose problems into subproblems	○	○	○	●
Methods require that goals are not unsatisfied	○	○	○	●
Methods are linked to primitive operators	○	○	○	●

This table compares HPNs with HTNS, HGNs, and classic planners along seven dimensions that distinguish them.

Plans for Future Work

In future research, we plan to augment the HPN theory and its implementation in HPD to:

- Favor methods with fewer unsatisfied goals in subproblems
  - *This should reduce the current reliance on state conditions*
- Allow OR branches in HPNs for nondeterministic outcomes
  - *This should support information-gathering operators*
- Incorporate durative operators with temporal constraints
  - *This will support procedures that include parallel actions*
- Integrate HPN problem solving with partial-order planning
  - *Use methods if available but chain backward when needed*

The last extension may hold the key to ***learning*** state and goal conditions on HPN methods ***analytically***.

Summary Remarks

This talk introduced *hierarchical problem networks*, a new representation for procedural expertise that:

- *Specifies how to decompose problems into subproblems*
- *Includes state conditions that constrain bindings on variables*
- *Includes goal conditions that constrain order of goal processing*

We also described HPD, an planning architecture that:

- *Includes an interpreter that generates hierarchical plans*
- *Mimics deterministic procedures given the right knowledge*
- *Falls back on search when methods are lacking conditions*

Experiments identified how state and conditions reduce search and aid scaling to problem complexity.