Hierarchical Planning

CompSci 767

2011

Outline

- Introduction
- ABSTRIPS
- ALPINE/PRODIGY
- Summary

Introduction

- Heuristic value of *n* often calculated by mapping node to abstract space and solving there.
- However, when abstract solution found, only information that is returned is solution length.
- Everything else about the abstract solution is thrown away!!!!!!

- *Can we use the abstract solution as constraints on our ground solution?*
- *Why don't we see more hierarchical planners in use today?*

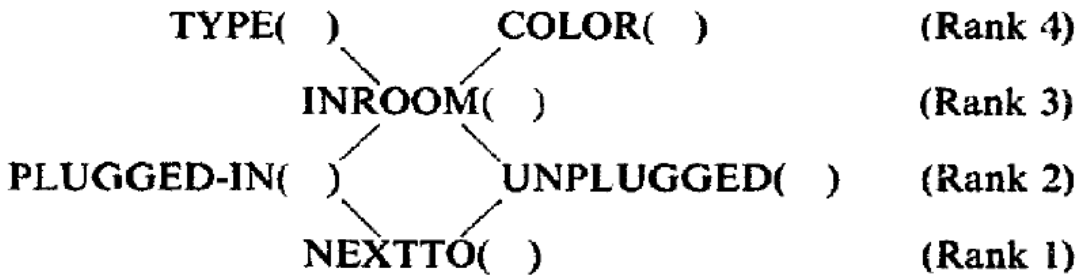
ABSTRIPS Overview

- Sacerdoti, 1974: "**Planning in a Hierarchy of Abstraction Spaces**", AIJ.
- Semi-automatically created an ordering of domain literals.
- This order defined abstraction levels for planning.
- Find solution at the most abstract level and recursively refine down to the ground level.
- If fails to solve subproblem at a level, then backtracks up to next higher level.

Determining ABSTRIPS Criticality Levels

- Given predetermined partial ordering of literals.
- All literals, whose truth value could not be changed, are assigned maximum value.
- Order of each remaining literal examined is determined by the partial ordering.
  - If short plan found to achieve literal from state when all previous literals were true, then literal assigned a criticality equal to its rank in partial ordering.
  - If no such plan found, literal assigned a criticality greater than the highest rank in partial order.

Assigning Criticality Levels

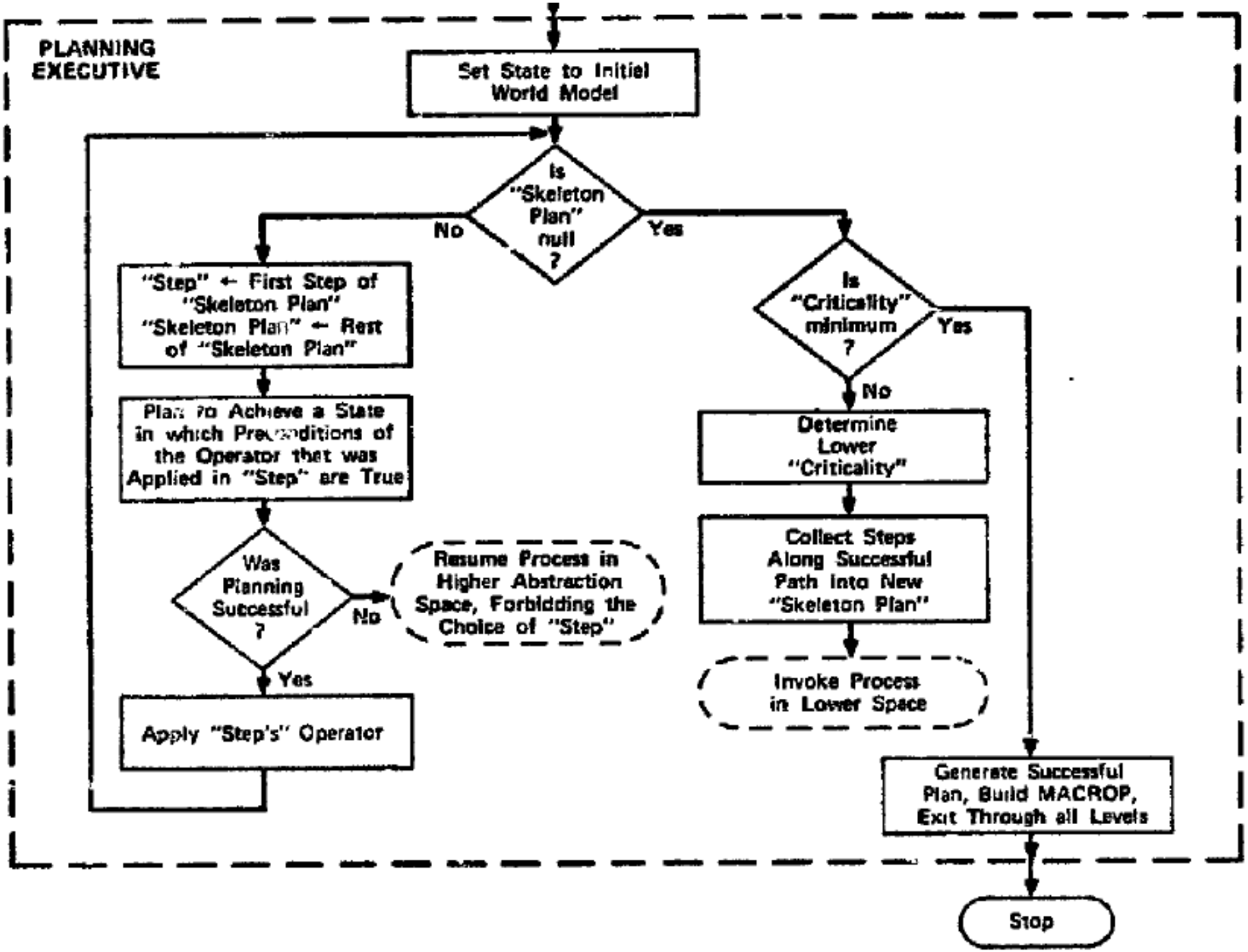


TurnOnLamp(*lamp***)**
pre: TYPE(*lamp*,**lamp**) ^ (**Exists r**x)(lNROOM(Me,rx) ^ INROOM(*lamp*,rx))
^ PLUGGED-IN(*lamp*) ^ NEXTTO(Me, *lamp*).

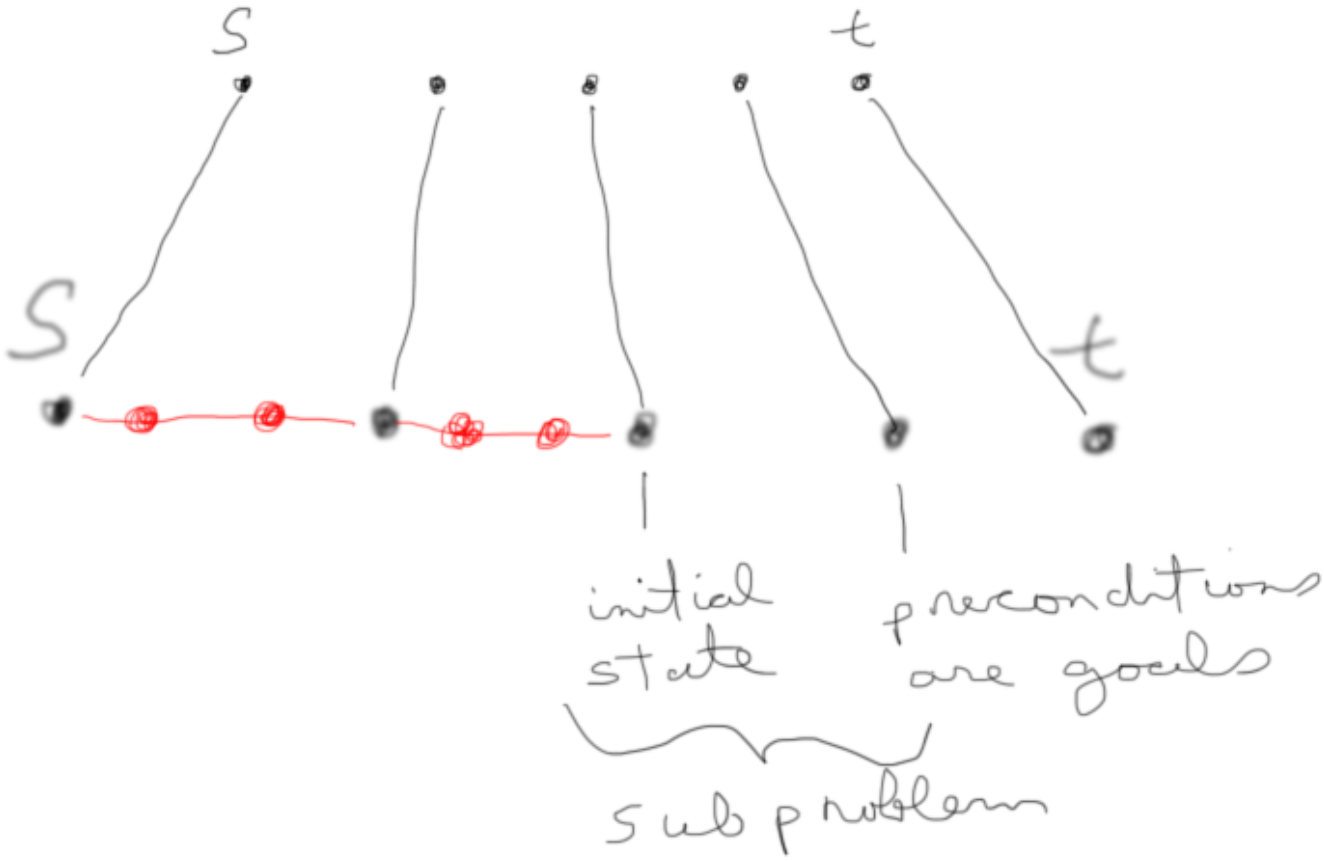
"TYPE" not changeable by any operator, so criticality -> 6.
"INROOM" cannot be achieved from just TYPE(*lamp*,**lamp**) is asserted,
so assigned a criticality > highest rank in partial order, i.e., -> 5
"PLUGGEDIN" can be achieved from "TYPE" and "INROOM", -> 2
"NEXTTO" can be achieved from "INROOM", -> 1

Using ABSTRIPS Criticality Levels

- Criticality level only apply to op preconditions.
- At “abstraction” level *L*, only “worry” about preconditions >= criticality level *L*.
- At top level, solve the problem but don’t worry about op preconditions.
- At lower levels, take abstract plan from above and now solve op preconds from this level.
- View abstract plan as sequence of problems.

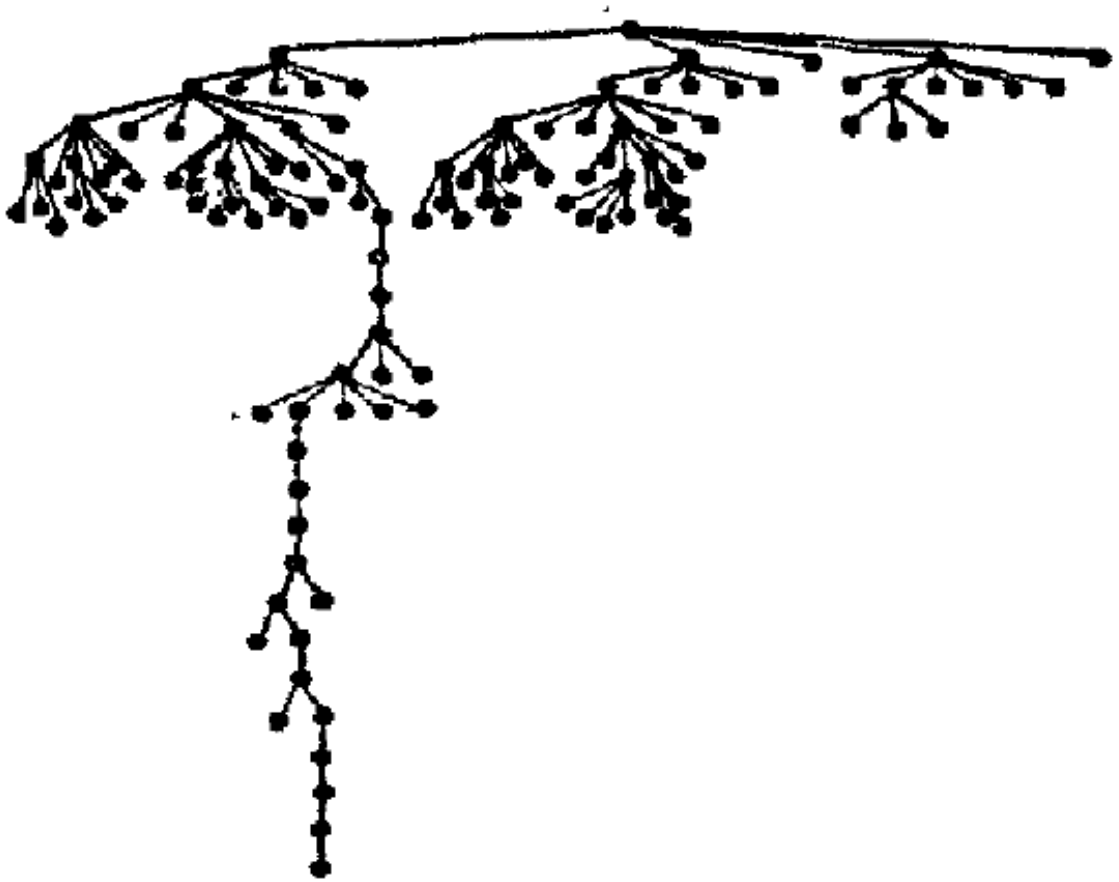

PLANNING EXECUTIVE
Set State to Initial World Model
Is "Skeleton Plan" null ?
No
Yes
"Step" ← First Step of "Skeleton Plan"
"Skeleton Plan" ← Rest of "Skeleton Plan"
Plan to Achieve a State in which Preconditions of the Operator that was Applied in "Step" are True
Was Planning Successful ?
No
Resume Process in Higher Abstraction Space, Forbidding the Choice of "Step"
Yes
Apply "Step's" Operator
Is "Criticality" minimum ?
Yes
No
Determine Lower "Criticality"
Collect Steps Along Successful Path into New "Skeleton Plan"
Invoke Process in Lower Space
Generate Successful Plan, Build MACROP, Exit Through all Levels
Stop

Abstract Plan Use



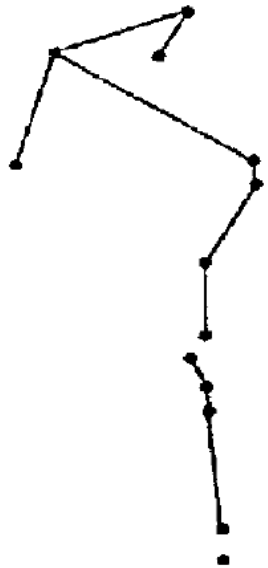
How well can ABSTRIPS do?

STRIPS

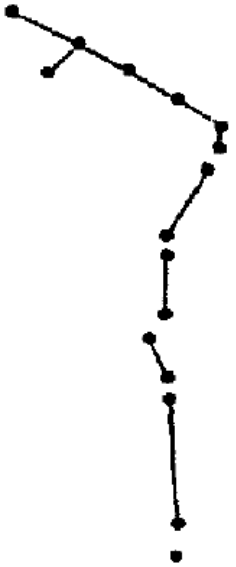


(a) STRIPS SEARCH TREE FOR THE SAMPLE PROBLEM

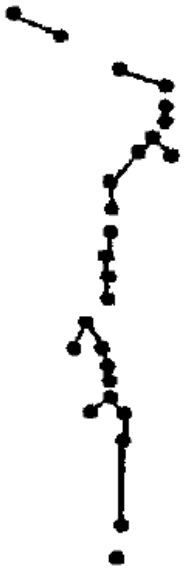
(b) ABSTRIPS SEARCH TREE IN THE SPACE OF CRITICALITY 6



(c) ABSTRIPS SEARCH TREES IN THE SPACE OF CRITICALITY 5



ABSTRIPS SEARCH TREES IN THE SPACE OF CRITICALITY 2



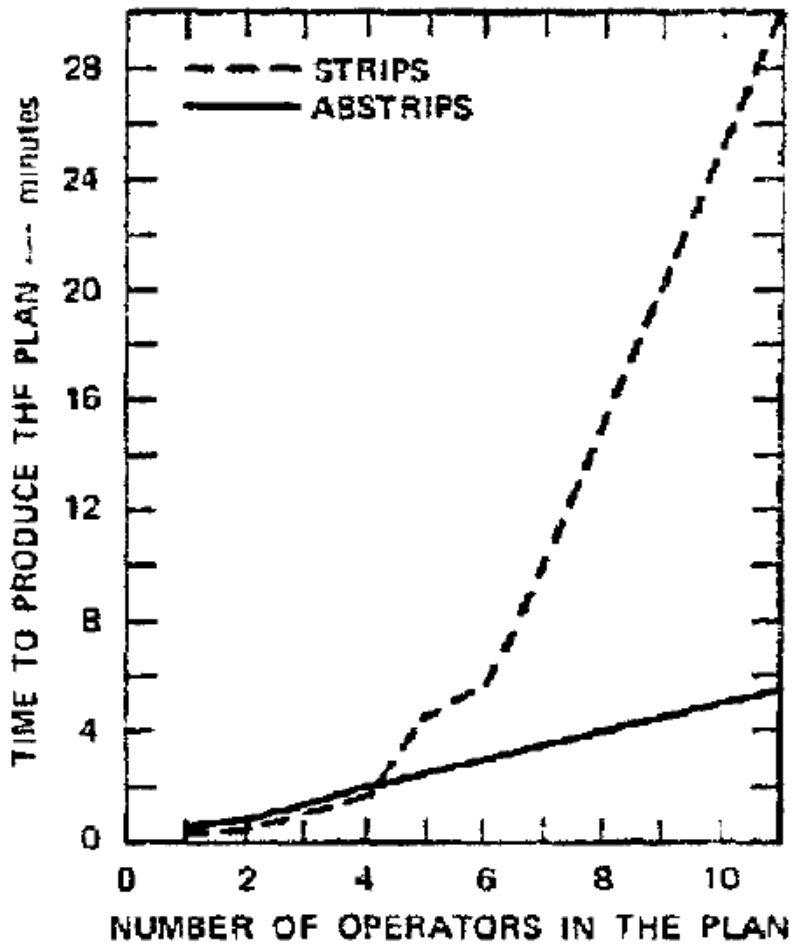
ABSTRIPS SEARCH TREES IN THE PROBLEM SPACE

ABSTRIPS vs STRIPS

Comparison of planning times and search trees

	PROBLEM 1	PROBLEM 2	PROBLEM 3	PROBLEM 4	PROBLEM 5
ABSTRIPS					
Time to find plan (minutes)	1 : 54	2 : 55	2 : 24	2 : 30	6 : 41
Total nodes in search trees	25	34	30	33	63
—by spaces[a]	5, 5, 5, 10	5, 7, 7, 15	3, 4, 11, 12	5, 7, 7, 14	5, 17, 16, 25
Nodes on solution path	24	32	28	32	54
—by spaces[a]	5, 5, 5, 9	5, 7, 7, 13	3, 4, 10, 11	5, 7, 7, 13	5, 11, 15, 23
Operators in plan	4	6	5	6	11
STRIPS					
Time to find plan (minutes)	1 : 40	5 : 44	4 : 34	9 : 47	>20 : 00[b]
Total nodes in search tree	10	33	22	51	—
Nodes on solution path	9	13	11	15	—
Operators in plan	4	6	5	7	—

Scaling Up


STRIPS
ABSTRIPS
TIME TO PRODUCE THE PLAN — minutes
28
24
20
16
12
8
4
0
0
2
4
6
8
10
NUMBER OF OPERATORS IN THE PLAN

ABSTRIPS Summary

- Showed that hierarchical planning could reduce complexity from exponential to polynomial.
- Semi-automated assignment of criticality level

- Had problems:
  - ignored subproblem interactions (deletes)
  - treated all goals at highest level
  - frequently backtracked across levels

ALPINE

- Knoblock, 1994, "**Automatically generating abstractions for planning**", AIJ.
- Fully automated partitioning of literals into abstraction hierarchy.
- Abstraction hierarchy guaranteed to satisfy *ordered monotonicity* property.
- Ordered monotonicity decreased backtracking across levels.

ABSTRIPS vs ALPINE approaches

- ABSTRIPS worked with ***relaxed*** versions of the problem (i.e., dropped precondition literals).
- ALPINE worked with ***reduced*** versions of the problem (i.e., removed literals from problem spaces).
- At abstraction level *L*, all lower level literals are removed from the state, the goal, and the operator descriptions.

Ordered Monotonicity

- Key Idea: literals established at one level are not affected by any operators that can be added at any lower level.
- An operator *affects* a literal if it can either add or delete that literal.
- Can create problem-independent hierarchies or problem-dependent ones.

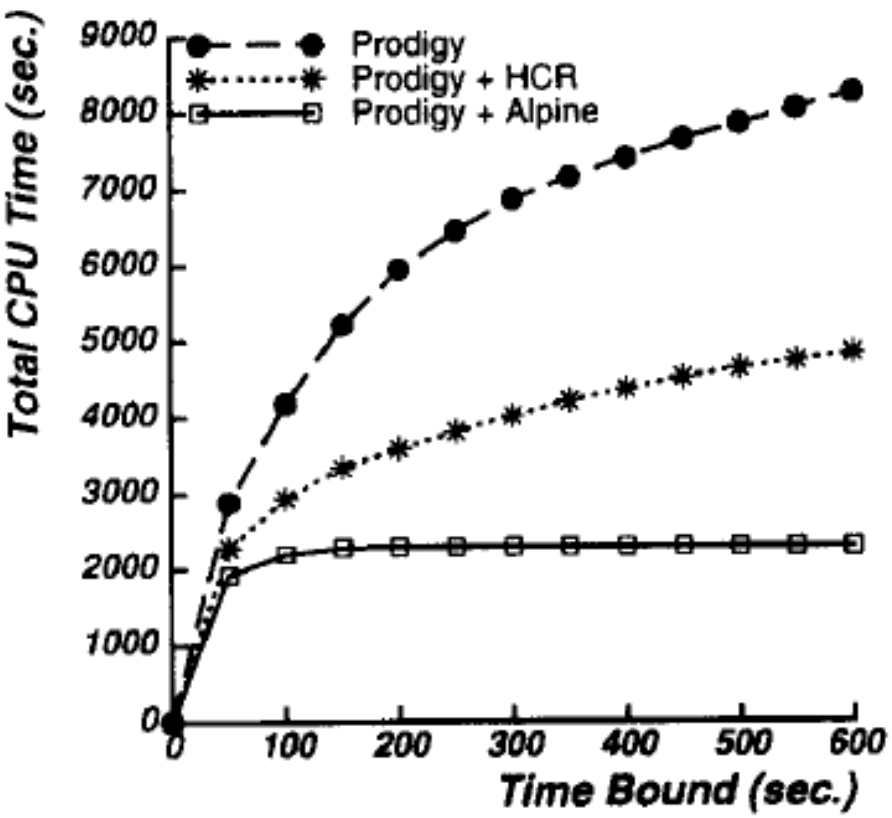
ALPINE

Problem-independent algorithm for determining constraints

Input: The operators that define the problem space.
Output: Sufficient constraints to guarantee ordered monotonicity.

```
function Find_Constraints(graph,operators):
  for each op in operators
    select lit1 in Effects(op)
      begin
      for each lit2 in Effects(op)
        begin
          Add_Directed_Edge(lit1,lit2,graph);
          Add_Directed_Edge(lit2,lit1,graph)
        end;
      for each lit2 in Preconditions(op)
        Add_Directed_Edge(lit1,lit2,graph)
      end;
  return(graph);
```

ALPINE vs Prodigy



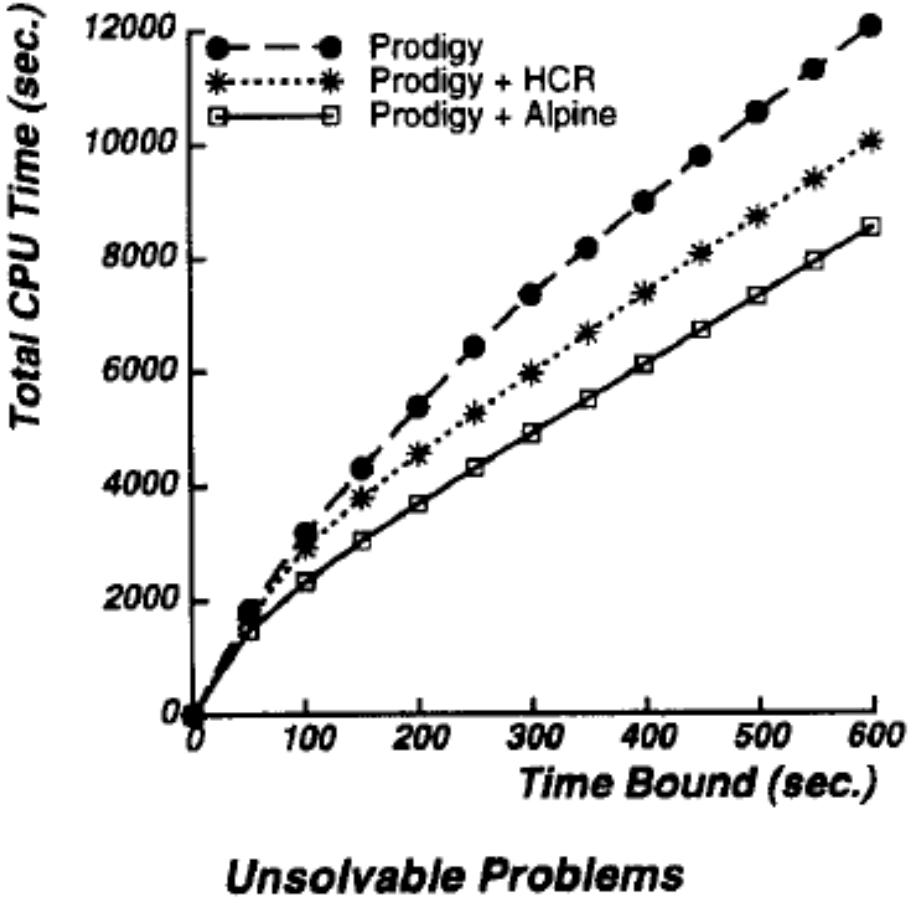


Fig. 10. Total CPU times in the robot planning domain.

ALPINE vs ABSTRIPS

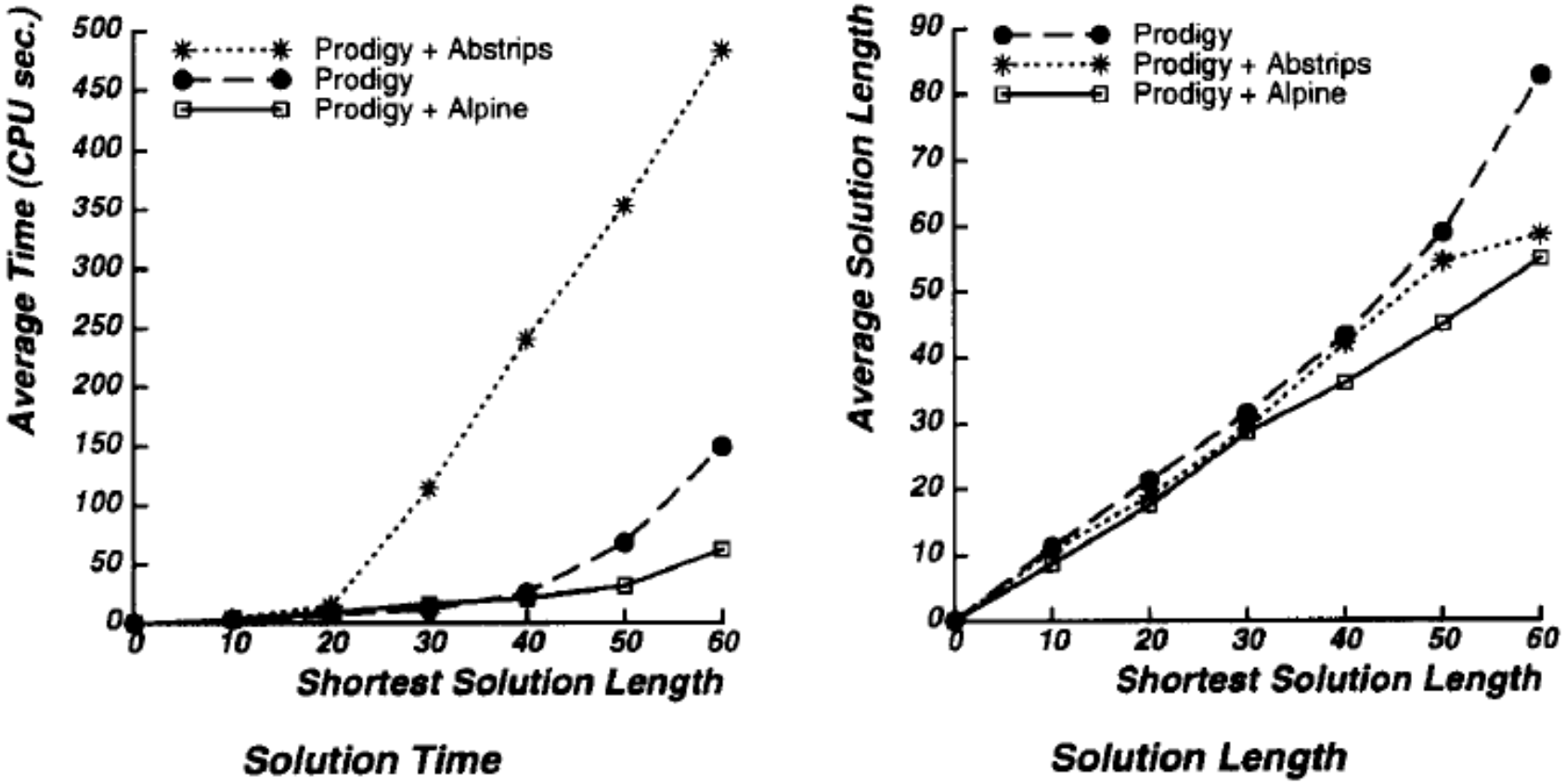


Fig. 18. Comparison of the average solution times and average solution lengths.

Experimental Caveat

- ABSTRIPS abstract hierarchy creation algorithm was design for best-first search
- ALPINE abstract hierarchy creation algorithm was design for depth-first search
- Depth-first search suffers more acutely from earlier bad decisions than does best-first.
- Can't ignore target search algorithm when designing abstraction creation algorithm!!!

ALPINE Summary

- Fully automated generation of abstraction hierarchy.
- Ordered monotonicity eliminated two sources of across level backtracking by:
  - using reduced models instead of relaxed models
  - tracking all sources of interactions
- Prodigy/ALPINE still backtracked when abstract plan unrefinable.
- Guaranteeing refinability called “downward refinability”, later systems tried to approximate this [Highpoint, Bacchus&Yang, 1994, “**Downward refinement and the efficiency of hierarchical problem solving”,** AIJ].

Status

- *Can we use the abstract solution as constraints on our ground solution?*
  - Yes, there are ways of automatically creating the abstraction hierarchies and of using them to generate subproblems at lower levels.
- *Why don't we see more hierarchical planners in use today?*
  - These approaches depend upon deleting preconditions – i.e., edge supergraphs.
  - Cannot guarantee any savings.
  - Can create exponentially longer solutions.

PostScript

- Weighted *A** and hierarchical search/planning:
  - Does lack of backtracking in hierarchical search imply weighted *A** can go directly to solution???
  - For what range of weights would this occur???
  - Once again, for what types of abstraction functions might this be true???

Epilogue

- Abstraction and hierarchical planning seem central to controlling complexity in real life.
- Why hasn't it worked out so far in AI?
- What are we missing?
  - Optimal planning vs conservative design.
  - No silver bullet!!!!
  - Need to better understand all the different ways of doing abstraction, approximation, and reformulation.