Searching Without a Heuristic: Efficient Use of Abstraction

Bradford Larsen
Ethan Burns
Wheeler Ruml

Robert C. Holte

University of New Hampshire

University of Alberta

Supported in part by the DARPA CSSG program, the NSF, and the NSERC.

Good Heuristics are Hard to Find

- Glued 15-puzzle: has unmovable tile
- Manhattan distance admissible
- Glued tile reduces effectiveness
- Natural solution: construct pattern database (PDB)

Start

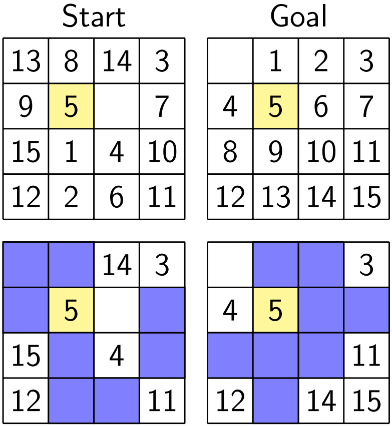
13	8	14	3
9	5		7
15	1	4	10
12	2	6	11

Goal

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Example: Glued 15-Puzzle PDB

- Abstract the puzzle by obscuring tiles
- Enumerate entire abstract state space backward from goal
- Store costs to goal in look-up table
- Use look-up table for heuristic estimates

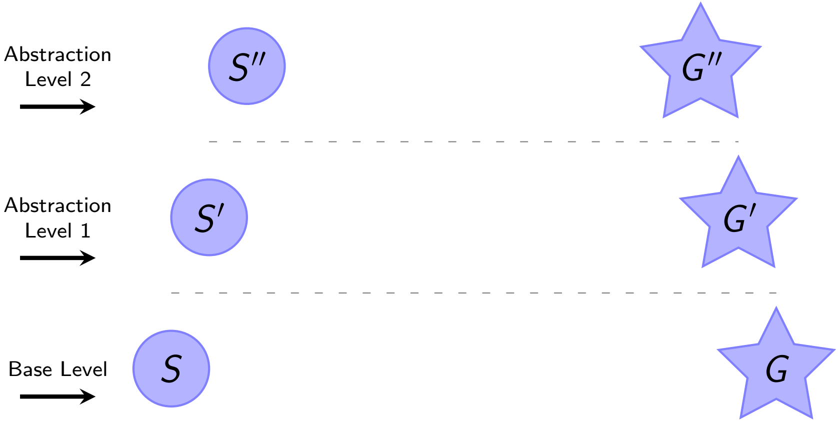


Pattern Database Shortcomings

Disadvantages when solving one or a few instances:

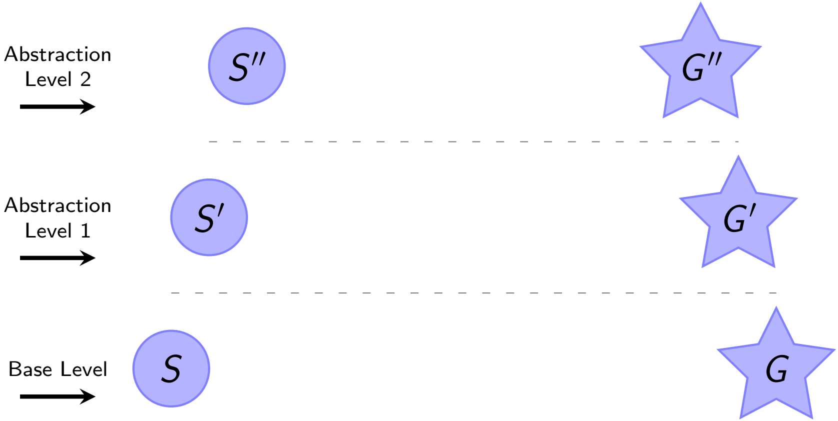
- Must enumerate *entire* abstract space
  - Expensive preprocessing phase
  - Database entry for *every* abstract state
- During single search, most entries go unused
- Database not reusable
  - when goal state changes
  - when operator costs change
- One abstraction for all instances

Hierarchical A* (Holte et al. 1996)



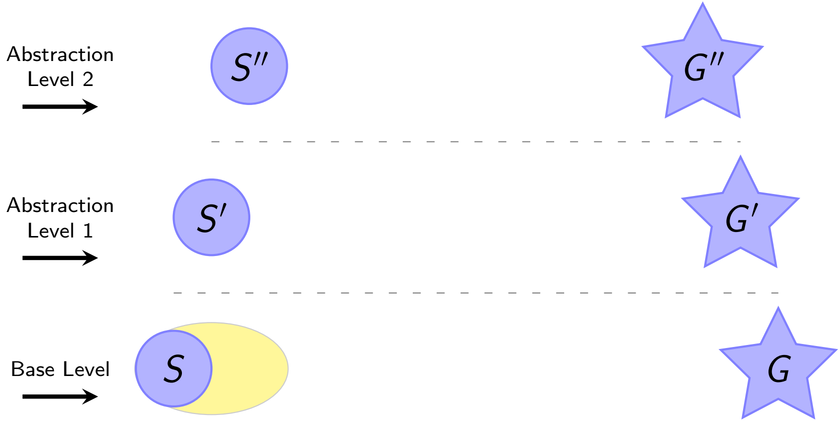
HA* uses a hierarchy of abstractions.

Hierarchical A* (Holte et al. 1996)



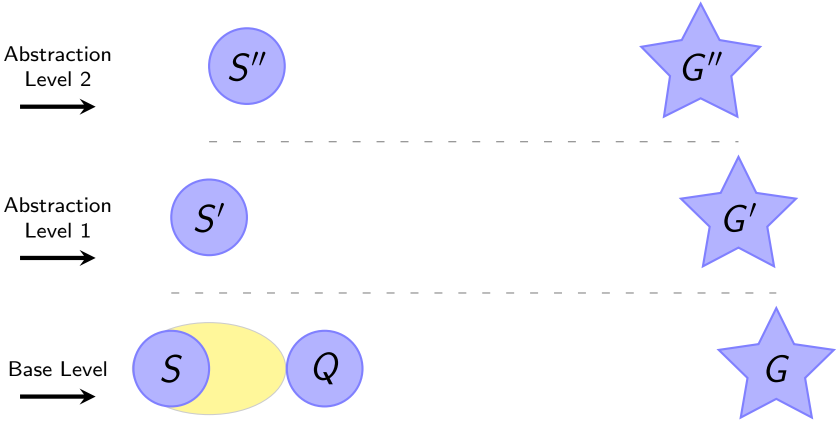
Objective: find the cheapest path from S to G.

Hierarchical A* (Holte et al. 1996)



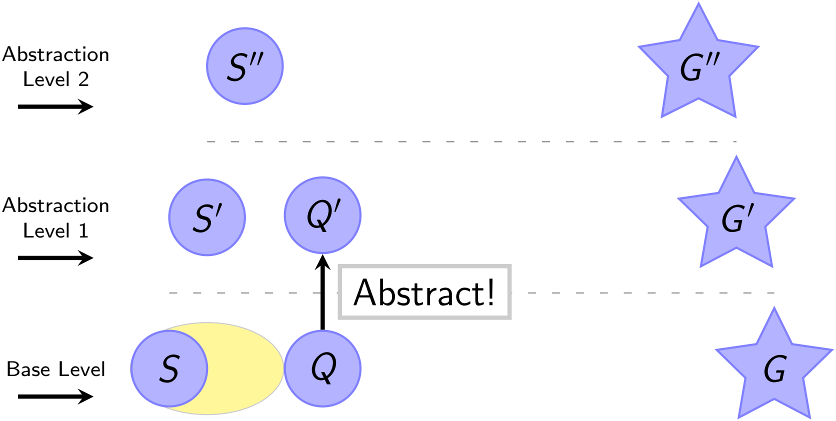
The yellow area represents generated states.

Hierarchical A* (Holte et al. 1996)



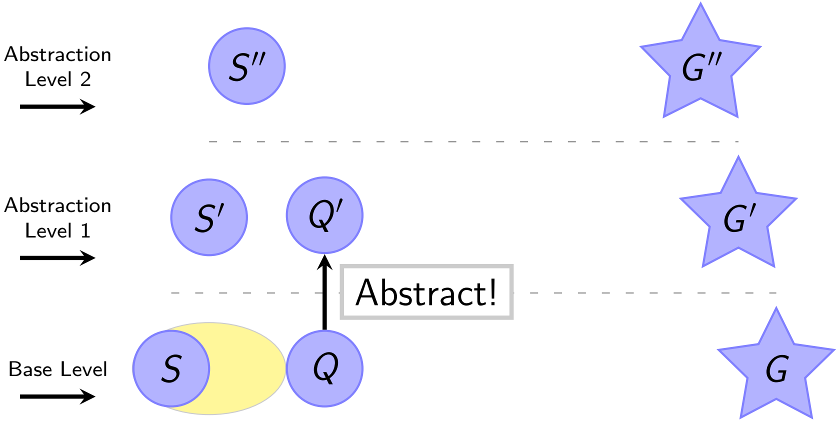
To generate Q, we need to know $h(Q)$.

Hierarchical A* (Holte et al. 1996)



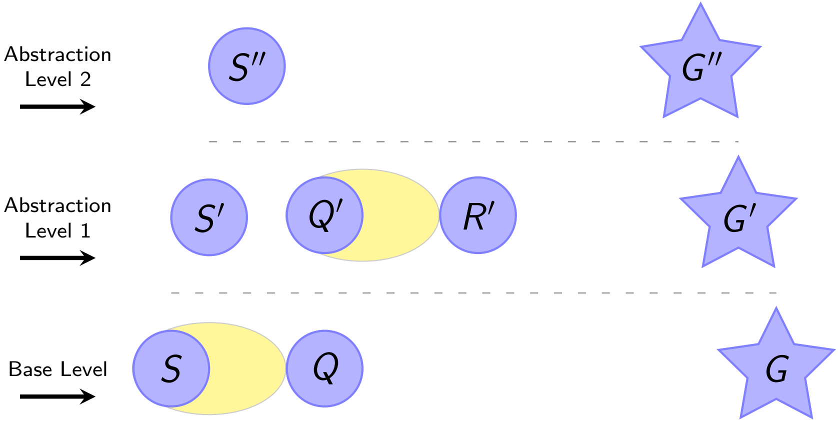
To find $h(Q)$: abstract Q and search at level 1.

Hierarchical A* (Holte et al. 1996)



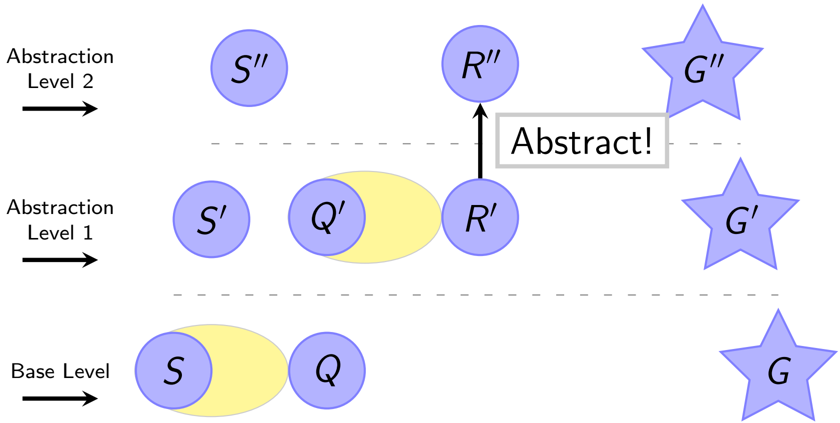
New objective: find cheapest path from Q' to G'.

Hierarchical A* (Holte et al. 1996)



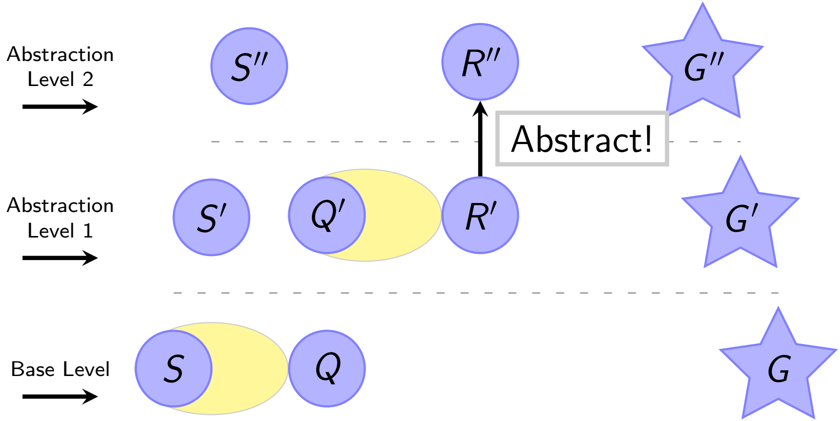
To generate R', we need to know $h(R')$.

Hierarchical A* (Holte et al. 1996)



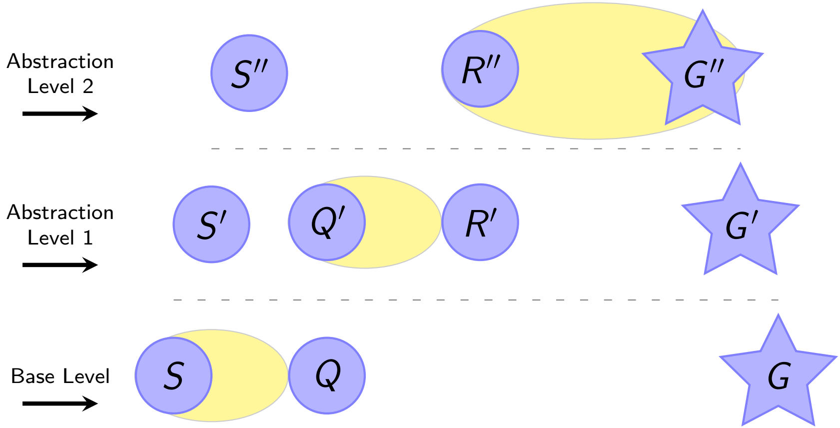
To find $h(R')$: abstract R' and search at level 2.

Hierarchical A* (Holte et al. 1996)



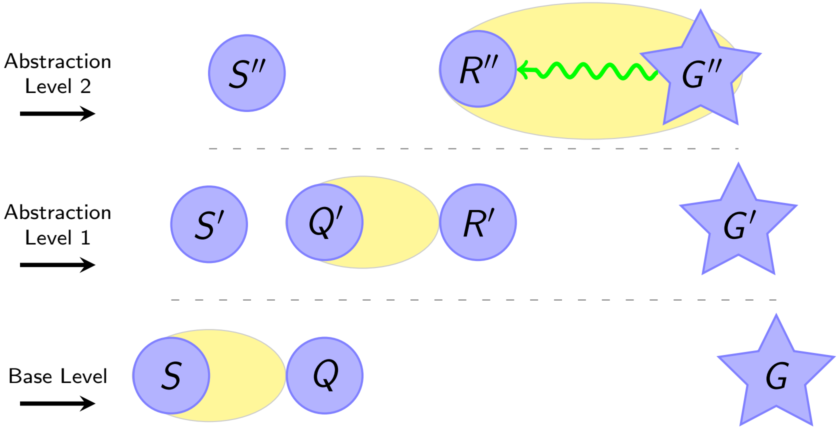
New objective: find cheapest path from R'' to G''.

Hierarchical A* (Holte et al. 1996)



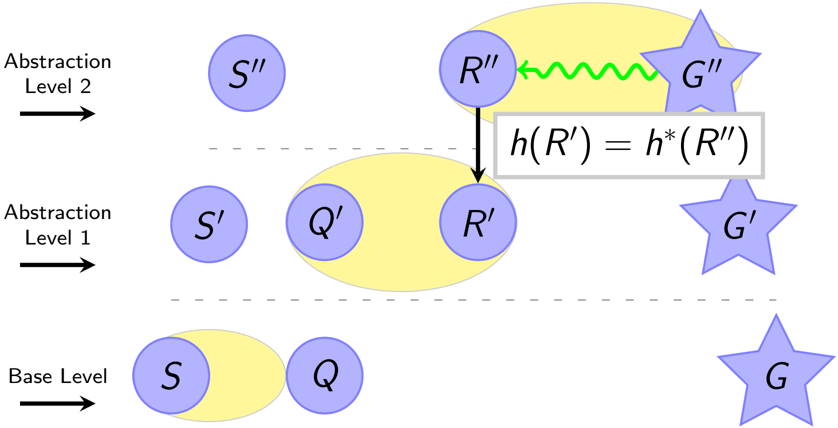
Use $h(n) = 0$ at the top level. Small search space!

Hierarchical A* (Holte et al. 1996)



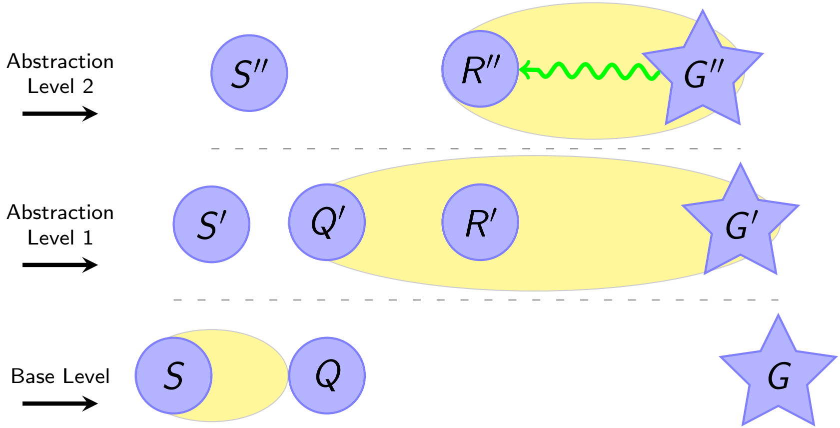
h^* values along the solution path are cached.

Hierarchical A* (Holte et al. 1996)



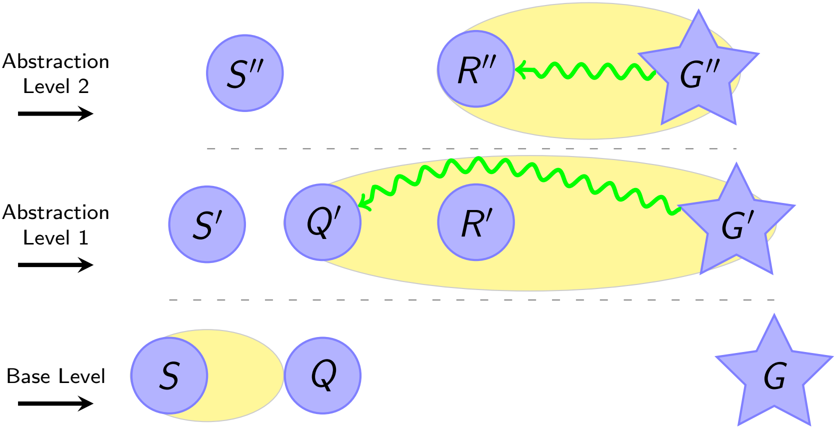
Use cost-to-goal at level 2 as $h(R')$.

Hierarchical A* (Holte et al. 1996)



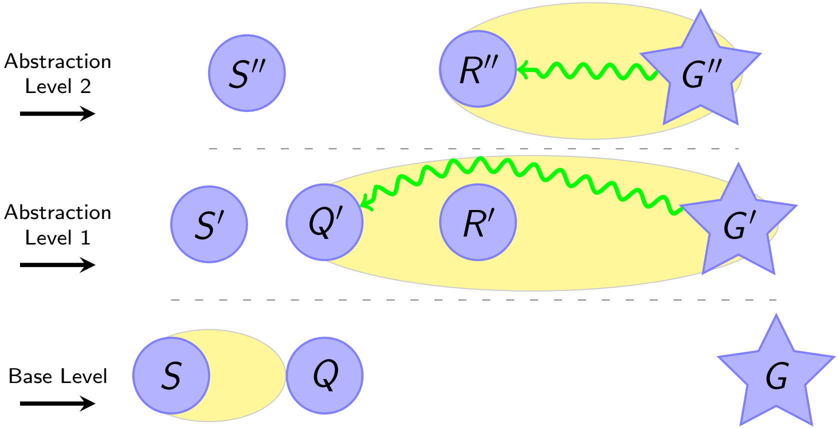
Eventually the search at level 1 finishes.

Hierarchical A* (Holte et al. 1996)



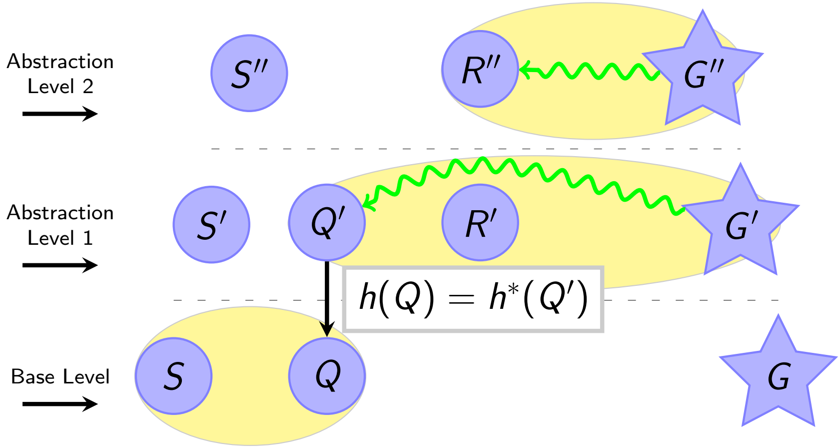
h^* values along the solution path are cached.

Hierarchical A* (Holte et al. 1996)



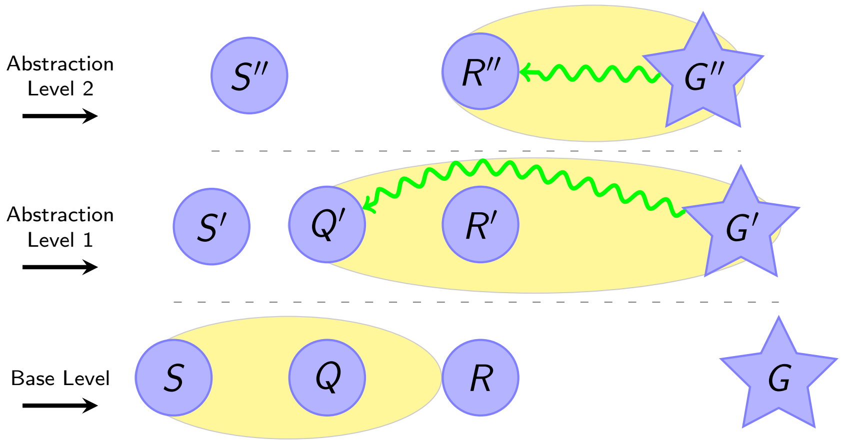
R' is not on the solution path.

Hierarchical A* (Holte et al. 1996)



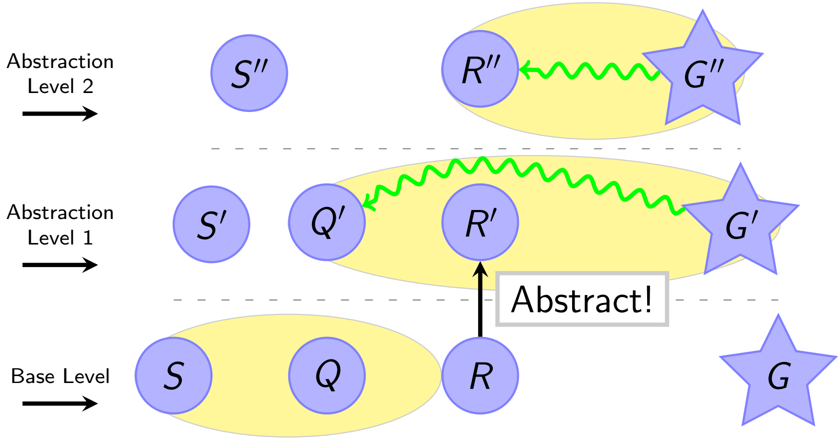
Use cost-to-goal at level 1 as $h(Q)$.

Hierarchical A* (Holte et al. 1996)



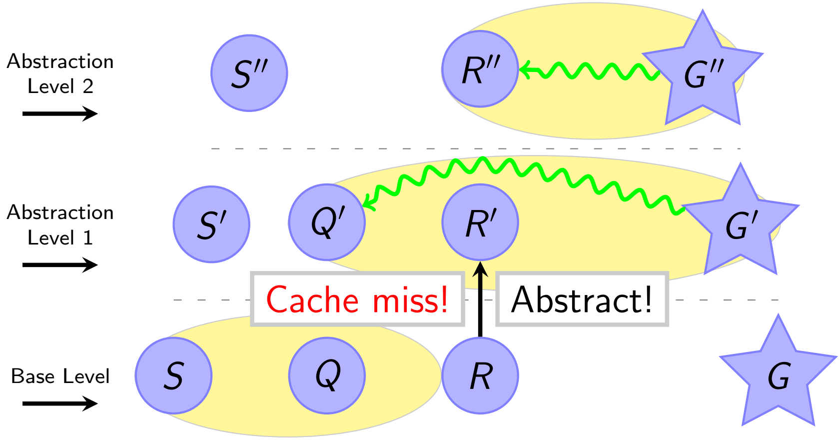
To generate R, we need to know $h(R)$.

Hierarchical A* (Holte et al. 1996)



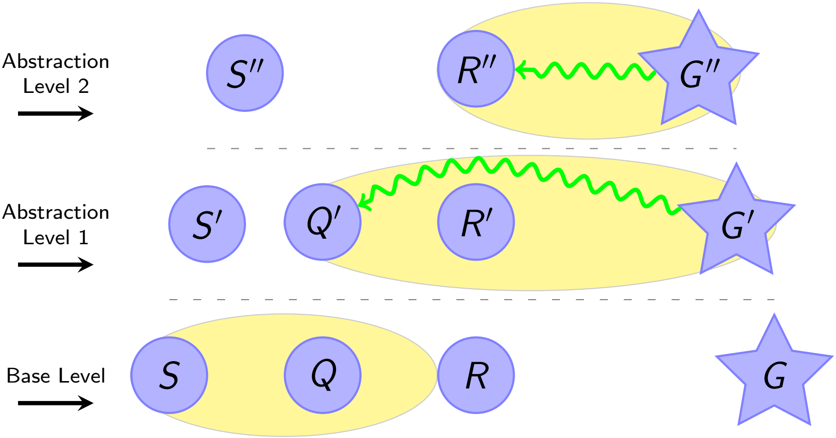
To find $h(R)$: abstract R and search at level 1.

Hierarchical A* (Holte et al. 1996)



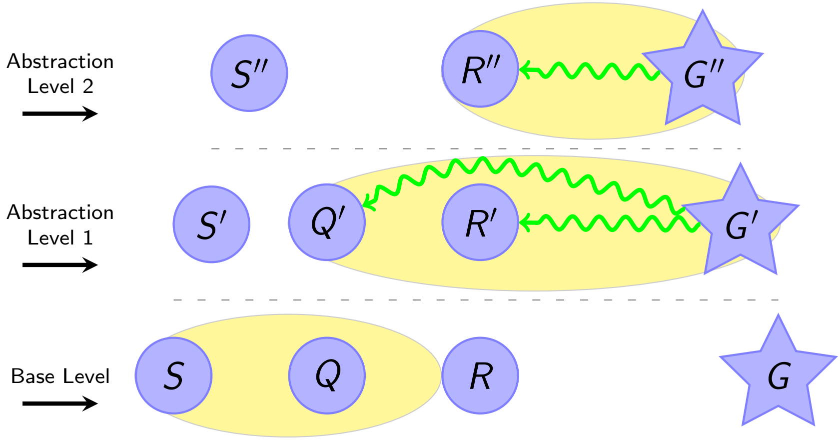
$h^*(R')$ is not cached: must search again.

Hierarchical A* (Holte et al. 1996)



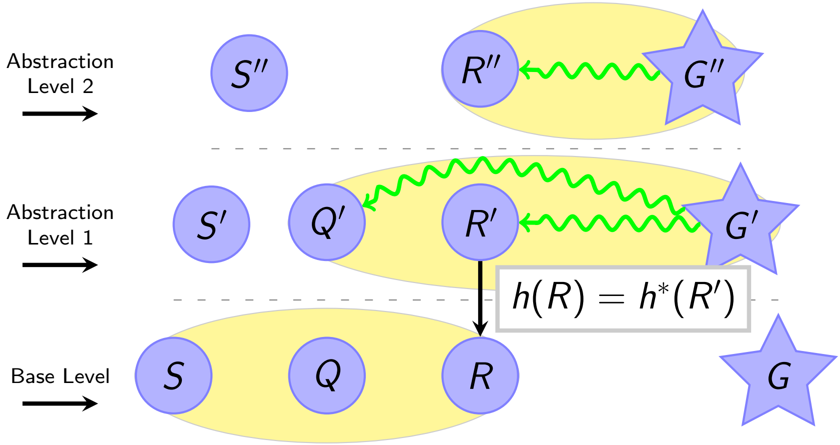
Eventually the search at level 1 finishes.

Hierarchical A* (Holte et al. 1996)



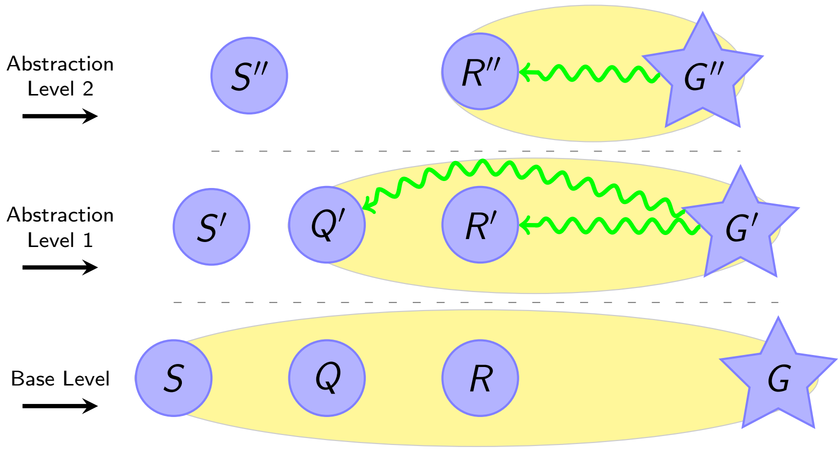
h^* values along the solution path are cached.

Hierarchical A* (Holte et al. 1996)



Use cost-to-goal at level 1 as $h(R)$.

Hierarchical A* (Holte et al. 1996)



Search proceeds in this manner until finished.

Hierarchical IDA* (Holte et al. 2005)

- Use modified IDA* at each level
- Used effective many-to-one abstractions
- Used instance-specific abstractions
  - specialized to each problem
  - more effective than domain-specific abstraction
- Complicated caching schemes *a la* HA*

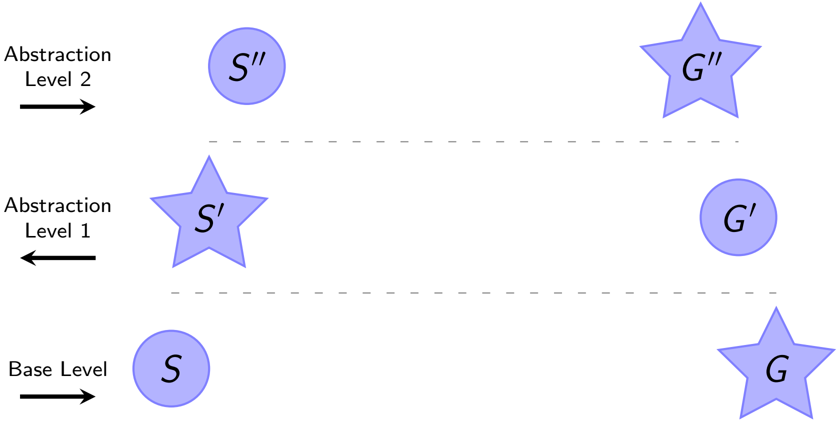
Hierarchical Heuristic Search Benefits

Advantages when solving one or a few instances:

- Abstract space visited lazily
  - No expensive preprocessing!
- Only generates cache entries that are required
- Likely faster & smaller than full PDB
- Natural to use instance-specific abstractions
  - Likely better than domain-specific abstractions

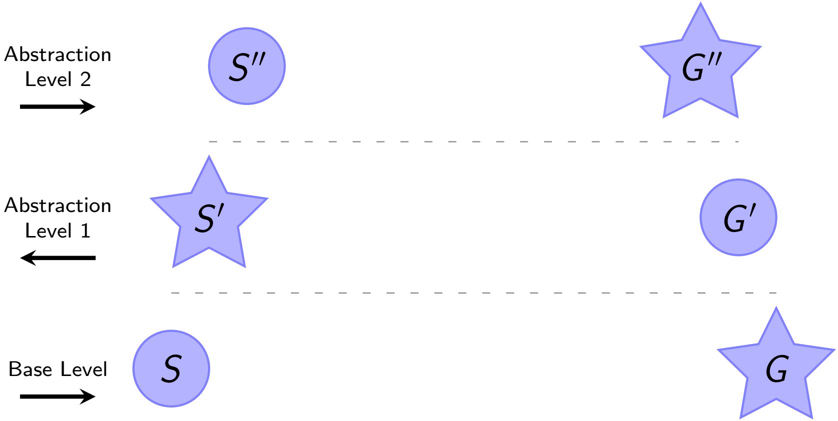
Outline

The Switchback Algorithm



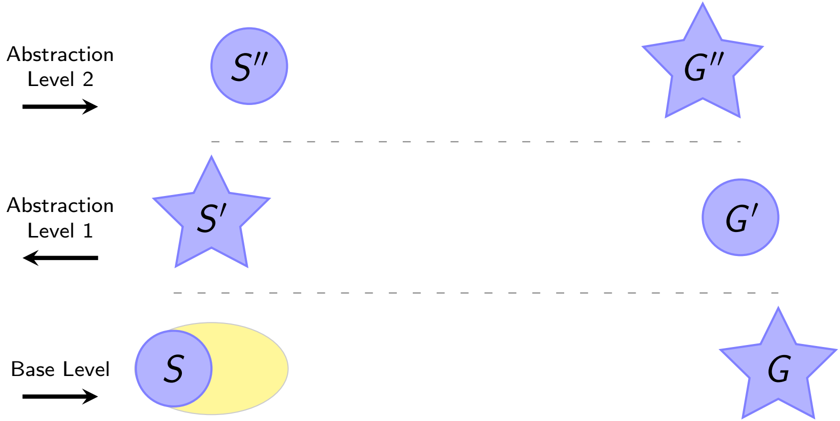
Switchback uses a hierarchy of abstractions.

The Switchback Algorithm



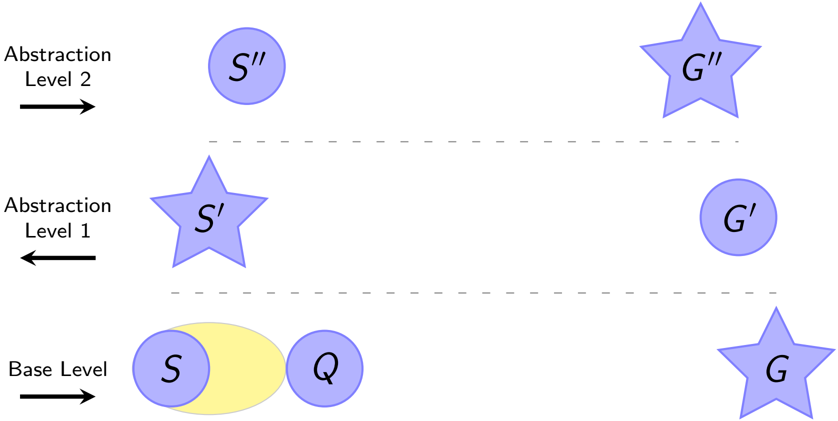
Objective: find the cheapest path from S to G.

The Switchback Algorithm



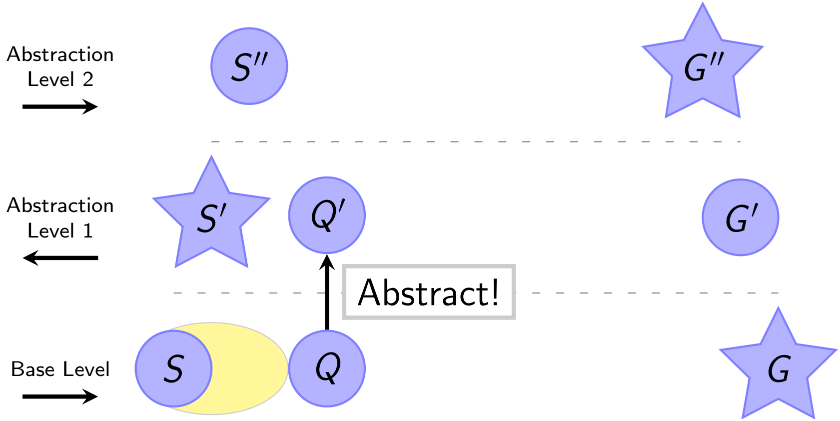
The yellow area represents generated states.

The Switchback Algorithm



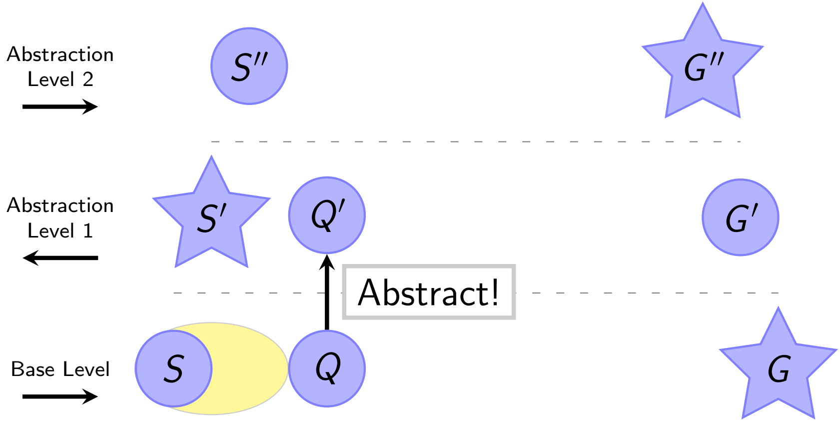
To generate Q, we need to know $h(Q)$.

The Switchback Algorithm



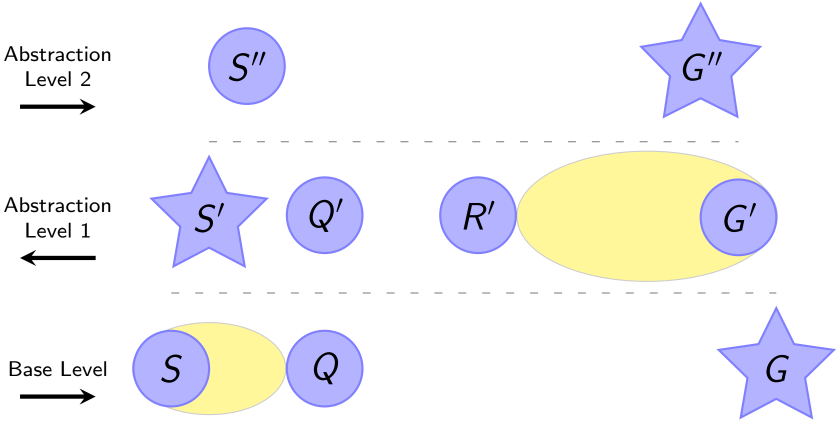
To find $h(Q)$: search backward at level 1.

The Switchback Algorithm



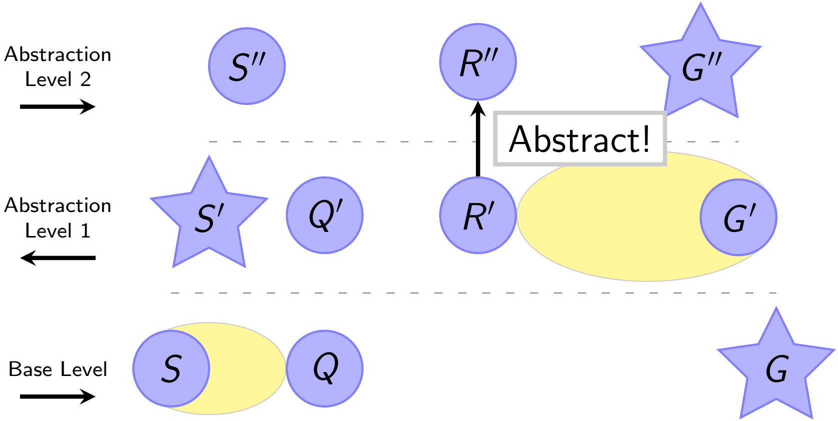
New objective: find cheapest path from G' to Q'.

The Switchback Algorithm



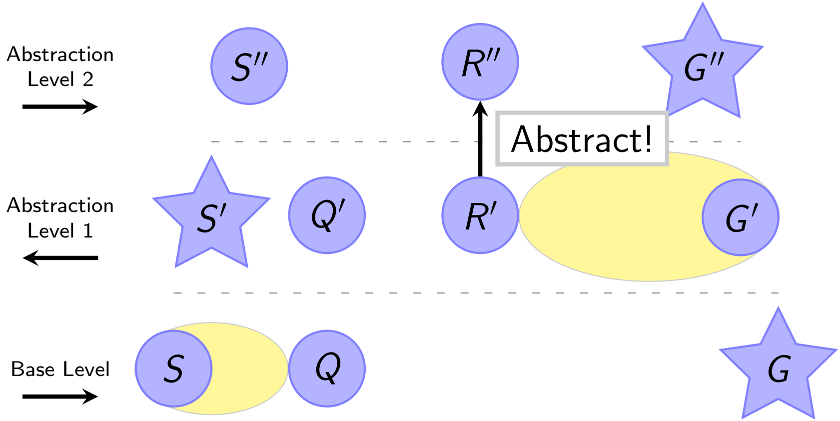
To generate R', we need to know $h(R')$.

The Switchback Algorithm



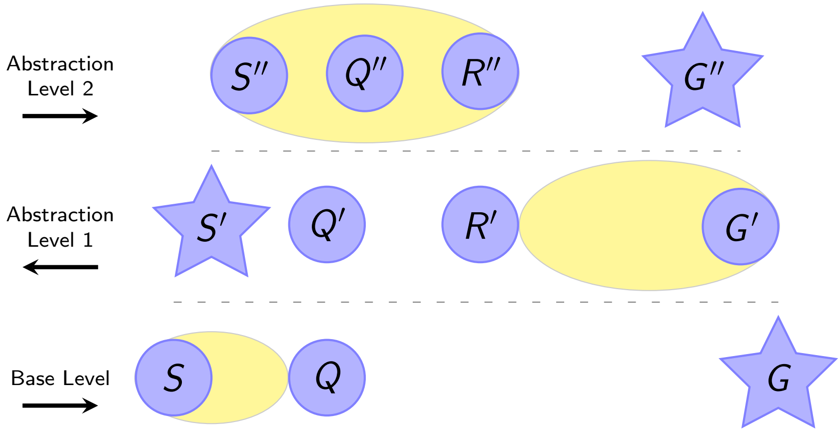
To find $h(R')$: search forward at level 2.

The Switchback Algorithm



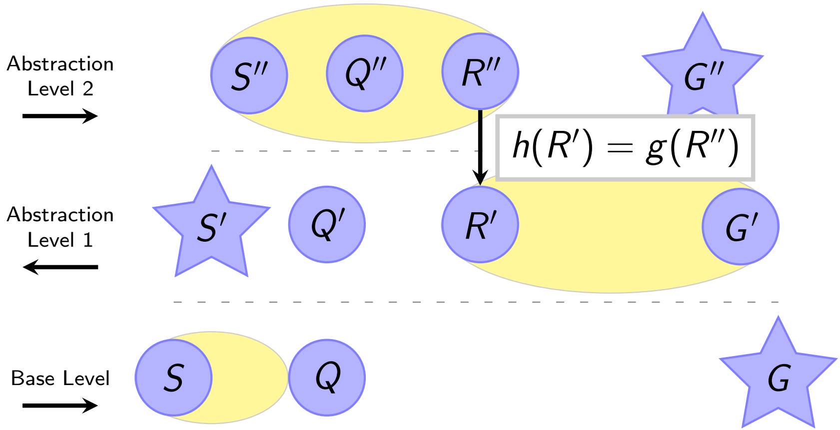
New objective: find cheapest path from S'' to R''.

The Switchback Algorithm



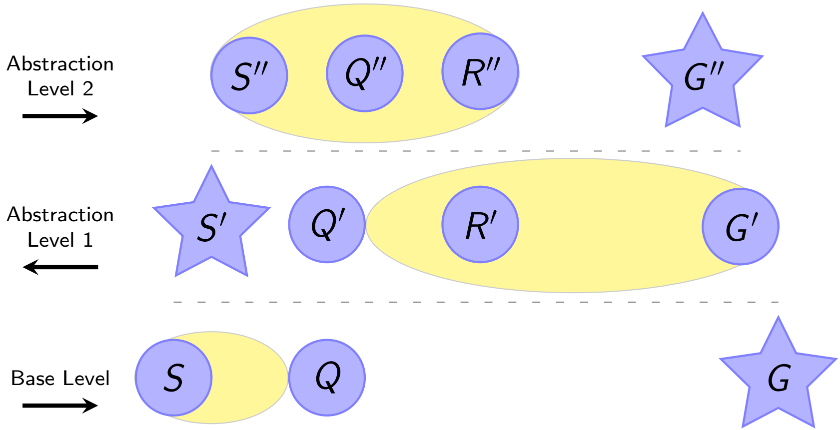
Use $h(n) = 0$ at the top level. Small search space!

The Switchback Algorithm



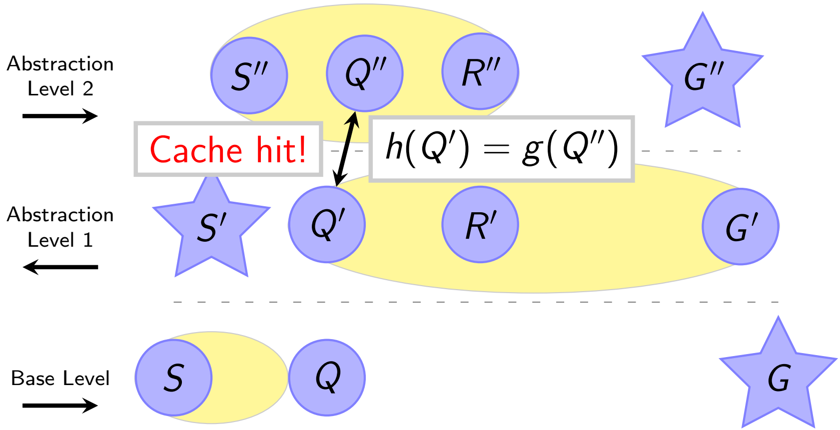
Use cost to R'' at level 2 as $h(R')$.

The Switchback Algorithm



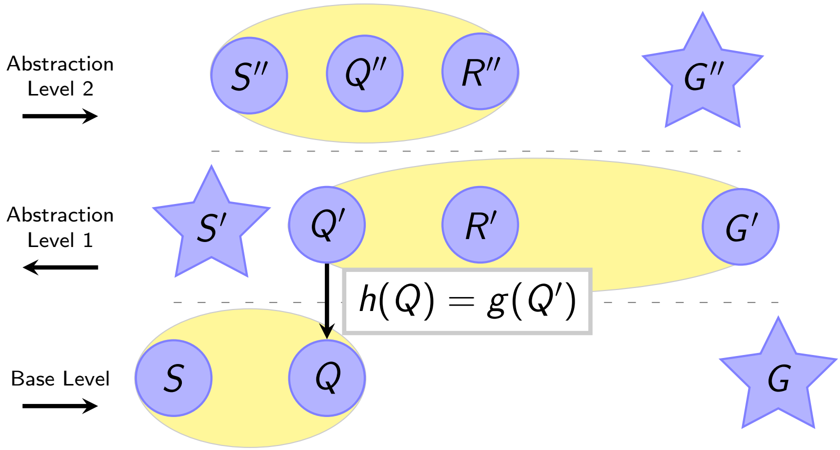
To generate Q', we need to know $h(Q')$.

The Switchback Algorithm



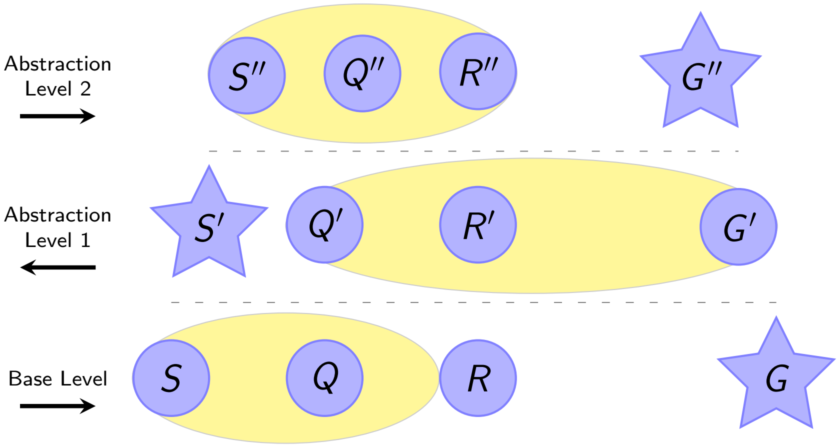
To find $h(Q')$: use cached cost to Q''.

The Switchback Algorithm



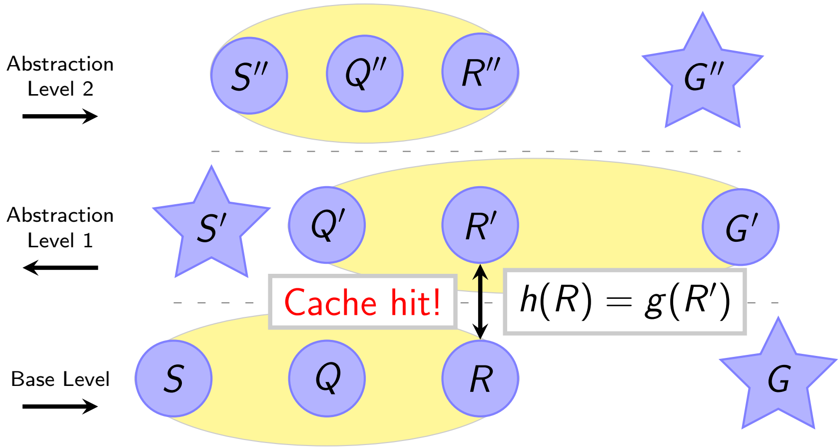
Use cost to Q' at level 1 as $h(Q)$.

The Switchback Algorithm



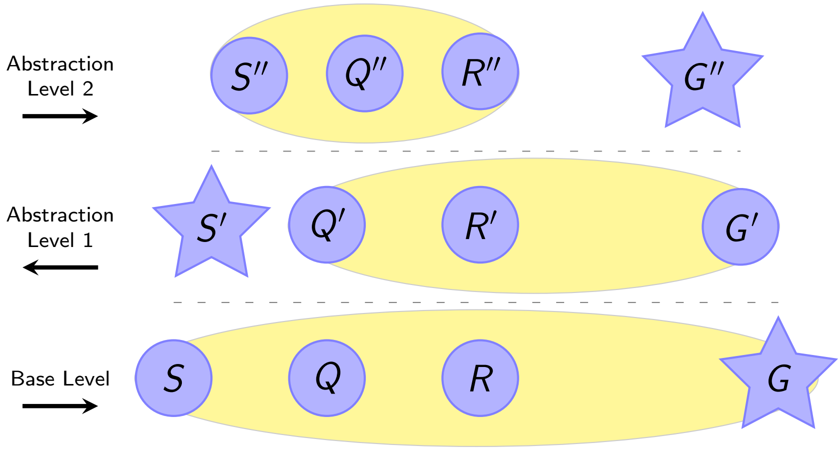
To generate R, we need to know $h(R)$.

The Switchback Algorithm



To find $h(R)$: use cached cost to R'.

The Switchback Algorithm



Search proceeds in this manner until finished.

Properties of Switchback

- Complete: terminates with solution if one exists
- Admissible: finds optimal solutions
- Efficient: expands states at most once
- See paper for details!

Outline

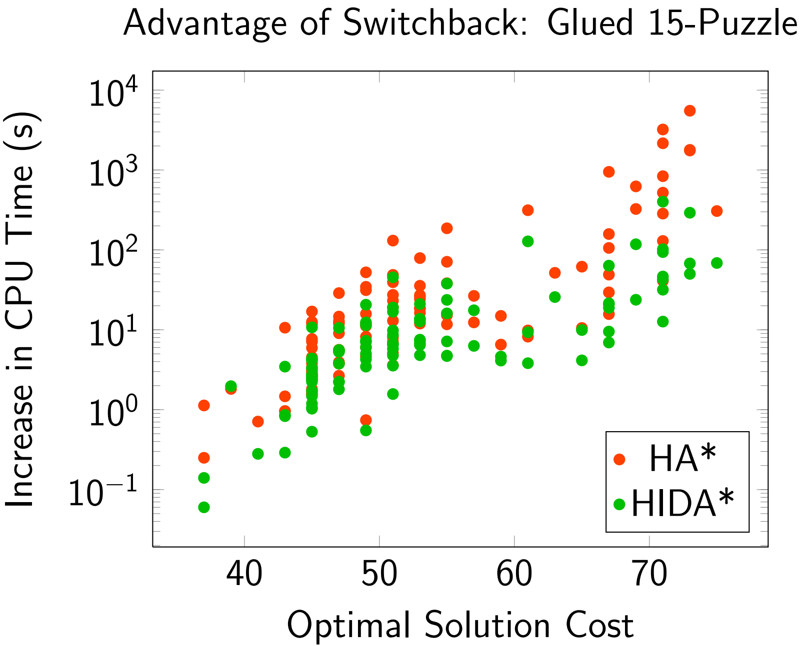
Experimental Setup

- HA*, HIDA*, and Switchback
- Several semi-standard domains
- Used custom abstraction hierarchies given by Holte et al. (2005)
- C++ implementation
- 47 GB memory limit, no time limit

Experimental Results: Glued 15-Puzzle

Averages from 100 instances

	CPU Time (s)		Nodes Gen. (M)	
Algorithm	Mean	Max	Mean	Max
Switchback	5	70	6	77
HA*	137	3286	12	99
HIDA*	21	172	20	132


Advantage of Switchback: Glued 15-Puzzle
Increase in CPU Time (s)
10^4
10^3
10^2
10^1
10^0
10^{-1}
40
50
60
70
Optimal Solution Cost
HA*
HIDA*

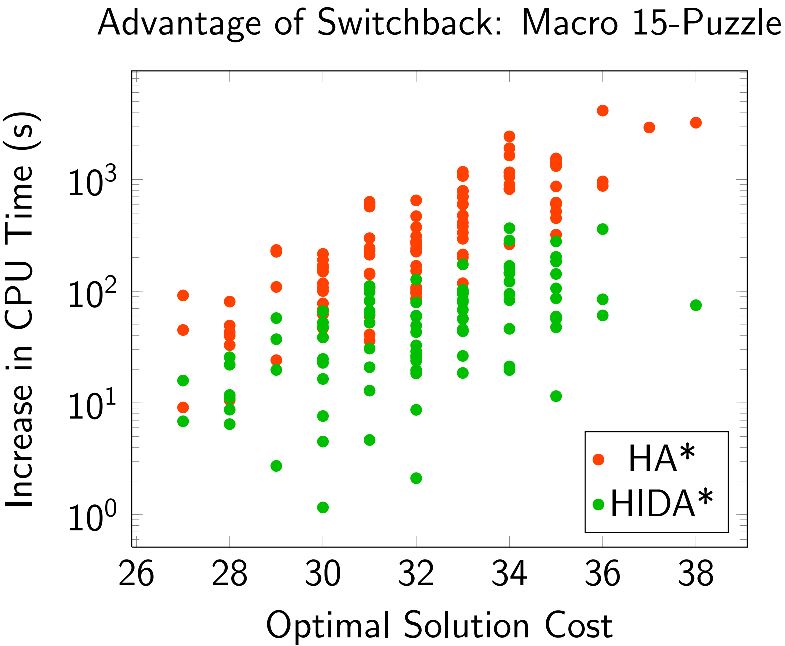
The Macro Tiles Puzzle

- Sliding tile puzzle variant
- Multiple tiles can be moved in one step
- Manhattan distance inadmissible
- Shallower solutions
- Higher branching factor
- More transpositions in state graph

Experimental Results: Macro 15-Puzzle

Averages from 100 instances

	CPU Time (s)		Nodes Gen. (M)	
Algorithm	Mean	Max	Mean	Max
Switchback	161	1127	182	948
HA*	708	4647	236	1074
HIDA*	223	1202	350	1746



Advantage of Switchback: Macro 15-Puzzle
Increase in CPU Time (s)
10^0
10^1
10^2
10^3
26
28
30
32
34
36
38
Optimal Solution Cost
HA*
HIDA*

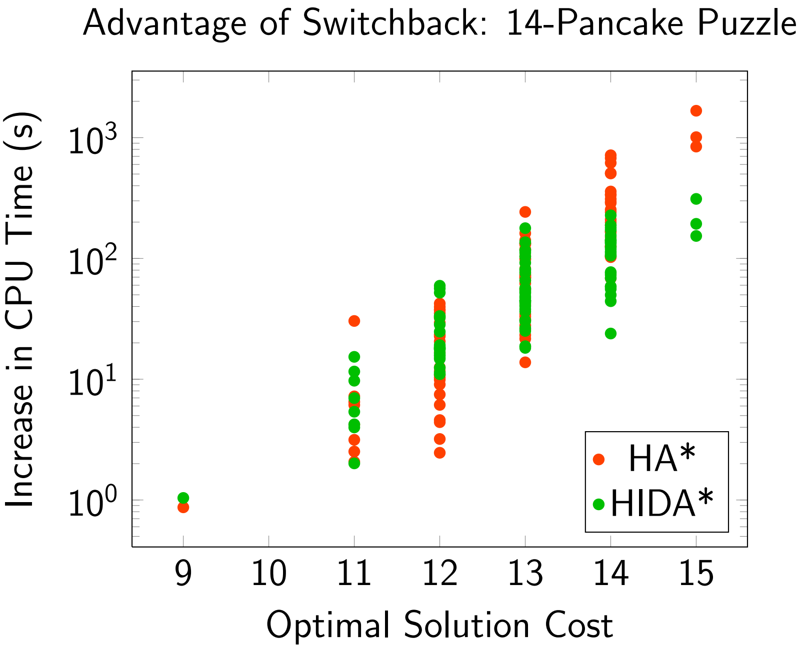
The Pancake Puzzle

- $\boxed{4 \mid 3 \mid 2 \mid 1 \mid 5} \rightarrow \boxed{1 \mid 2 \mid 3 \mid 4 \mid 5}$
- Goal: arrange numbers in order
- Can flip prefixes of the sequence
- Shallow solutions
- High branching factor

Experimental Results: 14-Pancake Puzzle

Averages from 100 instances

	CPU Time (s)		Nodes Gen. (M)	
Algorithm	Mean	Max	Mean	Max
Switchback	22	114	29	123
HA*	168	1772	38	156
HIDA*	89	401	87	311



Advantage of Switchback: 14-Pancake Puzzle
Increase in CPU Time (s)
10^0
10^1
10^2
10^3
9
10
11
12
13
14
15
Optimal Solution Cost
HA*
HIDA*

Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted

Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted
- Switchback is a better hierarchical search
  - avoids abstract state re-expansion
  - simple caching scheme
  - easy to implement

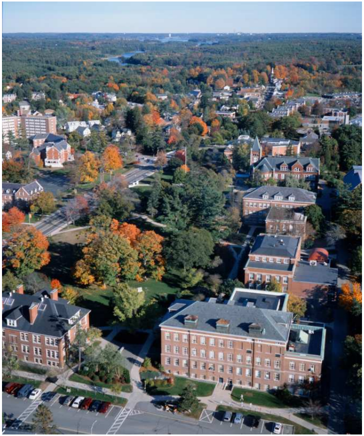
Summary

- Hierarchical search is an alternative to PDBs
  - Fills cache lazily
  - Can solve many problems in time to build PDB
  - Natural to use instance-specific abstractions
  - No cached heuristic values wasted
- Switchback is a better hierarchical search
  - avoids abstract state re-expansion
  - simple caching scheme
  - easy to implement
- Switchback should be widely applicable
  - when predecessor states easily computed
  - when good abstraction hierarchy available

The University of New Hampshire

Tell your students to apply to grad school in CS at UNH!

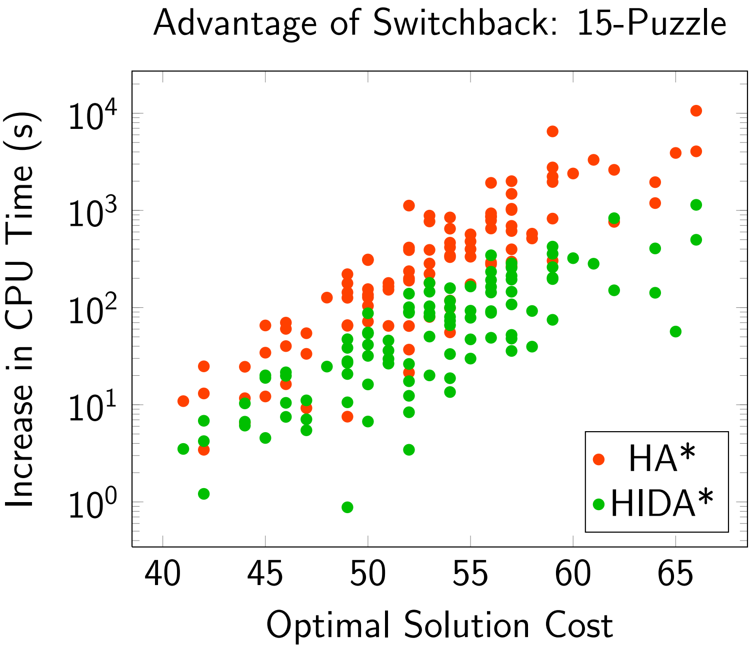
- friendly faculty
- funding
- individual attention
- beautiful campus
- low cost of living
- easy access to Boston, White Mountains
- strong in AI, infoviz, networking



Experimental Results: 15-Puzzle

Averages from 100 instances

	CPU Time (s)		Nodes Gen. (M)	
Algorithm	Mean	Max	Mean	Max
Switchback	83	1422	65	713
HA*	831	12034	110	1222
HIDA*	194	2563	197	2158


Advantage of Switchback: 15-Puzzle
Increase in CPU Time (s)
10^4
10^3
10^2
10^1
10^0
40
45
50
55
60
65
Optimal Solution Cost
HA*
HIDA*