Where Do Heuristics Come From ?

(Using Abstraction to Speed Up Search)

Robert C. Holte
Computing Science Department
University of Alberta

Complete set of slides and bibliography available at:
http://www.cs.ualberta.ca/~holte/Heuristics

Outline

Part 1: Introduction
- Refresher on heuristic search
- Using abstraction to create heuristics: basics
- Similar but different ideas
- Applications
- Historical notes

Part 2: Details

Part 3: Enhancements

Refresher on Heuristic Search

Search

- Find a shortest (least cost) path between two nodes (start, goal) in a given graph (state space).
- Typical "uninformed" algorithms (e.g. Dijkstra's algorithm) order their search based on **g(s)**, a state's distance from the start state.
- The state space is usually defined **implicitly**, by a set of operators.

Example: 8-puzzle

	1	2
3	4	5
6	7	8

181,440 states

Example Operator:
If the blank is in location (1,1) it can be exchanged with the tile in location (1,2).

Total number of operators: 24

For the general (NxN) sliding tile puzzle, the number of states grows exponentially with N but the number of operators grows quadratically.

Heuristic Search

- A heuristic, **h(s)**, estimates the distance from state s to the goal state.
- **f(s)**=g(s)+h(s)
  is the estimated cost of a path from start to goal through state s.
- Heuristic search algorithms use f(s) to prune and guide their search.

Manhattan Distance Heuristic

For a sliding-tile puzzle, Manhattan Distance looks at each tile individually, counts how many moves it is away from its goal position, and adds up these numbers.

	1
2	3

goal

3	
1	2

state s

MD(s) = 2 + 1 + 2 = 5

Heuristic Properties

- h(s) is admissible:
  - if it never overestimates distance to goal
  - guarantees certain algorithms will find a least-cost path from start to goal
- h(s) is monotone (consistent)
  - if f(s)=g(s)+h(s) never decreases along a path.
  - guarantees A* will never re-open a closed state

Heuristic Search Algorithms

Guaranteed to find optimal solutions*

A*, Breadth-first Heuristic search, IDA*, “C”, Branch and Bound, Limited Discrepancy Search, RBFS, SMA*, etc.

Might not find optimal solutions

Weighted A*, beam search, BULB, RTA*, etc.

* if h(s) is admissible

Terminology

- Expanding a state means generating its successors (children).
- A state is open if it has been generated but not expanded.
- A state is closed if it has been expanded.
- Open list = data structure holding all currently open states.
- Closed list = data structure indicating which states are closed.
- BFS = Breadth-first search

Using Abstraction to Create Heuristics – basics –

The Big Idea

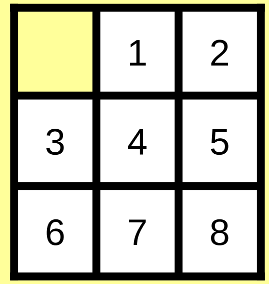
Create a simplified version of your problem.

Use the exact distances in the simplified version as heuristic estimates in the original.

Heuristics Defined by Abstraction

- An **abstraction** of state space S is any state space φ(S) such that:
  - for every state $s \in S$ there is a corresponding state $\varphi(s) \in \varphi(S)$.
  - $\text{distance}(\varphi(s_1), \varphi(s_2)) \le \text{distance}(s_1, s_2)$.
- Exact distances in φ(S) are admissible and consistent heuristics for searching in S.

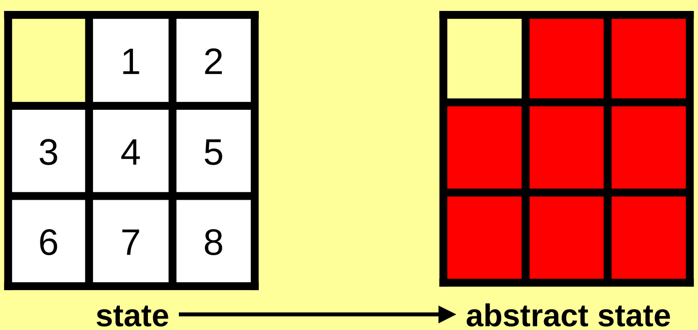
Example: 8-puzzle



181,440 states

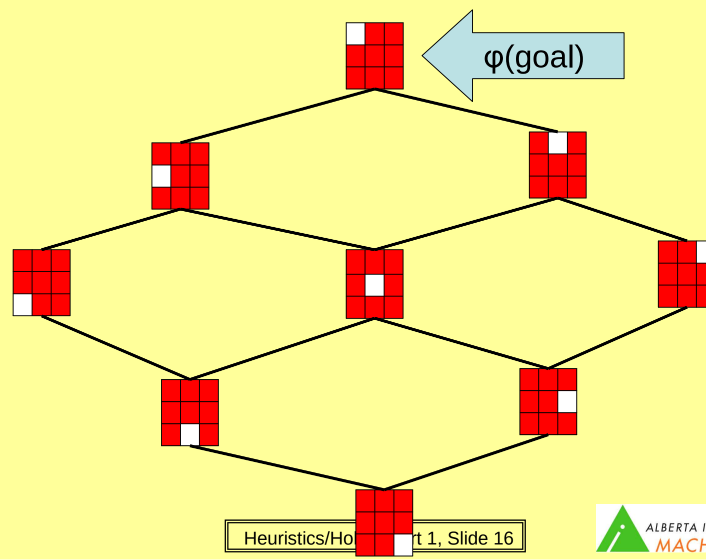
Domain = blank 1 2 3 4 5 6 7 8

Domain abstraction



Domain = blank 1 2 3 4 5 6 7 8
Abstract = blank ■ ■ ■ ■ ■ ■ ■ ■

Abstract State Space

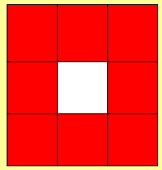


Calculating h(s)

Given a state, s

8	1	4
3		5
6	7	2

Compute the corresponding abstract state, φ(s)



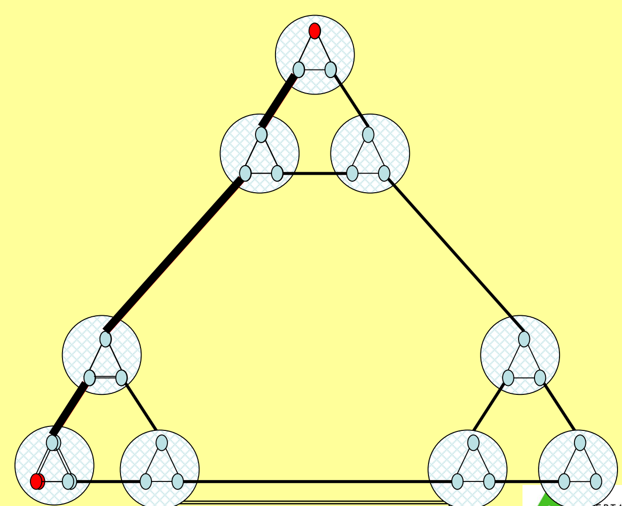
h(s) = distance(φ(s),φ(goal)) = **2**

Similar but Different Ideas

Abstract Path Refinement

- Use the abstract solution path as a skeleton
- Construct the final solution path by “fleshing out details” and filling in gaps
- Very fast, but solutions not necesarily optimal
- ABSTRIPS (Sacerdoti, 1974)
- ALPINE (Knoblock, 1994)
- Potential problem: what if the abstract solution cannot be refined ?
  (Bacchus & Yang, 1994)

Refinement Example



HOG Simulation Framework

- HOG (Hierarchical Open Graph)
  - C++ Framework for abstraction & refinement
  - Test-bed & visualization for abstraction in pathfinding
  - Two papers at AAAI'05 using HOG:
    - *Partial Pathfinding Using Map Abstraction and Refinement*
    - *Speeding Up the Convergence of Learning Real-time Search via Abstraction*
  - http://www.cs.ualberta.ca/~nathanst/hog.html

Learning Heuristics

- Use past problem-solving experience to invent or improve the heuristic.
- Example: the value function in reinforcement learning can be viewed as a heuristic function in settings where there is a set of goal locations giving reward and actions have costs.
- Often called speedup learning.

Lookahead Search

- To evaluate a node in game-tree search, it is common to search forward from the node to a certain depth, compute the static evaluation function at the lookahead frontier, and backup those values.
- Can also be done in single-agent search.
- Korf (1990)
- Bulitko et al. (2003, 2005)

Metalevel Reasoning

- "Reasoning at the meta-level concerns either choosing the right strategy from among alternatives or constructing a strategy by assembling a sequence of methods that together can accomplish a desired state."
  (Cox & Ram, 1999)
- 1960 (Newell, Shaw & Simon): GPS searched at the metalevel for a control strategy for GPS to use in searching at the base level.
- 1993 (Minton): Multi-Tac searches the space of combinations of CSP heuristics to find the optimal combination for the given problem and instance distribution.

Applications

Puzzles

- Rubik's Cube (Korf, 1997)
  - 10^{19} states
  - First random problems ever solved optimally by a general-purpose search algorithm
  - Hardest took 17 CPU-days
  - Best known MD-like heuristic would have taken a CPU-century
- 15-puzzle
  - 10^{13} states
  - Average solution time 0.021 seconds, with only 36,000 nodes expanded

Parsing

- Klein & Manning (2003)
- Used A* to find the most probable parse of a sentence.
- A "state" is a partial parse, g(s) is the "cost" of the parsing completed in s, h(s) estimates the "cost" of completing the parse.
- The heuristic is defined by simplifying the grammar, and is precomputed and stored in a lookup table.
- Special purpose code was written to compute the heuristic.
- Eliminates 96% of the work done by exhaustive parsing.

Dynamic Programming – SSR

- State Space Relaxation = mapping a state space onto another state space of smaller cardinality.
- Christofides, Mingozzi, and Toth (1981)
- Abstraction: very general definition and several different examples of abstractions for TSP and routing problems.
- Implemented but not thoroughly tested.
- Noted that the effectiveness of this method depends on how the problem is formulated.
- Did not anticipate creating a hierarchy of abstractions.

Dynamic Programming – CFDP

- Coarse-to-Fine Dynamic Programming
- C. Raphael (2001)
- Works on continuous or discrete spaces.
- Abstraction = grouping together states.
- Multiple levels of abstraction.
- Somewhat resembles refinement, but guaranteed to find optimal solution.
- Application: finding optimal convex region boundaries in an image.
- Algorithm illustrated at the end of Part 3

Weighted Logic Programs

- Felzenszwalb & McAllester (unpublished)
- Generalizes the statistical parsing and dynamic programming methods to the problem of finding a least-cost derivation of a set of statements (the “goal”) given a set of weighted inference rules.
- Inference at multiple levels of abstraction is interleaved.
- Application: finding salient contours in an image.

QoS Network Routing

- Li, Harms & Holte (2005)
- Find a least-cost path from start to goal subject to resource constraints.
- Each edge in the network has a cost and consumes some amount of resources.
- There are separate h(s) functions for the cost and for each type of resource.
- $h_r(s)$ is defined as the minimum cost of reaching the goal from state s subject only to constraints on resource r.

Sequential Ordering Problem

- Hernadvolgyi (2003)
- S.O.P. is the Travelling Salesman Problem with:
  - Asymmetric costs
  - Precedence constraints (must visit city A before city B)

Co-operative Pathfinding

- Silver (2005)
- Many agents, each trying to get from its current position to its goal position.
- Co-operative = agents want each other to succeed and will plan paths accordingly.
- Need a very efficient algorithm (because in computer games very little CPU time is allocated to pathfinding).

Vertex Cover

- Felner, Korf & Hanan (2004)
- fastest known algorithm for finding the smallest subset of vertices that includes at least one endpoint for every edge in the given graph .

Multiple Sequence Alignment

- Korf & Zhang (2000)
- McNaughton, Lu, Schaeffer & Szafron (2002)
- Zhou & Hansen (AAAI, 2004)
- Sets of N sequences are optimally aligned according to a mismatch scoring matrix.
- The heuristic is to find optimal matches of disjoint subsets of size k<N and add their scores.

Building Macro-Tables

- Hernadvolgyi (2001)
- A macro-table is an ultra-efficient way of constructing suboptimal solutions to problems that can be decomposed into a sequence of subgoals.
- For the j^{th} subgoal, and every possible state that satisfies subgoals 1…(j-1), the macro-table has an entry – a sequence of operators that maps the state to a state satisfying subgoals 1…j.
- Solutions are built by concatenating entries from the macro-table.
- Constructing the table is the challenge. Each entry is found by search. Heuristics are needed to find optimal entries in reasonable time.

Planning

- Edelkamp, 2001
- Bonet & Geffner, 2001
- Haslum & Geffner, 2000
- Abstraction is computed automatically given a declarative state space definition.
- Has been used successfully with a variety of different abstraction methods and search techniques. Some guarantee optimal solutions, many do not.

Constrained Optimization

- Kask & Dechter (2001)
- Mini-bucket elimination (MBE) provides an optimistic bound on solution cost, and therefore can be used to compute an admissible heuristic for A*, branch-and-bound, etc.
- MBE relaxes constraints. The objective function $\min_{\{a,b,c\}}\{f(a,b)+g(b,c)\}$ is relaxed to $\min_{\{a,b\}}\{f(a,b)\} + \min_{\{b,c\}}\{g(b,c)\}$, in effect dropping the constraint that the two values of b be equal.
- Applications include max-CSP and calculating the most probable explanation of observations in a Bayesian network.

Historical Notes

Prehistory: Two Key Ideas

<u>Using Lower Bounds to Prune Search</u>

1958: branch-and-bound

1966 (Doran & Michie): Graph Traverser, first use of estimated distance-to-goal to guide state space search.

1968 (Hart, Nilsson, Raphael): A*

<u>Using Abstraction to Guide Search</u>

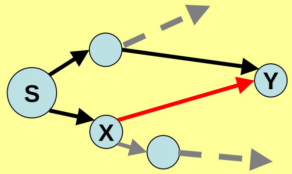
1963 (Minsky): abstraction=simplified problem + refinement

1974 (Sacerdoti): ABSTRIPS

Somalvico & colleagues (1976-79)

- Brought together the two key ideas.
- Proposed mechanically generating an abstract space by dropping preconditions.
- Proved this would produce admissible, monotone heuristics.
- Envisaged a hierarchy of abstract levels, with search at one level guided by a heuristic defined by distances at the level above.

Edge Supergraph



- Relaxing preconditions introduces additional edges between states and might add new states (by making a state reachable that is not reachable with the original preconditions).
- e.g. there is no edge from X to Y because of a precondition. If it is relaxed, there is an edge.

Gaschnig (1979)

- Proposed that the cost of solutions in space S could be estimated by the exact cost of solutions in auxiliary space T.
- Estimates are admissible if T is an edge supergraph of S.
- Observes: “If T is solved by searching this could consume more time than solving in S directly with breadth-first search.”
  – T should be supplied with an efficient solver

Valtorta (1980,1984)

- Proved that Gaschnig was right!
- Theorem: If T is an edge supergraph of S, and distances in T are computed by BFS, and A* with distances in T as its heuristic is used to solve problem P, then for any $s \in S$ that is necessarily expanded if BFS is used to solve P, either:
  – s is expanded by A* in S, or
  – s is expanded by BFS in T

Pearl (1984)

- Famous book, ***Heuristics***
- Popularized the idea that heuristics could very often be defined as exact costs to "relaxed" versions of a problem.
- To be efficiently computable, the heuristics should be semi-decomposable.
- Proposed searching through the space of relaxations for semi-decomposable ones.

Mostow & Prieditis (1989)

- ABSOLVER, implemented the idea of searching through the space of abstractions AND speed-up transformations.
- Reiterated that computing a heuristic by search at the abstract level is generally ineffective.
- Had a library with a variety of abstractions and speedups, not just "relax" and "factor".
- First successful automatic system for generating effective heuristics.
- Emphasized that success depends on having the right problem formulation to start with.

Mostow & Prieditis cont'd

- When a good abstraction is found, ABSOLVER calls itself recursively to create a hierarchy of abstractions, in order to speedup the computation of the heuristic.

Added in 1993 (Prieditis):

To make a heuristic "effective" precompute all the heuristic values before base-level search begins and store them in a hash table (today called a "pattern database").

Hansson, Mayer, Valtorta (1992)

- Generalized Valtorta's theorem to show that a hierarchy of abstractions created by relaxing preconditions was no use.
- Pseudocode for Hierarchical A*.

Using Memory to Speed Up Search

- 1985 (Korf): IDA*
- 1989 (Chakrabarti et al.): MA*
- 1992 (Russell): IE, SMA*
- 1994 (Dillenburg & Nelson): Perimeter Search
- 1994 (Reinefeld & Marsland): Enhanced IDA*
- 1994 (Ghosh, Mahanti & Nau): ITS

Culberson & Schaeffer (1996)

- 1994: technical report with full algorithm and results for pattern databases (PDB)
- 1996: first published account of PDBs
- Impressive results: 1000x faster than Manhattan Distance on the 15-puzzle.
- Several good ideas:
  - A general and effective type of abstraction
  - Efficiently precomputing and storing all the abstract distances
  - Exploiting problem symmetry
  - "Dovetailing" two PDBs

Holte (1996)

- 1994: published the Hierarchical A* idea.
- 1996: published working HA* algorithm, generalized Valtorta's Theorem to all kinds of abstractions, and showed (theoretically and experimentally) that speedup was possible with Hierarchical Heuristic Search if homomorphic abstractions are used.

Generalized Valtorta's Theorem

- If $\varphi(S)$ is any abstraction of S, for any $s \in S$ that is <u>necessarily expanded</u> if BFS is used to solve problem P, if A* is used to solve P using distances in $\varphi(S)$ computed by BFS as its heuristic, then either:
  - s is expanded by A* in S, or
  - $\varphi(s)$ is expanded by BFS in $\varphi(S)$