11. Evaluation

Jonathan Schaeffer
jonathan@cs.ualberta.ca
www.cs.ualberta.ca/~jonathan

Heuristic Evaluation Function

- Most of the magic in a single-agent searcher is in the evaluation function
- To obtain an optimal answer, we need an admissible lower bound
- Search tree size is strongly tied to the quality of the evaluation function
  - Unlike Alpha-beta where the evaluation function influenced the quality of the answer, but not really the size of the search

Issues

- How do we obtain an admissible evaluation function?
- How do we improve the quality of the evaluation function?
- What happens with a non-admissible heuristic?

Admissible Evaluations

- Consider a *relaxed* version of the application
- Eliminate a rule to simplify the calculation of the heuristic distance
- An exact solution to a relaxed problem is usually a good heuristic for the original problem

Example

- Consider Manhattan Distance for path-finding
- Original problem
  - Move man to goal subject to obstacles
- Relaxed problem
  - Move man to goal assuming no obstacles

Methodology

- Define the problem formally
- Remove one of the restrictions
- Evaluate whether the resulting problem is “easy” to evaluate and whether the results are worthwhile
- Could be automated, although this is an ongoing research topic

Multiple Heuristics

- There may be multiple good heuristics, each of which performs better in different circumstances
- Given N admissible heuristics, could compose a new, more powerful heuristic:

  $H = MAX(h_1, h_2, h_3, \ldots h_N)$

Heuristic Evaluation

- Right now, best way is to do this by hand; automated techniques are still in their infancy
- Apply application-dependent knowledge to decide on a heuristic(s)

Pattern Database [1]

- Endgame databases are a perfect lower bound for a (small) subset of positions
- Pattern databases computer lower bounds for subsets (patterns) of a state
- Using extra data, can improve the lower bound estimate

Pattern Databases

- Define a subset of the state
- Enumerate all possibilities for that subset and pre-compute optimal distances to solving that relaxed problem
- The larger the subset the more effective the heuristic

Using a Pattern Database

7	12	15	
			3
	11	13	
B		14	

B			3
			7
			11
12	13	14	15

Pre-compute the minimum number of moves to achieve a subset of the goal state.

Using a Pattern Database

- Many real-world problems have symmetries that can be exploited
- 15-puzzle symmetry
  - reflect horizontally and vertically
  - reflect along all four axis
  - use the maximum of all lower bounds

15-puzzle: H0

- Simplest heuristic evaluation function
- Value = 0 if a goal node
- Value = 1 if not a goal node

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

H = 1

15-puzzle: H1

- Count the number of misplaced tiles
- Assumes cost of placing a misplaced tile is 1

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

H = 12

15-puzzle: H2

- Manhattan Distance
- Count number of horizontal and vertical moves to place each tile

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

H = 28

Search = 540,860 nodes

15-puzzle: H3

- Add in linear conflicts
- Two tiles in a row/column that have to swap positions (3 and 7)

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

H = 30

15-puzzle: H4

- Pattern databases
- Use the pattern database shown earlier

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

H = 38

Search (F) = 5,303 nodes
Search(C) = 2,367 nodes

Experiments (1)

- Standard set of 100 test positions
  - Korf, 1985
- A* quickly runs out of memory (512 MB)
  - can only solve the “easy” problems
- Must use iterative deepening to reduce storage needs

Experiments (2)

- IDA*
  - linear storage in search depth (0 MB)
- Transposition table
  - 2^{18} entries, 20 bytes per entry (5 MB)
- Endgame database
  - all positions <= 22 moves of the goal (120 MB)
- Pattern database
  - all subsets of 8 tiles (500 MB)

Experiments (3)

IDA*	36,302,808,031	100.00
+ TT	13,662,973,000	37.64
+ DB	19,419,742,608	53.49
+ TT + DB	8,869,627,254	24.43
+ PDB	34,987,894	0.10
+ TT+DB+PDB	21,261,747	0.06

1707-fold improvement!

Lessons Learned

- Eliminate unnecessary work
- Any improvements to a state's evaluation will pay enormous benefits in the search
  - space/time tradeoffs
  - e.g. linear conflicts: a subset of pattern databases but without the storage and run-time costs

Space/Time

- Korf has a hypothesis that there is a linear relationship between execution time and storage used
  - Extreme cases: 0 storage and complete storage
  - In between, roughly doubling your storage can be used to reduce the tree by roughly a factor of two

Non-admissible Heuristics

- Admissible heuristics guarantee optimality
- Non-admissible heuristics are OK as long as you do not mind the possibility of non-optimal solutions

WIDA*

- Multiple the *h* value by a small constant > 1.0
- This has the effect of concentrating search on paths where the *h* value is small
- For many applications, this results in (near)-optimal solutions, but with a much improved execution time

References

[1] J. Culberson and J. Schaeffer. "Pattern Databases", *Computational Intelligence*, vol. 14, no. 4, pp. 318-334, 1998.