Heuristic Graph Search: Another Reason AI is Cool

Nick Deibel

Quiz Section – 5/31/02

A Dijkstra-Like Scenario

Your company owns a delivery truck that will have to make many trips in a day to various warehouses, always starting at warehouse A and ending at warehouse B. However, it will never repeat visit the same warehouse twice in one day. Because of demand, you cannot waste time refueling the truck until it reaches B. The truck can travel at most K miles on a single tank of gas. You are given a graph where the nodes represent the warehouses and the directed edges represent the highways connecting the two warehouses. Each edge is weighted according to the length of that highway.

As you want your drivers to avoid taking paths of longer than K miles, design an algorithm that tells you if there exists any simple path from A to B (no nodes/warehouses repeated) whose length is greater than K miles.

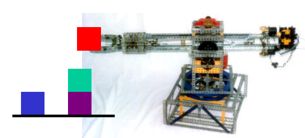
Whoops, That's ***REALLY*** Hard

- Longest Path is an NP-Complete problem.
- No known polynomial-time algorithm solves it.
  - A poly-time algorithm does exist for DAGS.
- This is a difficult problem for even small graphs.

Huge Graphs

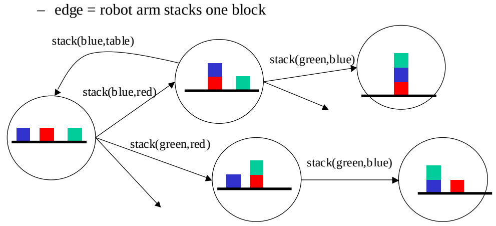
- Consider some really huge graphs…
  - All cities and towns in the World Atlas
  - All stars in the Galaxy
  - All ways 10 blocks can be stacked


Huh???



Implicitly Generated Graphs

- A huge graph may be implicitly specified by rules for generating it on-the-fly
- Blocks world:
  - vertex = relative positions of all blocks
  - edge = robot arm stacks one block



Blocks World

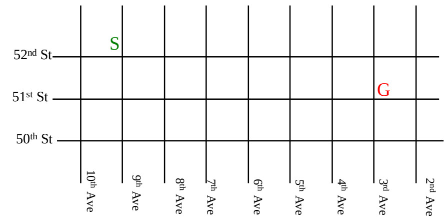
- Source = initial state of the blocks
- Goal = desired state of the blocks
- Path source to goal = sequence of actions (program) for robot arm!
- n blocks ≈ n^n vertices
- 10 blocks ≈ 10 billion vertices!

Problem: Branching Factor

- Cannot search such huge graphs exhaustively. Suppose we know the goal is only *d* steps away.
- Dijkstra's algorithm is basically breadth-first search (modified to handle arc weights)
- Breadth-first search (or for weighted graphs, Dijkstra's algorithm) – If out-degree of each node is 10, potentially visits 10^d vertices
  - 10 step plan = 10 billion vertices visited!

An Easier Case

- Suppose you live in Manhattan; what do you do?

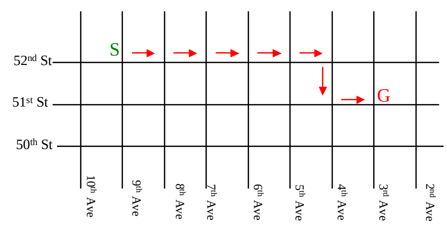


Best-First Search

- The *Manhattan distance* (Δ x+ Δ y) is an estimate of the distance to the goal
  - a heuristic value
- Best-First Search
  - Order nodes in priority to minimize estimated distance to the goal h(n)
- Compare: BFS / Dijkstra
  - Order nodes in priority to minimize distance from the start

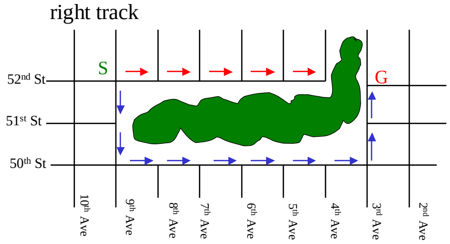
Best First in Action

- Suppose you live in Manhattan; what do you do?



Problem 1: Led Astray

- Eventually will expand vertex to get back on the right track

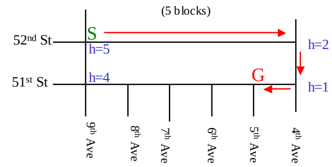


Problem 2: Optimality

- With Best-First Search, are you *guaranteed* a shortest path is found when
  - goal is first seen?
  - when goal is removed from priority queue (as with Dijkstra?)

Sub-Optimal Solution

- No! Goal is by definition at distance 0: will be removed from priority queue immediately, even if a shorter path exists!



Synergy?

- Dijkstra / Breadth First guaranteed to find *optimal* solution
- Best First often visits *far fewer* vertices, but may not provide optimal solution
  - *Can we get the best of both?*

Heuristics

- A rule of thumb, simplification, or educated guess.
- Reduces the search for solutions in large solution spaces
- Unlike algorithms, heuristics do not guarantee optimal, or even feasible, solutions.

A* ("A star")

- Order vertices in priority queue to minimize

(distance from start) + (estimated distance to goal)

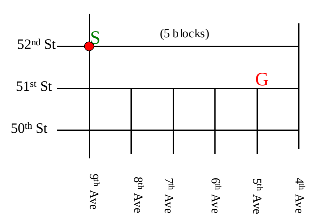
f(n) = g(n) + h(n)

f(n) = priority of a node
g(n) = true distance from start
h(n) = heuristic distance to goal

Optimality

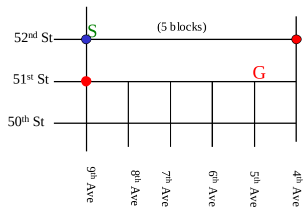
- Suppose the estimated distance (h) is *always* less than or equal to the true distance to the goal
  - heuristic is a *lower bound on true distance*
- Then: when the goal is removed from the priority queue, we are guaranteed to have found a shortest path!

Problem 2 Revisited



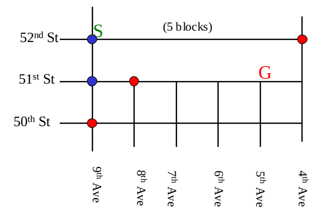
vertex	g(n)	h(n)	f(n)
52nd & 9th	0	5	5

Problem 2 Revisited



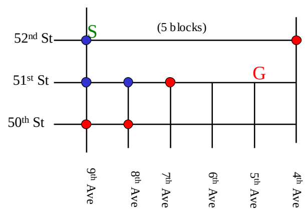
vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
51^{st} & 9^{th}	1	4	5

Problem 2 Revisited



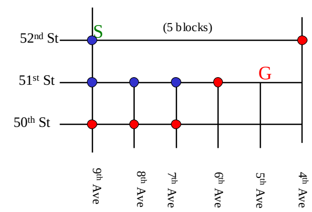
vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
51^{st} & 8^{th}	2	3	5
50^{th} & 9^{th}	2	5	7

Problem 2 Revisited



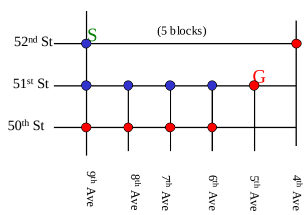
vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
51^{st} & 7^{th}	3	2	5
50^{th} & 9^{th}	2	5	7
50^{th} & 8^{th}	3	4	7

Problem 2 Revisited



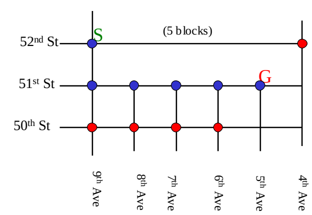
vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
51^{st} & 6^{th}	4	1	5
50^{th} & 9^{th}	2	5	7
50^{th} & 8^{th}	3	4	7
50^{th} & 7^{th}	4	3	7

Problem 2 Revisited



vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
51^{st} & 5^{th}	5	0	5
50^{th} & 9^{th}	2	5	7
50^{th} & 8^{th}	3	4	7
50^{th} & 7^{th}	4	3	7

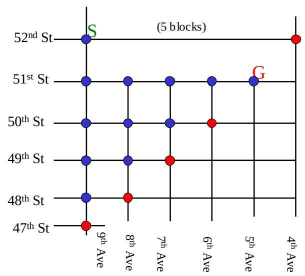
Problem 2 Revisited



vertex	g(n)	h(n)	f(n)
52^{nd} & 4^{th}	5	2	7
50^{th} & 9^{th}	2	5	7
50^{th} & 8^{th}	3	4	7
50^{th} & 7^{th}	4	3	7

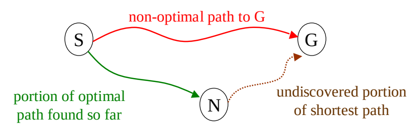
DONE!

What Would Dijkstra Have Done?



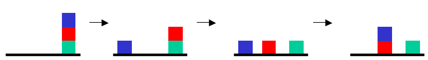
Proof of A* Optimality

- A* terminates when G is popped from the heap.
- Suppose G is popped but the path found isn't optimal:
  priority(G) > optimal path length c
- Let P be an optimal path from S to G, and let N be the last vertex on that path that has been *visited but not yet popped*.
  There must be such an N, otherwise the optimal path would have been found.
  priority(N) = g(N) + h(N) ≤ c
- So N should have popped before G can pop. Contradiction.



What About Those Blocks?

- "Distance to goal" is not always physical distance
- Blocks world:
  - distance = number of stacks to perform
  - heuristic lower bound = number of blocks out of place



\# out of place = 1,   true distance to goal = 3

Other Real-World Applications

- Routing finding – computer networks, airline route planning
- VLSI layout – cell layout and channel routing
- Production planning – "just in time" optimization
- Protein sequence alignment
- Many other "NP-Hard" problems
  - A class of problems for which no exact polynomial time algorithms exist – so heuristic search is the best we can hope for