Rook Jumping Mazes:
a Computer Science design project

Todd W. Neller

Maze © Adrian Fisher
Traditional Mazes

- Lines, hedges, corn rows, etc. delineate paths
- Find a path from start to goal

From “The Amazing Book of Mazes” by Adrian Fisher
Logic Mazes, Quick Mazes

- Smaller, more complex states and/or transitions
- Find a “path” (state sequence) from start state to goal state
- Small footprint, novel rules, fast accessibility and solving

From “The Amazing Book of Mazes” by Adrian Fisher
Basic form: square grid, start state (square), goal state, jump numbers for each non-goal state.

Jump number: Move *exactly* that many squares forward, backward, left, right. (*Not* diagonally.)

Objectives:
- Find a path from start to goal
- Find the shortest of these paths
Adrian Fisher “Rook Jumping Maze”

Robert Abbott “No U–Turn Number Maze” Forward, left, right but not back
The number of possible $5 \times 5$ rook jumping mazes configurations with a center goal: $4^{16} \times 3^8 > 2.8 \times 10^{13}$ (a lot)

The number of possible $n \times n$ mazes is bounded above by $(n-1)^{n^2}$

The number of good puzzle configurations is considerably less (many needles in a very large haystack)

Can’t generate and test all, but can search for a good one
First, I need a way to rate the maze relative (un)desirability
- e.g. penalize if goal not reachable from a state

Then, I need a method for looking around:
- Start with a random maze configuration
- Change a random position to a random different jump
- …but sometimes these changes are counterproductive
The Drunken Topographer

- Imagine an extremely hilly landscape with many hills and valleys high and low
- Goal: find lowest spot
- Means: airlift a drunk!
- Starts at random spot
- Staggers randomly
- More tired $\rightarrow$ rejects more uphill steps
Super–Drunks, Dead–Drunks, and Those In–Between

- The Super–Drunk never tires
  - Never rejects uphill steps
  - How well will the Super–Drunk search?
- The Dead–Drunk is absolutely tired
  - Always rejects uphill steps
  - How well will the Dead–Drunk search?
- Now imagine a drunk that starts in fine condition and very gradually tires.
What Blacksmiths Knew

- Quenching
  - Heated metal into a cold water barrel
  - Rapid cooling → brittle metal

- Annealing
  - Heated metal allowed to cool slowly
  - Slow cooling → strong metal
What Statistical Physicists Learned

- Large number of atoms in a random configuration $\rightarrow$ high energy state
- High temperature (energy input) $\rightarrow$ atoms reconfigure freely to higher or lower energy states
- Low temperature $\rightarrow$ atoms reconfigure less freely (usually to lower energy states)
- Metropolis algorithm (Metropolis et al., 1953)
Rook Jumping Maze Generation

- **State**: a total configuration of jumps
- **Energy**: rating of maze’s undesirability
- **Step**: select a random position and change to a different random jump
- The prime design challenge is to define a good energy function, scoring a maze’s undesirability.
- What are examples of undesirable characteristics?
Penalize Unreaching States

- We definitely want to have a solution, and we may want to have all states to have a path to the goal.
- Score: Add 1 per *unreaching* state, i.e. state with no path to goal.

```
[ 1]  1   1   1   2
4   1   2   3   2
1   2   2   2   2
3   3   2   1   1
2   4   3   2  GOAL
```

Distances to Goal:
```
4   5   4   3   2
3   4   3   4   2
4   5   2   4   1
3   2   2   2   1
5   6   4   5   0
```

score = getNumUnreaching();
Prioritizing Objectives

- First priority: no unreaching states
- Second priority: maximize minimum path length from start to goal
- Find the range of 2\textsuperscript{nd} priority ratings \( r \)
- Solution: Multiply each 1\textsuperscript{st} priority unit by \((r+1)\)
- Example:
  - Max path length less than rows times columns
  - Multiply number of unreaching states by rows times columns, and subtract minimum path length from start to goal
Reward Longer Solution Paths

- Score: Add rows*cols per unreachingle state, i.e. state with no path to goal. Subtract minimum path length from start to goal (if path exists).

\[
\begin{bmatrix}
3 & 3 & 4 & 4 & 3 \\
4 & 1 & 2 & 3 & 2 \\
3 & 1 & 1 & 1 & 2 \\
3 & 2 & 3 & 3 & 2 \\
3 & 4 & 1 & 3 & \text{GOAL}
\end{bmatrix}
\]

Distances to Goal:
18  7  10  17  8
14  5  12  15  13
3  4  3  2  1
19  6  11  18  12
15  8  9  16  0

\[
\text{score} = \text{rows} \times \text{cols} \times \text{getNumUnreaching}();
\]
\[
\text{if (getDistance(startState, goalState) != NONE)}
\]
\[
\text{score} -= \text{getDistance(startState, goalState)};
\]
Possible Considerations

- Should all states be reachable?
- Which structures are/aren’t enjoyable challenging?
- Is distance the right measure? Should number of choice points along the path be used instead?
- Etc. etc. etc. → lots of room for creativity!
- If you can define the measure, we can program the measure.
- Experiment and observe the change in mazes generated
Stochastic local search is a simple, powerful algorithm for finding good configurations in a vast space of configurations, if:

- One can identify a good “local” step, and
- One can characterize relative (un)desirability via an energy function.

It is in the energy function that art and science meet.

- Energy measure is often non-trivial and requires careful consideration and creativity

You can become an expert maze designer.

- Experiment, observe, introspect, express
Possible Projects

- Free iPhone app
- Publication of paper on maze design
- Public, walkable, student-generated jumping maze
- Daily maze on department website
- Showcase of student work at Celebration 2010
- Puzzle book
- ??? What would be fun for you? What would add most to your portfolio / resumé?
References


Traveling Salesman Problem

- Have to travel a circuit around \( n \) cities (\( n = 400 \))
- Different costs to travel between different cities (assume cost = distance)
- **State**: ordering of cities (> \( 8 \times 10^{865} \) orderings for 400 cities)
- **Energy**: cost of all travel
- **Step**: select a portion of the circuit and reverse the ordering
Sketch of Simulated Annealing

- Pick any starting state
- While gradually cooling the temperature:
  - Randomly change the state
  - Compare old and new energy
  - If new energy lower $\rightarrow$ accept new state
  - Otherwise accept new state with a probability computed from the energy change $\Delta E$ and temperature $T$ (probability $e^{-\Delta E/kT}$)