Assessing IDA*

- Q: What if we substitute A* for depth-first?
- Problem: too many iterations if there are many different $f$-values
- Alternatives: use more space
  - RBFS
  - SMA*
Beam Search

- Put a bound on the size of the fringe
- After inserting successors (in breadth-first order), sort by priority and truncate to fit bound
- Like shining a fixed-width “beam” into the search space
- Implications
  - Optimality?
  - Completeness?
  - Complexity?

Local Search

- Applicable when we don’t care about paths—just solution state.
- Avoid space problems entirely: maintain only one a finite number of solution candidates
  - Perhaps only one!
- Repeatedly tweak those candidates in the hopes of arriving at a solution.
  - How do we tweak the solutions?
Example: 8 Queens

Find an arrangement of queens such that no queen attacks another.

8 Queens Heuristic

$h = 1$ a local minimum $h = 17$

$h$: number of pairs of queens that attack each other
Local Beam Search

1. Randomly generate $k$ initial states
2. Generate successors for each of them
3. If any successor is a goal, then return it and exit
4. Otherwise put all successors into queue, and sort queue.
5. Remove all but the $k$ best nodes from the queue, and go to step 2

- How is this different than doing $k$ random restarts?
- Can also have the stochastic variation, where the $k$ nodes kept are chosen with some weighted probability based on heuristic value

Genetic Algorithm

- Parallel hill climbing
- Candidate successors generated by crossover and mutation
- Actual successors then selected based on fitness
GA Steps

- Initialize population of size \( N \)
- Repeat \( N \) times:
  - Randomly select two “parents” from population, with probability proportional to fitness
  - Construct “child” by crossing over parents
  - Apply mutation with small probability

Crossing Over

- Randomly select crossover point.
- Child is same as parent1 up to crossover point, parent2 after that.
Genetic Algorithm: Your turn!

- You’ll need a sheet of paper and a pencil
  - Write down four random numbers, $x_1$, $x_2$, $x_3$, and $x_4$.
  - Each number should be between $[1, 9]$.
  - Seriously. They need to be random!

- You are our initial population!

Genetic Algorithm: Fitness

- Compute your fitness:
  \[ f = |6x_1^2x_2^2 + 18x_1^2x_4 - 70x_1^2 - 21x_2^2x_3 - 63x_3x_4 + 245x_3 + 24x_2^2 + 72x_4 - 280| \]

- (In our case, small fitnesses are good.)
Genetic Algorithm: Reproduction

- Who has low fitnesses?

- Sexual reproduction (without mutation) by crossing \((x_1,x_2)\) with \((x_3,x_4)\)

\[
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
\end{array} \quad \begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
\end{array}
\rightarrow
\begin{array}{cccc}
  x_1 & x_2 & x_3 & x_4 \\
\end{array}
\]

parents

offspring

Changing the genetic representation

- Our fitness function can be factored like this:

\[
\left| (2x_1^2 - 7x_3 + 8)(3x_2^2 + 9x_4 - 35) \right|
\]

- What does this tell us about what our genetic representation should be?

\[
\begin{array}{cccc}
  x_1 & x_3 & x_2 & x_4 \\
\end{array} \quad \begin{array}{cccc}
  x_1 & x_3 & x_2 & x_4 \\
\end{array}
\rightarrow
\begin{array}{cccc}
  x_1 & x_3 & x_2 & x_4 \\
\end{array}
\]

parents

offspring
Adding Mutation

- We can randomly flip bits too...

Hill Climbing

- aka Gradient descent
- Requires heuristic $h$ measuring quality of soln
- Algorithm:
  - Find all incremental modifications of candidate soln
  - Pick best one
  - Repeat
Example: Map Labeling

Example: Map Labeling
Example: Map Labeling

Example: Map Labeling
3SAT Example

\[(P_1\lor\neg P_2\lor P_3)\land(P_1\lor\neg P_2\lor\neg P_4)\land(P_1\lor\neg P_3\lor\neg P_4)\land \]
\[\neg(P_1\lor P_2\lor\neg P_3)\land(P_2\lor\neg P_3\lor\neg P_4)\land(P_2\lor\neg P_3\lor\neg P_4)\land \]
\[\neg(P_1\lor\neg P_3\lor P_4)\land(P_1\lor\neg P_3\lor P_4)\land(\neg P_2\lor\neg P_3\lor P_4)\land \]
\[\neg(P_1\lor P_2\lor P_3)\]

Q: What’s a good fitness function?

GSAT

procedure GSAT(\phi)
    for i := 1 to Max-tries
        T := random truth assignment
        for j := 1 to Max-flips
            if T satisfies \phi then return T
            else Poss-flips := set of vars that increase satisfiability most
            V := a random element of Poss-flips
            T := T with V's truth assignment flipped
        end
    end
return “no satisfying assignment found”
Hill Climbing Terrain

- Local maxima
- Plateaux
- Ridges

Hill-Climbing: 2-d Ridge

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Stochastic Variations

- Stochastic hill climbing
  - Select among positive steps at random
  - Probability proportional to steepness
- Random restarts
  - Repeat hill climbing from randomly chosen initial state
  - Return best local maximum found

No clear answer on how often to restart from scratch versus trying to “repair” a current candidate that’s stuck or making slow progress.

Simulated Annealing

- Hill climbing, but take worse-appearing steps with some probability
  - Generate random neighbor
    - If it is an improvement, accept;
    - else accept with probability < 1
  - probability decreases exponentially with the “badness” of the move, temperature
- Annealing: Decrease temperature gradually

- Stochastic Gradient Descent is similar
  - Useful for optimization with many simultaneous “soft” constraints
  - Temperature decreases as $1/T$
    - Actually takes a long time for the temperature to get really small.
GA: Discussion

- Appealing analogy to natural selection with sexual reproduction

- Does it work?
  - Hard to characterize in general
  - Depends crucially on string rep’n of state
  - Intuition: GA maintains good “building blocks” in population
  - Not generally better than simpler stochastic local search methods

Assessing Local Search

- Key advantages
  - Very little memory
  - Can often find reasonable solutions in large or infinite (continuous) state spaces where other systematic approaches are unsuitable

- Usually incomplete and not optimal
Adversarial Search

- Next time...