Local Search

Toolbox so far

- Uninformed search
  - BFS, DFS, Iterative deepening DFS, Uniform cost search
- Heuristic search
  - A*

Review

- Search:
  - Use in environments that are static, discrete, 100% observable, deterministic.
- Things we care about:
  - Completeness, optimality, time/space complexity.

it's not the
DESTINATION
that matters,
it's the
JOURNEY

New Idea: **Local Search**

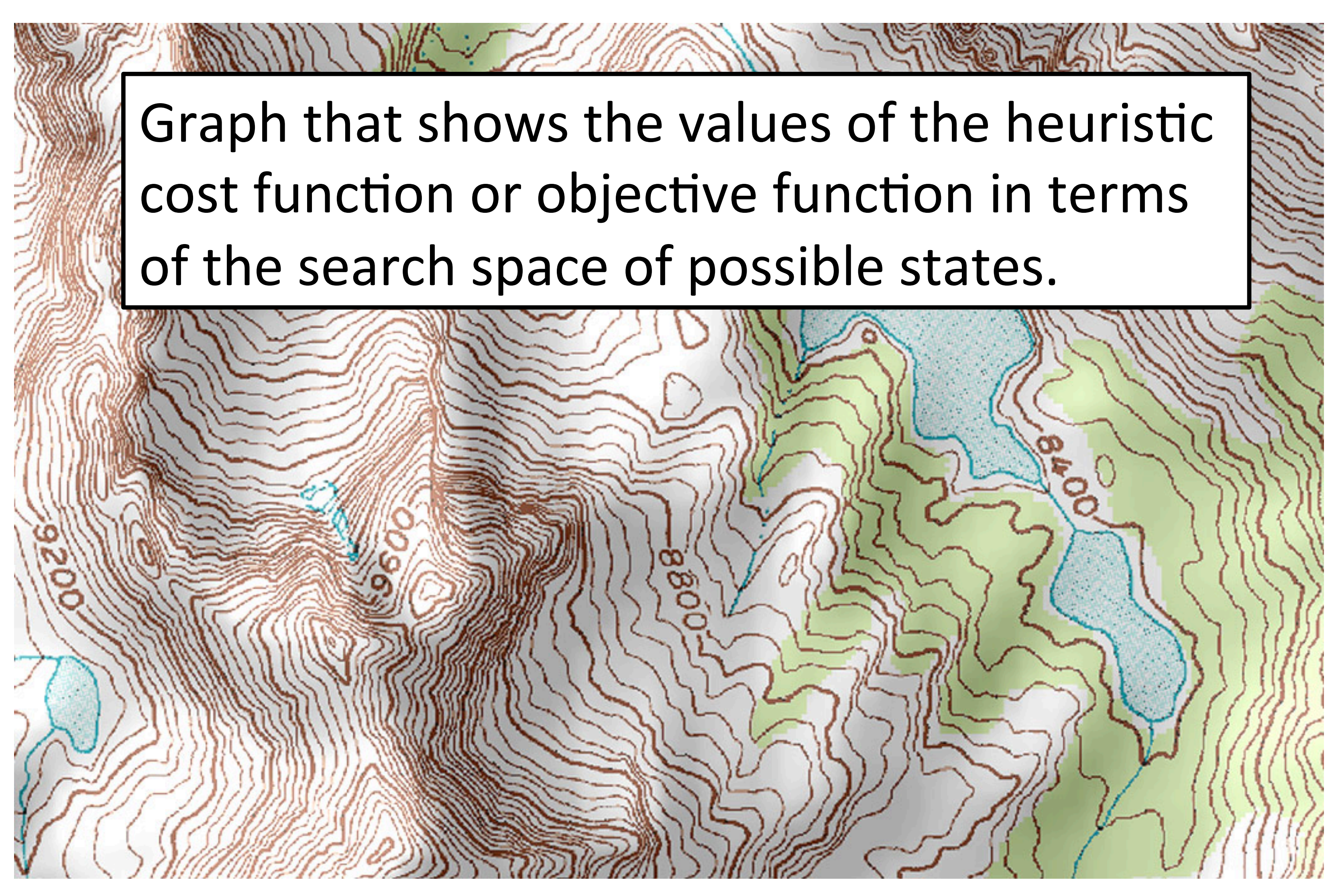
- Can be used when path from start state to goal doesn't matter (only the goal matters).
- Process is slightly different than "normal" search:
  - Nodes/states are always complete solutions to the problem, not partial solutions.
  - One ***current node*** is maintained that has the best solution at the moment.
  - ***Actions*** generate new nodes with new complete solutions.

Local Search

- Benefits:
  - Use very little memory, often constant.
  - Can search very large state spaces quickly.
- Useful in optimization problems.

State-space landscape

State-space landscape



Graph that shows the values of the heuristic cost function or objective function in terms of the search space of possible states.

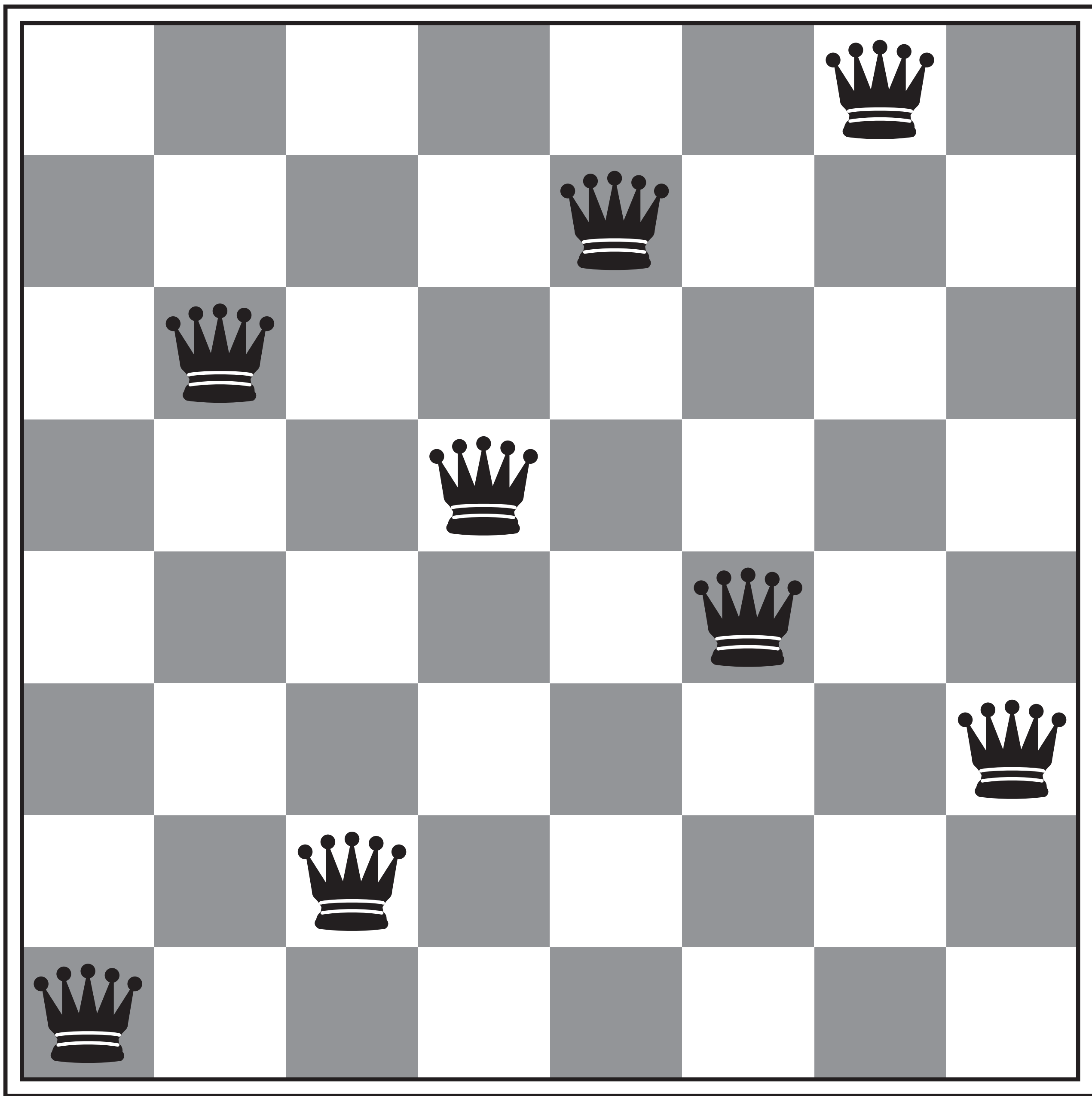
Hill climbing algorithm

- Loop that looks at all possible neighbors of the current state, and picks the one that increases the optimization function the most.

function HILL-CLIMBING(*problem*) **returns** a state that is a local maximum

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  loop do
    neighbor ← a highest-valued successor of *current*
    if neighbor.VALUE ≤ current.VALUE **then return** *current*.STATE
    current ← *neighbor*

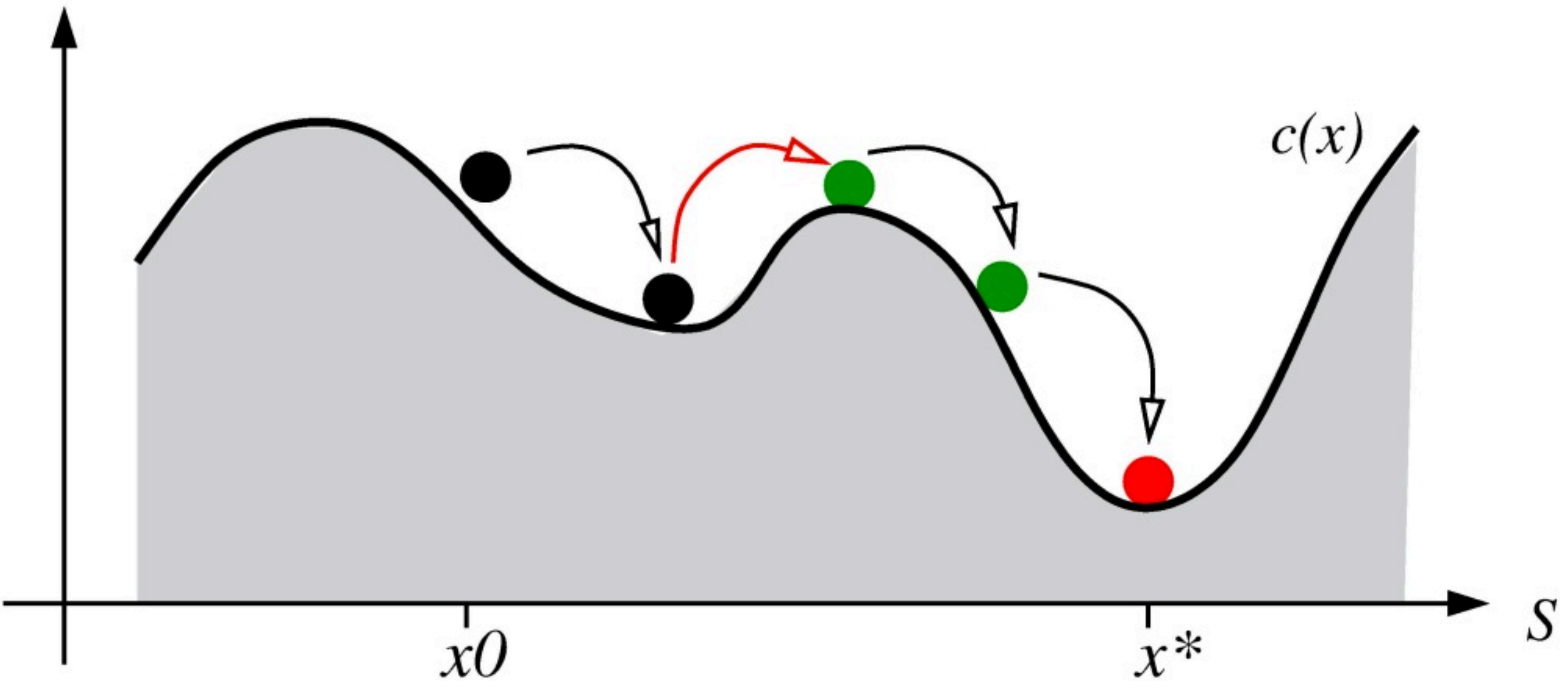
<table>
<tr><td>18</td><td>12</td><td>14</td><td>13</td><td>13</td><td>12</td><td>14</td><td>14</td></tr>
<tr><td>14</td><td>16</td><td>13</td><td>15</td><td>12</td><td>14</td><td>12</td><td>16</td></tr>
<tr><td>14</td><td>12</td><td>18</td><td>13</td><td>15</td><td>12</td><td>14</td><td>14</td></tr>
<tr><td>15</td><td>14</td><td>14</td><td>♛</td><td>13</td><td>16</td><td>13</td><td>16</td></tr>
<tr><td>♛</td><td>14</td><td>17</td><td>15</td><td>♛</td><td>14</td><td>16</td><td>16</td></tr>
<tr><td>17</td><td>♛</td><td>16</td><td>18</td><td>15</td><td>♛</td><td>15</td><td>♛</td></tr>
<tr><td>18</td><td>14</td><td>♛</td><td>15</td><td>15</td><td>14</td><td>♛</td><td>16</td></tr>
<tr><td>14</td><td>14</td><td>13</td><td>17</td><td>12</td><td>14</td><td>12</td><td>18</td></tr>
</table>



Variants

- Stochastic hill climbing
- Random-restart hill climbing

Simulated annealing



```
function SIMULATED-ANNEALING(problem, schedule) returns a solution state
  inputs: problem, a problem
          schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(problem.INITIAL-STATE)
  for t = 1 to ∞ do
      T ← schedule(t)
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← next.VALUE – current.VALUE
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(ΔE/T)
```

Temperature gets lower over time

Probability is lower with smaller T, and lower with bigger abs(delta-E)

Beam search

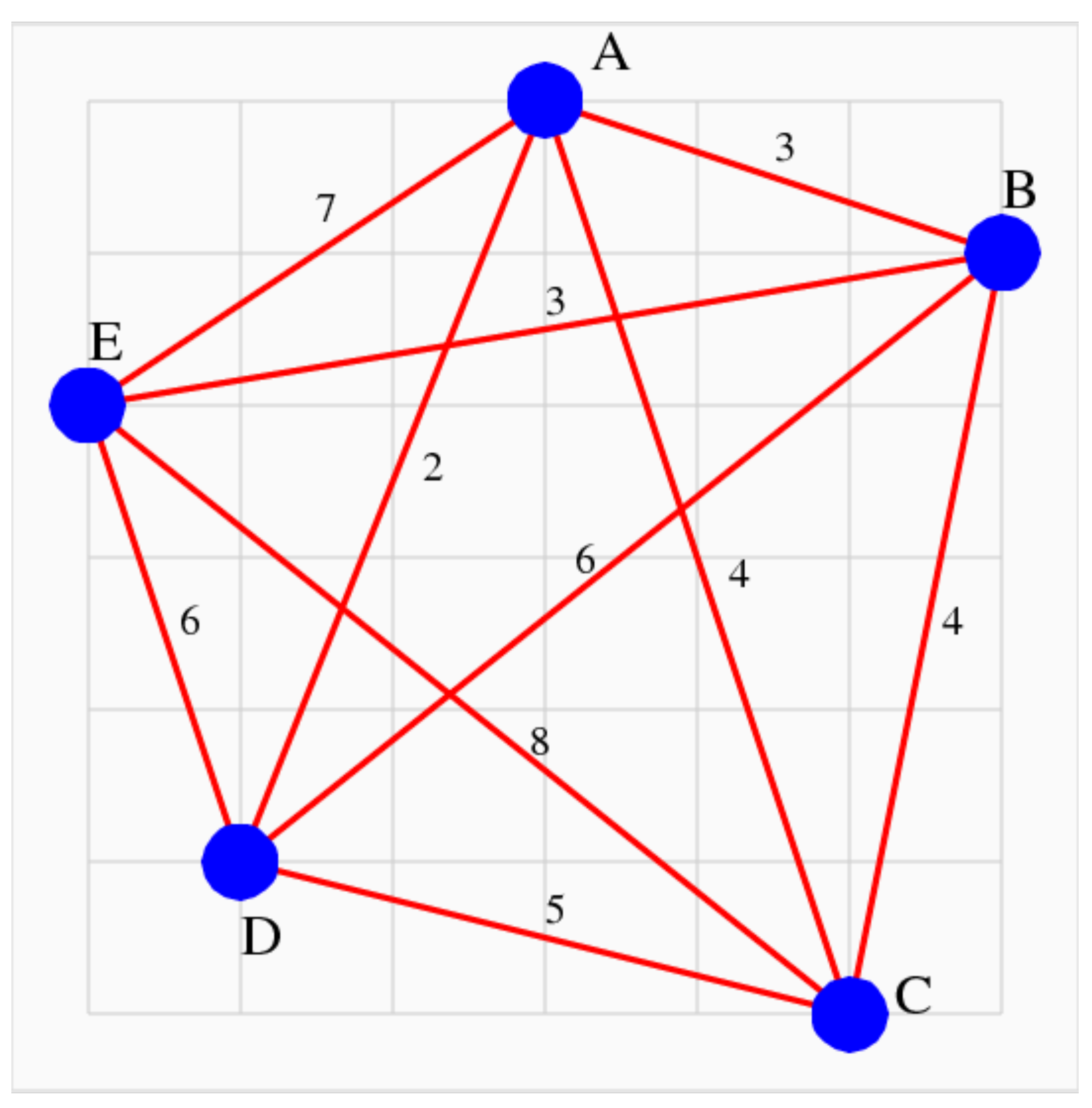
- Variation of hill climbing
  - Use k current states
  - Generate all of their successors
  - Take k best
- Variation: stochastic beam search
  - Adds in probabilistic idea from simulated annealing.
  - Same as above, but take k best successors based on probability.

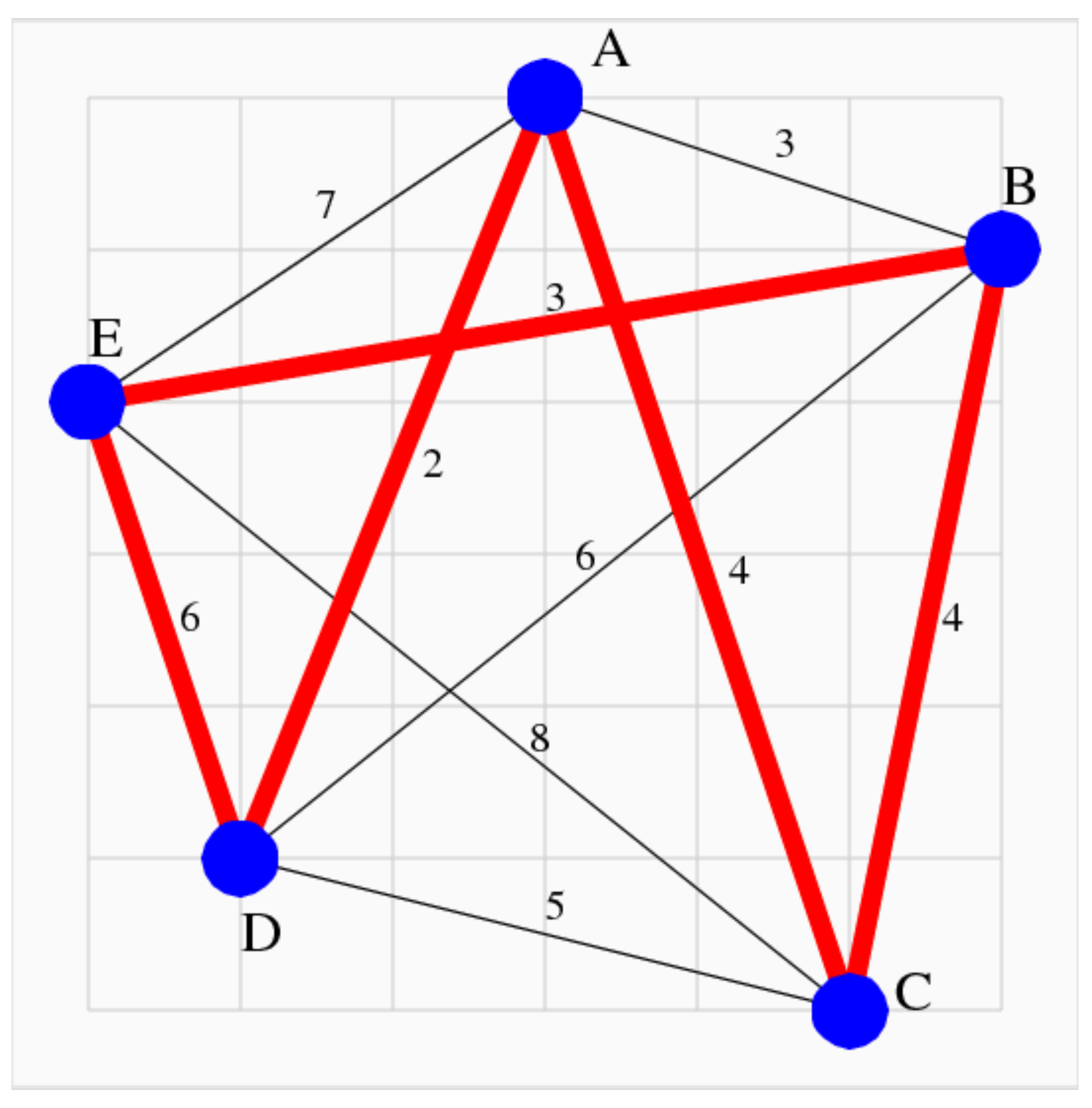
Genetic algorithms

- Variation on stochastic beam search.
- Successor states are generated using two parent states, not one. (Crossover)
- Mutation: Randomly modifies a current state.

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
  inputs: population, a set of individuals
          FITNESS-FN, a function that measures the fitness of an individual

  repeat
      new_population ← empty set
      for i = 1 to SIZE(population) do
          x ← RANDOM-SELECTION(population, FITNESS-FN)
          y ← RANDOM-SELECTION(population, FITNESS-FN)
          child ← REPRODUCE(x, y)
          if (small random probability) then child ← MUTATE(child)
          add child to new_population
      population ← new_population
  until some individual is fit enough, or enough time has elapsed
  return the best individual in population, according to FITNESS-FN
```


A
B
C
D
E
3
7
3
2
6
4
4
6
8
5


A
B
C
D
E
7
3
3
2
6
6
4
4
8
5