Hill Climbing

Algorithm 1 Hill Climbing

```
current ← a random state
while true do
    next ← get-best-neighbour(current)          → lowest cost .
    if cost(current) ≤ cost(next) then
        break                                   local optimum .
    end if
    current ← next                              move to neighbour .
end while
return current
```

002

Hill Climbing with Sideway Moves

Allow ≤ 100 consecutive sideway moves.

Algorithm 2 Hill Climbing with Sideway Moves

```
current ← a random state     ( initialize sideway limit & count.
while true do
    next ← get-best-neighbour(current)
    if cost(current) < cost(next) then
  (     strict local minimum
  (     stop/break.
    end if
    if cost(current) == cost(next) then
  (     if not over sideway limit
  (         then move to neighbour and increment sideway count.
        otherwise break.
    end if
  ( move to neighbour.
  (  reset sideway count.
end while
return current
```

Hill Climbing with Tabu List

Do not go back ≤ 100 recently visited nodes.

- How do you keep track of the most recent nodes visited?

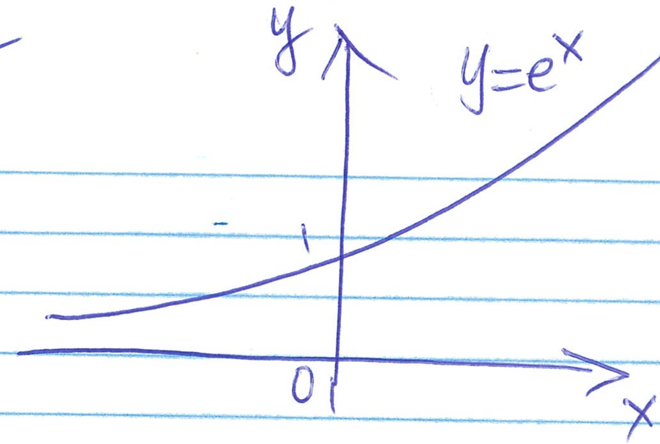
  queue.

- How would you update the list? (only useful on a plateau).

  if best neighbour in queue.
      consider next best neighbour.
  else
      if queue size below limit.
          add to queue.
      else.
          add to queue, remove the oldest.

Simulated Annealing

$$e^{\frac{\Delta E}{T}}$$



$\Delta E = -10$ Change $T=100$ to $T=10$.

$$e^{\frac{-10}{100}} = e^{-0.1} \approx 0.9 \qquad e^{\frac{-10}{10}} = e^{-1} \approx 0.36.$$

$\Delta E < 0$. $T > 0$. $\frac{\Delta E}{T} < 0$.

$$T\downarrow \quad \left|\frac{\Delta E}{T}\right|\uparrow \quad \frac{\Delta E}{T}\downarrow \quad e^{\frac{\Delta E}{T}}\downarrow.$$

$T=10$. $\Delta E = -10$ to $\Delta E = -100$.

$$e^{\frac{-10}{10}} = e^{-1} = 0.36 \qquad e^{\frac{-100}{10}} = 0.000045.$$

$$\Delta E\downarrow \quad |\Delta E|\uparrow \quad \left|\frac{\Delta E}{T}\right|\uparrow \quad \frac{\Delta E}{T}\downarrow \quad e^{\frac{\Delta E}{T}}\downarrow.$$

Genetic Algorithm 8-Queens.

	24748552	32752411	24415124	32543213
fitness	24	23	20	11
prob	31%	29%	26%	14%

parents	parents	parents	parents
24748552	32752411	24748552	24415124
32752411	24415124	32543213	32752411
child.	child.	child.	child.
24748411	32415124.	24743213	24752411
↓	↓	↓	↓
24748415	32415124.	54743213	24712411

- Keep track of multiple states.
- cost function: - fitness.
- neighbour relation:
  - choose parents proportional to fitness
  - cross over
  - mutate.

Early in the process
population is diverse., child can be very diff from parents
large steps in the space
Later on
individuals are similar, small steps.