NSTC

Local Search
population v. individual search

- Evolutionary algorithms are *population-based* search
  - population of candidate solutions
  - each evaluated for fitness
  - compete for selection
  - next generation: have a new (fitter?) population

- there are also *individual-based* searches
  - a single candidate solution
  - evaluate which individual to choose next
  - next generation: have a new (fitter?) individual
  - can be more efficient (fewer fitnesses to calculate)
  - but can more easily get “stuck” in local optima
  - *local* search
**local search procedure**

- A *local search* comprises a trace of execution:
  - **trace** $T_m = \langle (s_0, f_0), (s_1, f_1), \ldots, (s_m, f_m) \rangle$
  - $s_k$ are individual points in the search space
  - $f_k$ are the fitness values of the corresponding $s_k$

- Consecutive $s_k$ are related in a particular way:
  - $\forall k: 0 .. (m - 1) \quad s_{k+1} \in N(s_k)$

- The *neighbourhood function* $N(s_k)$ defines the set of points that are deemed to be “near to” or “in the locality” of $s_k$
  - one “move” away from $s_k$
  - the ones to look at in the next step
    - a reduced search space – a *local* search space
some local search questions

• how do you determine the **start state** $s_0$ ?

• how do you define the **neighbourhood** function $N$ ?

• how do you determine **which member** of the neighbourhood $N(s_k)$ is chosen to be the next state $s_{k+1}$ ?
“hill climbing” metaphor

• let the current “solution” or “point” be $s_k = x$

• define the neighbourhood $N(x)$ to be the set of solutions that are “immediately next to $x$”
  ▪ eg, adjacent grid positions

• if possible, move to a neighbouring point that improves the fitness $f(x)$, otherwise stop
  ▪ choose any $y \in N(x)$ as next solution, provided $f(y) \geq f(x)$
    ▪ weak hill-climbing (don’t go down)
  ▪ choose $y \in N(x)$ such that $f(y) = \max \{ z : N(x) \bullet f(z) \}$
    ▪ steepest gradient ascent (climb as fast as you can)

• hill-climbing can work very well – particularly if you climb the right hill – but there is a problem...
weak hill climbing: pseudocode

\[
\begin{align*}
\text{init}(s_0); \\
s &:= s_0; \\
\text{repeat} \\
& \quad s_{\text{new}} \in N(s); \\
& \quad \Delta := f(s_{\text{new}}) - f(s); \\
& \quad \text{if } (\Delta < 0) \text{ then } s := s_{\text{new}}; \quad // \text{ accept} \\
& \quad \text{else } \text{skip}; \quad // \text{ reject} \\
\text{until } \text{stopping criterion} \\
\text{return } s;
\end{align*}
\]
local optimisation - hill climbing

\[ N(x) = \{x-1, x+1\} \]

trajectory: \( x_0 \rightarrow x_1 \rightarrow x_2 \), since \( f(x_0) < f(x_1) < f(x_2) > f(x_3) \)

trajectory: \( y_0 \rightarrow y_1 \rightarrow y_2 \rightarrow x_{\text{opt}} \), since \( f(y_0) < f(y_1) < f(y_2) < f(x_{\text{opt}}) > f(x_{\text{opt}}+1) \)

really want to get here

starting choice of \( y_0 \) is much better!
a solution

allow non-improving moves: so that it is possible to go down, in order to rise again, to reach the global optimum
annealing : physics

- **annealing** is a thermal process for obtaining low energy states of a solid in a heat bath
  - increase the temperature of the heat bath until the solid melts
    - in liquid phase particles arrange themselves randomly
    - they “jiggle about” because of their thermal energy
  - carefully **decrease** the temperature of the heat bath until the particles arrange themselves in the ground state of the solid
    - in ground state the particles are arranged in a highly structured lattice and the energy of the system is minimal

- compare with **quenching** – very *rapid* lowering of temperature (eg. by dropping into a bath of cold water)
  - particles don’t have time to arrange themselves in the ground state
annealing: probabilities

- **thermal equilibrium** is characterised by the **Boltzmann** distribution
- the probability $p_i$ of being in state $i$ with energy $E_i$ is
  \[ p_i \propto \exp\left(\frac{-E_i}{kT}\right) \]
  - at high temperatures, high energy states are likely
  - as the temperature drops, only lower and lower energy states are likely
simulated annealing: analogy

- **simulated** annealing mimics the trajectory of physical transitions between states of various energies as the system cools
- candidate solutions $x$ are analogues of the states of a physical system $i$
- **cost** of a solution $f(x)$ is analogous to the **energy** of a state $E_i$
- transitions between candidate solutions are carried out **probabilistically** in an analogous manner to the distribution for describing physical state transitions
  - parameterised by a simulated “temperature” $T$
  - solutions “jiggle” out of local optima
simulated annealing: algorithm

- Improving moves always accepted.
- Non-improving moves are accepted with a probability depending on the temperature parameter \( T \), and how much worse the move is, \( \Delta \):
  - The worse the move, the less likely it is to be accepted.
  - The cooler the temperature, the less likely a worsening move is to be accepted.
- Temperature \( T \) starts high, and is gradually reduced as the search progresses:
  - Initially (when things are “hot”) virtually any move is accepted.
  - At the end (when things are nearly “frozen”) only improving moves are accepted.
    - And the search effectively reduces to hill-climbing.
simulated annealing: pseudocode

\[ \text{init}(s_0, T_0, L_0); \]
\[ k := 0; \]
\[ s := s_0; \]
repeat
  for \( l := 1 \) to \( L_k \) do
    \[ s_{\text{new}} \in N(s); \]
    \[ \Delta := f(s_{\text{new}}) - f(s); \]
    if \( \Delta < 0 \) then \( s := s_{\text{new}}; \) // accept
    elseif \( \exp(-\Delta/T) > U(0, 1) \) then \( s := s_{\text{new}}; \) // accept
    else skip; // reject
  enddo
  ++ \( k \);
  \[ L_k := \text{new}L(k); \]
  \[ T_k := \text{new}T(k); \]
until stopping criterion
return \( s; \)
cooling schedule

• most common is **geometric cooling**
  - reduce the temperature by some multiplicative factor $a$, where $0 < a < 1$, so $T_{k+1} := a \times T_k$
  - cooling factors most typically in the range 0.8 – 0.99 (with a bias towards the higher end)

• other methods are possible, e.g. **logarithmic cooling**
  - the rough rate of cooling is generally more important than the precise means of reduction
  - **the rate of cooling matters a lot**

• if you think the search isn’t going well (getting “stuck”) then you can “reheat” the system (raise $T$)
  - no guarantees : you may still get stuck in local optima
achieving thermal equilibrium

• at each temperature a number $L_k$ of trial moves are investigated.

• how big should $L_k$ be?

• this is harder to say
  ▪ there is some theoretical advice on how many moves you need to consider, but most people simply “experiment”
  ▪ people want results in good time – and so feel a need to use small $L_k$ – if not getting good enough results, then use larger values

• how should $L_k$ be changed?
  ▪ some spend less time at higher temperatures
  ▪ many simply make $L_k$ constant for all $k$
initial state

• what should $s_0$ be?

• most common approach to initial state selection is just random choice

• to overcome some of the limitations of the technique, can carry out multiple runs with different starting states
initial temperature

• what should \( T_0 \) be?

• we want a temperature at which a lot of moves are accepted
  ▪ this depends on the (unknown) search landscape
  ▪ one way is to progressively increase the temperature and execute the inner loop
    ▪ some tools progressively **double** the temperature as the means of determining \( T_0 \)
  ▪ when the acceptance rate reaches, say, 95%, we have an appropriate \( T_0 \) and can begin the annealing proper
stopping criterion

• when should we stop looking?

• usually time constrained

• various criteria:
  1. no state change for a “long time” (you decide)
  2. temperature below some threshold
    • Lundy & Mees aim to provide a result within $\varepsilon$ of the global optimum with probability $p$, in a search space size $|S|$
      $T < \frac{\varepsilon}{\ln\left(\frac{|S|}{p}\right)}$
  3. a real solution has been detected

neighbourhood: TSP example

- one aspect of neighbourhood definition concerns rapid cost function evaluation
- sometimes it is possible to simply calculate the change in cost function; eg, in the TSP:

\[
\text{new} = 4 + 7 + 5 + 6 + (4 + 5) = \text{old} + ((4 + 5) - (8 + 9)) = \text{old} + \Delta
\]
## neighbourhood: subset sum example

- You are given a set of integers $S = \{S_1, S_2, \ldots, S_n\}$
- Can you find a subset $Q$ of these integers that sums to a particular value $V$?

<table>
<thead>
<tr>
<th>in the subset: $Q$</th>
<th>43</th>
<th>76</th>
<th>96</th>
<th>32</th>
<th>86</th>
</tr>
</thead>
<tbody>
<tr>
<td>not in the subset: $S - Q$</td>
<td>40</td>
<td>56</td>
<td>13</td>
<td>97</td>
<td></td>
</tr>
</tbody>
</table>

Partition the set: $S = Q \cup (S - Q)$

A “move” takes an element from one partition to the other: from $(S - Q)$ to $Q$, or vice versa.

Eg, let target sum $V = 274$

Current cost $= |43 + 76 + 96 + 32 + 86 - 274|$

$= 333 - 274 = 59$

Delta cost easy to calculate (if you maintain current subset sum)
neighbourhoods

- note: **you define the neighbourhood**
  - variations are clearly possible

- for the subset sum problem
  - change the in/out status of a single element;
  - could have changed the status of 2, 3, 4, ... such elements
  - the number of elements in the neighbourhood gets larger with such $k$-element modification

- need to take into account the cost function too
  - the changes in cost should not be too radical
  - want some degree of **continuity** (smoothness) in the neighbourhood
lack of memory

• simulated annealing is a tremendously *simple* search
  ▪ but can be highly effective

• theory based on Markov chains
  ▪ Markov => “lack of memory” property
  ▪ once the search reaches a state, it doesn’t matter how it got there – it effectively “forgets” its past
  ▪ to move from $s_k$ to a neighbouring state $s_{k+1}$ that state must
    • be selected for consideration, with some probability $q_{k,k+1}$
    • pass the acceptance test, with some probability $p_{k,k+1}$
      • these probabilities may vary between temperatures
      • within a temperature cycle they are history independent
remembering your history

• there is nothing in standard annealing to prevent you going back to previously visited states during the search

  “Those who cannot remember the past are condemned to repeat it”
  -- [George Santayana, 1905]

• **tabu search** aims to incorporate memory as part of its search procedure
tabu search memory

- **tabu list**
  - when a move is taken (or state visited), it is placed on the tabu list for some number $L$ moves
  - that move cannot be retaken for the next $L$ moves – it is “tabu”
  - this form of short term memory helps avoid cycles in the search
  - promotes **diversification**

- **aspiration**
  - if taking a tabu move would give the best result yet then it may be taken! – promotes **convergence**

- **frequency**
  - long-term memory can be used to ensure particular move types are not taken too frequently over the whole search
  - promotes **diversification**
Example is a permutation problem representing order of filter applications (a move is simply a pairwise swap).

Tabu length $L = 3$ for this example

the best move is tabu, so we take the next one

the best move is tabu, but aspiration suggests we should take it anyway
local search

- local search can be very effective
- if one run doesn’t work, try multiple runs
- if we can identify several local optima, we may be able to use this information search the space more effectively
- and there is other information available during search ...
local search trajectories

• there may be a great deal of useful information in the trajectory (trace) of a search
  ▪ we are doing *guided* search, and each decision whether to move to a new state is based on information from the cost function landscape
  ▪ why throw all this away?
  ▪ the final result is only one of the “outputs” from a search
• sometimes analysis of the *trajectory* taken may provide information on the desired solution
  ▪ even when the final *result* is not that desired solution

• you will hear more about *trajectories* later in the module
why bother with anything else?

- the local optimum you end up in may depend very much on the initial starting state $s_0$
  - if the search space is sufficiently large it may be very unlikely that local search will find the global optimum (within any reasonable amount of time)

- **population based** (eg, EAs) approaches may give better results
  - can sample search spaces more effectively
  - and learn features of high performing solutions
    - other approaches may get you near the summit, then local search gets you to the peak within view: the “reading spectacles” of local search are great, other techniques have better “binoculars”

- **bio-inspired** searches:
  - evolutionary algorithms, swarms, artificial immune systems, ...
local search: summary and comments

• investigates solutions “close to” the current solution
  ▪ search trace is a walk around the search space, taking only “small steps”
    ✤ what is a “small step” is defined by you

• some steps you want to take: convergence
  ▪ eg. always accept improving moves; aspiration

• others you take in hope of reward later: diversification
  ▪ eg. probabilistically accept non-improving moves; tabu lists

• simple local search is a good place to start
  ▪ cheap to implement
  ▪ try it first, then something more sophisticated if you need to