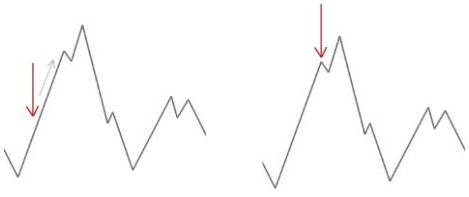
Computational Intelligence

Unit # 11

Global Optimization

- Simulated Annealing
- Tabu Search

Hill Climbing - Recap



- A hill climber algorithm will simply accept neighbour solutions that are better than the current solution. When the hill climber can't find any better neighbours, it stops.

Simulated Annealing

- Developed in 1983, SA is another widely used metaheuristic that enables the search process to escape from a local optimum.
- The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects.
- The heat causes the atoms to become unstuck from their initial positions (a local minimum of the internal energy) and wander randomly through states of higher energy; the slow cooling gives them more chances of finding configurations with lower internal energy than the initial one.

Analogy

- In simulated annealing, a temperature variable is used to simulate this heating process.
- The temperature is initially set as high and then is gradually cooled as the algorithm runs.
- While this temperature variable is high the algorithm will be allowed, with more frequency, to accept solutions that are worse than our current solution. This gives the algorithm the ability to jump out of any local optimums.
- As the temperature is reduced so is the chance of accepting worse solutions, therefore allowing the algorithm to gradually focus in on a area of the search space in which hopefully, a close to optimum solution can be found.
- This gradual 'cooling' process is what makes the simulated annealing algorithm remarkably effective at finding a an optimum solution when dealing with large problems which contain numerous local optimums.

Working of the Algorithm

- At each step, the SA heuristic considers some neighboring state *s'* of the current state *s*, and probabilistically decides between moving the system to state *s'* or staying in state *s*.
- If *s'* is better than our current state, It is accepted unconditionally.
- If however, it is not better, we need to consider a couple of factors to compute its probability of acceptance:
  - how much worse the *s'* is; and
  - how high the current 'temperature' of our system is. At high temperatures the system is more likely accept solutions that are worse.
- s' is accepted with a probability
  - $P = \exp(\Delta f/T)$

  Where Δf is the increase in energy and T is the control parameter "temperature".
- These probabilities ultimately lead the system to move to states of lower energy.

Outline of Simulated Annealing Algorithm

- Initialization
  - Start with a feasible initial trial solution
- Iteration
  - Use the *selection rule* to select the next trial solution.
  - If none of the immediate neighbors of the current trial solution are accepted, the algorithm is terminated.
- Check the temperature schedule
  - When the desired number of iterations have been performed at the current value of T, decrease T to the next value in the temperature schedule and resume performing iterations at this next value.
- Stopping Rule
  - Same as used for other algorithms discussed in this course

Sub-Tour Reversal in TSP

- A sub-tour reversal adjusts the sequence of cities visited in the current trial solution by selecting a subsequence of the cities and simply reversing the order in which that subsequences of cities is visited.
- The subsequence being reversed can consist of as few as two cities, but also can have more.
- The sub-tour reversal is an example of a local improvement procedure.

TSP Example: Temperature Schedule

- Five iterations are performed at each of the five values of T (T_1, T_2, T_3, T_4, T_5) in turn, where
  - $T_1 = 0.2Z_c$ when Z_c is the objective function value for the initial trial solution
  - $T_2 = 0.5T_1$
  - $T_3 = 0.5T_2$
  - $T_4 = 0.5T_3$
  - $T_5 = 0.5T_4$

TSP Example: Working

- Initial Trial
  - A – B – C – D – E – F – G – A, Zc = 69, T1=0.2xZc = 13.8
- Second Iteration
  - Pick two random numbers to find the swapping slots. Suppose the sub-tour of cities C – D is reversed.
  - A – B – D – C – E – F – G – A, Zn = 65
- Third Iteration
  - Reverse C – E – F
  - A – B – D – F – E – C – G – A, Zn = 64
- Fourth Iteration
  - Reverse C – G
  - A – B – D – F – E – G – C – A, Zn = 66 and Zc = 64 implies $\Delta f = -2$
  - Prob {acceptance} = exp (-2/13.8) = 0.865
  - If the next random number generated is less than 0.865, this candidate solution will be accepted as the next trial solution. Otherwise it will be rejected.

Tabu Search

Tabu Search

- The word tabu (or taboo) comes from Tongan, a language of Polynesia, where it was used by the aborigines of Tonga island to indicate things that cannot be touched because they are sacred.
- According to Webster's Dictionary, the word now also means "a prohibition imposed by social custom as a protective measure" or of something "banned as constituting a risk."

Tabu Search

- Tabu search uses common-sense ideas to enable the search process to escape from a local optimum.
- It uses a local search procedure to find a local optimum.
- A key strategy of tabu search is that it then continues the search by allowing *non-improving moves* to the best solutions in the neighborhood of the local optimum.
- The danger with this approach is that after moving away from a local optimum, the process will cycle right back on the same local optimum.

Tabu List

- To avoid this, a tabu search temporarily forbids moves that would return to (or perhaps towards) a solution recently visited.
- A **tabu list** records these forbidden moves, which are referred to as tabu moves.
- The only exception to forbidding such a move is if it is found that a tabu move actually is better than the best feasible solution found so far.
- This use of *memory* to guide the search by using tabu lists to record some of the recent history of the search is a distinctive feature of tabu search.

Outline of a Basic Tabu Search Algorithm

- Initialization
  - Start with a feasible initial trial solution.
- Iteration
  - Use an appropriate local search procedure to define the feasible moves into the local neighborhood of the current trial solution.
  - Eliminate from consideration any move on the current tabu list unless that move would result in a better solution than the best trial solution found so far.
  - Determine which of the remaining moves provides the best solution and adapt it regardless of whether it is better or worse than the current trial solution.
  - Update the tabu list. If the tabu list is already full, delete the oldest member of the tabu list.
- Stopping Rule
  - Same as used for other algorithms discussed in this course

Example Revisited

	A	B	C	D	E	F	G
A	0	12	10				12
B	12		8	12			
C	10	8		11	3		9
D		12	11		11	10	
E			3	11		6	7
F				10	6		9
G	12		9		7	9	

A – B – C – D – E – F – G - A : 12 + 8 + 11 + 11 + 6 + 9 + 12 = 69

Working of Tabu Search

- Initial solution
  - A – B – C – D – E – F – G – A : Distance = 69
  - Tabu list: Blank at this point
- Iteration 1: Reverse C – D
  - Deleted Links: B – C, D – E
  - Added Links: B – D, C – E
  - Tabu List: B – D, C – E
  - A – B – D – C – E – F – G – A : Distance = 65
- Iteration 2: Reverse C – E – F
  - Deleted Links: D – C, F – G
  - Added Links: D – F, C – G
  - Tabu List: B – D, C – E, D – F, C – G
  - A – B – D – F – E – C – G – A : Distance = 64

Working of Tabu Search (Cont'd)

- Iteration 3: Reverse C – G
  - Deleted Links: E – C, G – A
  - Added Links: E – G, C – A
  - Tabu List: ~~B – D, C – E~~, D – F, C – G, E – G, C – A
  - A – B – D – F – E – C – G – A : Distance = 66
- The process continues in a similar fashion until one of the stopping condition is satisfied.