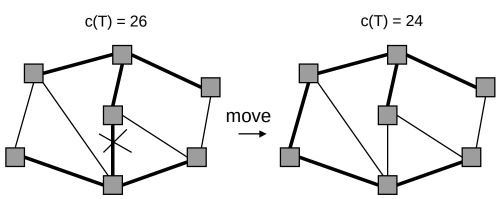
CSE 589
Applied Algorithms
Spring 1999

Local Search
Simulated Annealing

Local Search Algorithms

- Start with an initial solution that is usually easy to find, but is not necessarily good.
- Repeatedly modify the current solution to a nearby one looking for better ones.

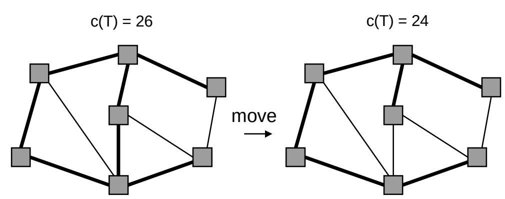
Neighborhood of a Solution



Cut an edge breaking tree into two trees.

Add an edge joining the two parts together again.

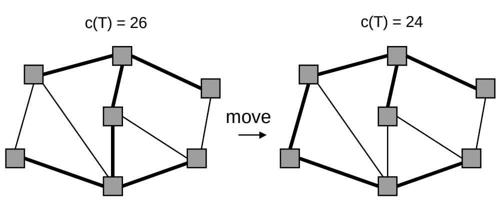
Equivalent Move



Add an edge forming a cycle.

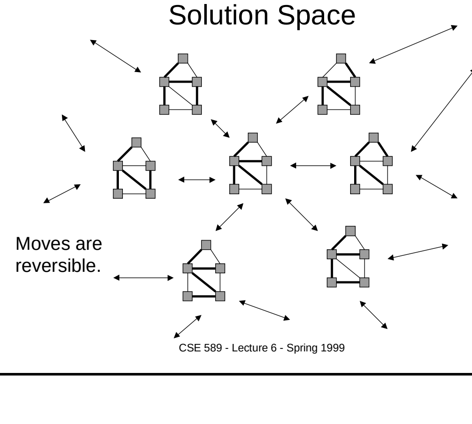
Delete an edge in the cycle.

Recall Cost



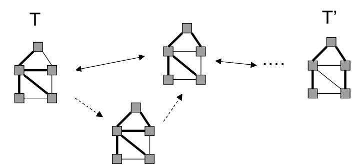
c(T) = sum of squares of the degrees of the vertices in T

Solution Space



Every Spanning Tree is Reachable from Every Other

- Let T and T' be two spanning trees. We can move T closer to T' by adding an edge from T' to T and removing an edge in the cycle formed that is not in T'.



Potential of Local Search

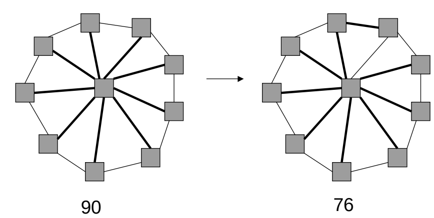
- Since every spanning tree is reachable from any other, then starting with an arbitrary spanning tree we can move to an optimal one using local search.
- Impediment: there can be exponentially may spanning trees. The search space is exceedingly large.
- In what direction do we search.

Greedy Local Search

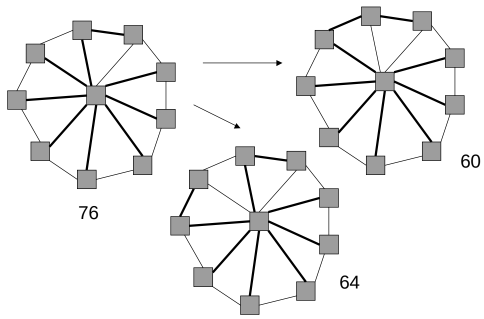
- Find the best neighbor and continue.

```
T := an initial tree;
best-cost := c(T);
repeat
    cost := best-cost;
    for each neighbor T' of T do
       if c(T') < best-cost then
          T := T';
          best-cost := c(T);
until (best-cost = cost)
return(T)
```


Greedy Example (1)



Greedy Example (2)



Analysis of Greedy Local Search

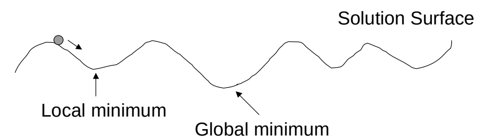
- Assume n vertices, m edges and D is the sum of the squares of the degrees in G. $D \leq n^2$.
- There are at most D iterations in the algorithm.
- Each iteration consists of looking at each edge in the spanning tree and replacing it with some other edge, and checking for a cycle and computing costs. This is roughly $O(n^2m)$ time per iteration.
- Total time is $O(D\ n^2m) = O(n^4m)$ (worst case).

Notes on Greedy Local Search

- Can be very effective for some problems. The worst case time is not that bad.
- Examining all the neighbors and choosing the best is sometimes called “steepest decent”.
- An alternative is “random decent”. Randomly choose a neighbor and move to it if its cost is smaller.
- Another alternative is “first decent”. Try the neighbors in order and move to the first improvement.

Local Minimum Problem

- Greedy local search leads to a local minimum in the solution space, not necessarily a global minimum.

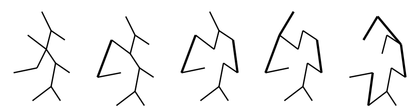


Double Moves

- Instead of just looking one move away, look two moves away to find a better spanning tree.
- Unfortunately, this incurs another huge factor of n^2m.

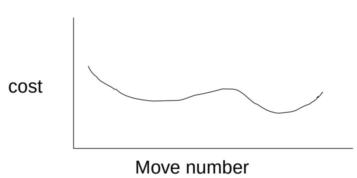
N -1 Move Strategy

- Start with all edges unlocked. For each move we lock the new edge that was added, until all edges are locked or no moves possible.
- We move even if the cost goes up!



- Pick the best in the sequence to be the new best tree.

Avoiding Local Minima



Multiple Move Strategies

- Multiple move stategies were pioneered by Kernighan and Lin (1970).
- They have been proven to be very effective for the traveling salesman problem and the minimum cut graph partitioning problem.

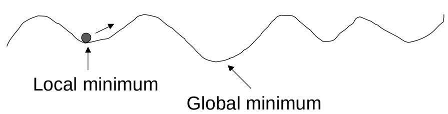
Using Randomness to Avoid Local Minima

- We maintain a trial solution.
  – Generate a random move from the trial solution.
  – If the move would beat the trial solution then accept it as the new trial solution.
  – If the move does not improve the solution then accept it with some small probability.
- This enables us to navigate the entire solution space and not get caught in a local minimum.

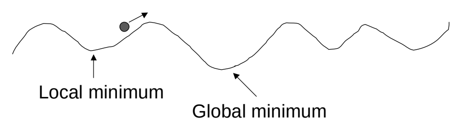
Simulated Annealing

- Kirkpatrick (1984)
- Analogy from thermodynamics.
- The best crystals are found by annealing.
  – First heat up the material to let it bounce from state to state.
  – Slowly cool down the material to allow it to achieve its minimum energy state.

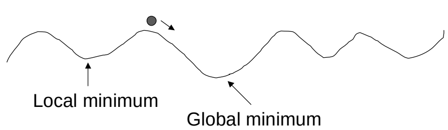
Heating and Cooling Helps (1)



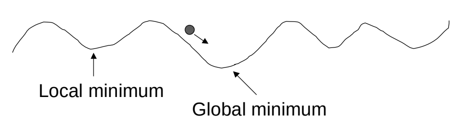
Heating and Cooling Helps (2)



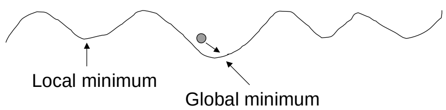
Heating and Cooling Helps (3)



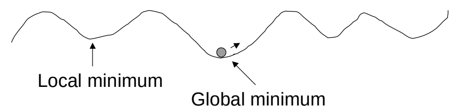
Heating and Cooling Helps (5)



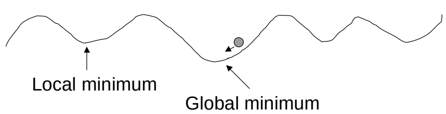
Heating and Cooling Helps (6)



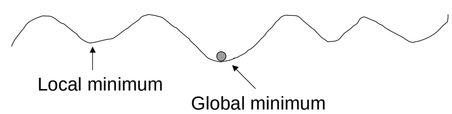
Heating and Cooling Helps (7)



Heating and Cooling Helps (8)



Heating and Cooling Helps (9)



Annealing Concepts

- Solution space S, x in S is a solution
- E(x) is the energy of x
- x has a neighborhood of nearby states
- T is the temperature
- Cooling schedule, in step t of the algorithm
  - Fast cooling $T = ae^{-bt}$
  - Slower cooling $T = at^{-b}$

Metropolis Algorithm

```
initialize T to be hot;
choose a starting state;
repeat
   generate a random move
   evaluate the change in energy dE
   if dE < 0 then accept the move
   else accept the move with probability e^(-dE/kT)
   update T
until T is very small (frozen)
```

Applied to Load Balanced Spanning Tree

- A state is a spanning tree.
- T’ is a neighbor of T if it can be obtained by deletion of an edge in T and insertion of an edge not in T.
- Energy of a spanning tree T is its cost, c(T).
  - If T’ is the random neighbor of T then dE = c(T’) - c(T).
- Probability of moving to a higher energy state is $e^{-(c(T') - c(T))/kT}$
  - Higher if either c(T’) - c(T) is small or T is large.
  - Low if either c(T’) - c(T) is large or T is small.

Notes on Simulated Annealing

- Not a black box algorithm.
  - Requires tuning the cooling parameters and the constant k in the probability expression $e^{-dE/kT}$.
- Has been shown to be very effective in finding good solutions for some optimization problems.
- Known to converge to optimal solution, but time of convergence is very large. Most likely converges to local optimum.
- Very little known about effectiveness generally.

Randomness and Relaxation

- Reduce problem to integer programming or integer semi-definite programming. (Goemans and Williamson, 1994)
  - Relax to linear programming or semi-definite programming yielding a non-integer solution. There are polynomial time algorithm for LP and SDP.
  - Randomized rounding to achieve a good integer solution.
  - Provable bounds on approximation.

Genetic Algorithms

- Maintain a population of solutions.
  - Good solutions mate to obtain even better ones as children.
  - Bad mutations are killed.