Minimum spanning tree problem

- Consider a connected graph G=(V,E)
  - V={$v_1, v_2, \ldots, v_n$} finite set of vertices
  - E={$e_1, e_2, \ldots, e_n$} finite set of edges
  - each edge e_i has positive real weight w_i representing distance or cost
  - the Minimum Spanning Tree (MST) is a least-weight subgraph connecting all vertices of graph G

Minimum spanning tree problem

- Let x be a binary decision variable defined as
  - $x_i = \begin{cases} 1 & \text{if edge ei is selected} \\ 0 & \text{otherwise} \end{cases}$
  - let T denote the set of all spanning trees for G
  - the minimum spanning tree problem can be formulated as:
    - $\min\{\ z(x) = \sum_{.i=1}^{.m} w_i\, x_i \mid x \in T\ \}$
  - easy to solve in polynomial time

Quadratic MST problem (q-MST)

- Takes two types of cost into consideration
  - direct cost wi
  - interactive cost due to interaction between edges
  - let Cik denote interactive cost due to selecting edges ei,ek
  - the problem can be formulated as
  - $\min\{ z(x) = \sum_{.i=1}^{.m} \sum_{k=1,k\neq i}^{m} c_{ik} x_i x_k + \sum_{.i=1}^{.m} w_i x_i \mid x \in T \}$

Heuristic Algorithms

- Heuristic algorithm H1: (average contribution)
  - if edge e_k is singly taken into consideration we can rewrite z(x) as:
  - $z(x)=[w_k + \sum_{.j \neq k} (c_{jk} + c_{kj})\, x_j]\, x_k + z_2(x)$
  - where $z_2(x)$ no longer involves x_k
  - the summation has m-1 terms but only n-1 terms are non zero
  - thus the average contribution of edge e_k is
  - $P_k = w_k + (n-1)/(m-1) \sum_{.j \neq k} (c_{jk} + c_{kj})$
  - using P_k instead of w_k reduces the q-MST to MST

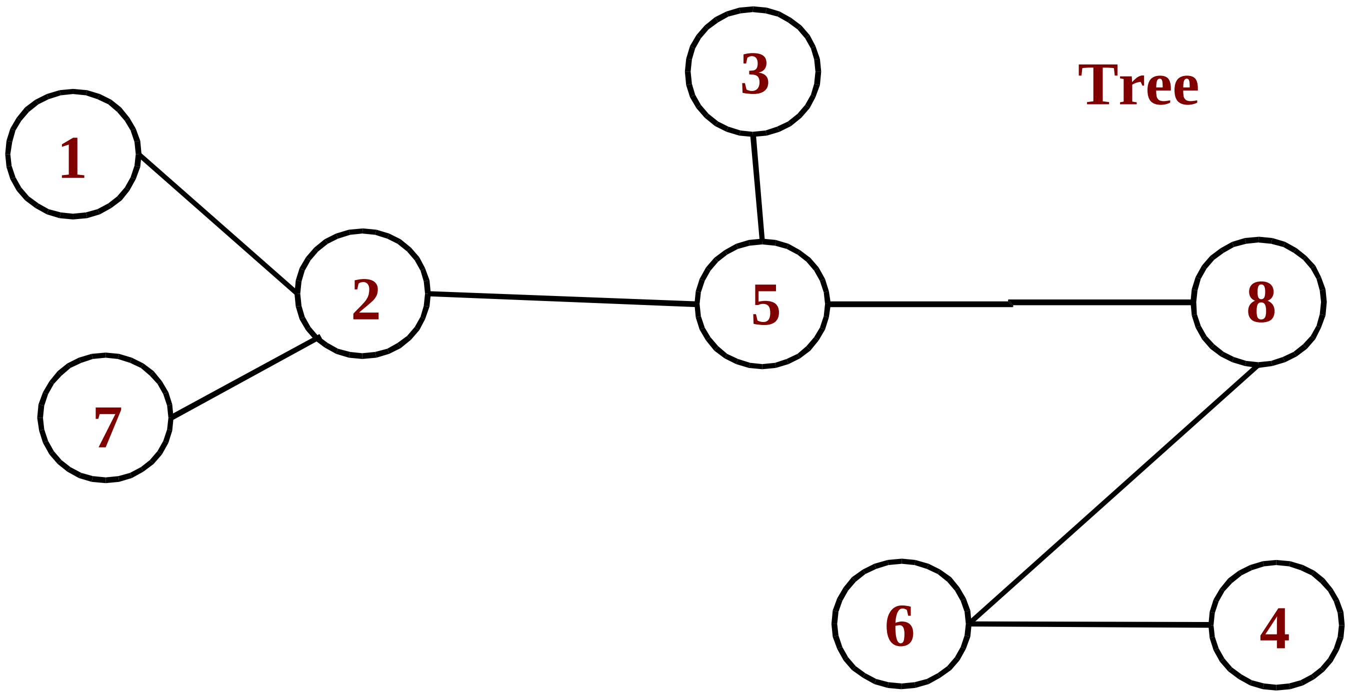
Heuristic Algorithms

- Heuristic Algorithm H2: (sequential fixing)
- similar to H1 but the average contribution for any edge is adjusted based on the number of edges already in the tree

Genetic algorithm for q-MST

- Possible representations:
  - binary array of all edges (size=m) very large!!
  - vector of integers: for each node, which node is its parent ; special number for the root's parent (size=n)
    - problem: does not insure feasible offspring
  - Prufer number (size=n-2)
    - insures feasible offspring
    - problem: low causality

Prufer number



Tree

2	5	6	8	2	5

Prufer number

Converting a tree to a Prufer number

- Step 1: Let i be lowest-numbered leaf in tree T. let j be the parent of i. Then j becomes the rightmost digit of the Prufer number P(T). P(T) is built by appending digits to the right (left to right).
- Step 2: Remove i and edge (i,j) from further consideration.
- Step 3: If only 2 nodes remain, P(T) has been formed, so stop; otherwise return to Step 1.

Converting a Prufer number to a tree

- Step 1: Let P(T) be the original Prufer number, and let Q(T) be the set of nodes that are not part of P(T).
- Step 2: Repeat until no digits are left in P(T):
  - let i be the lowest-number in Q(T) and j be the leftmost digit of P(T).
  - add edge (i,j) to tree T.
  - remove j from P(T) and i from Q(T). If j does not occur anymore in remaining part of P(T), put it in Q(T).
- Step 3: There should be exactly 2 nodes x and y remaining in Q(T). Add edge (x,y) to T and stop.

GA for q-MST problem

- Proposed by Zhou and Gen in 1998
- relies on the Prufer number representation
- Crossover: uniform
- Mutation: Uniform in the range 1 to n
- Fitness: convert Prufer number to tree and evaluate total cost using q-MST objective
- Selection: $(\mu+\lambda)$-selection
  - select μ best chromosomes from μ parents and λ offspring then use crossover and mutation to create λ offspring

Results

- GA always beats H1 and H2
- H1 sometimes beats H2 (strange?)
- GA's advantage decreases as problem size increases (why?)