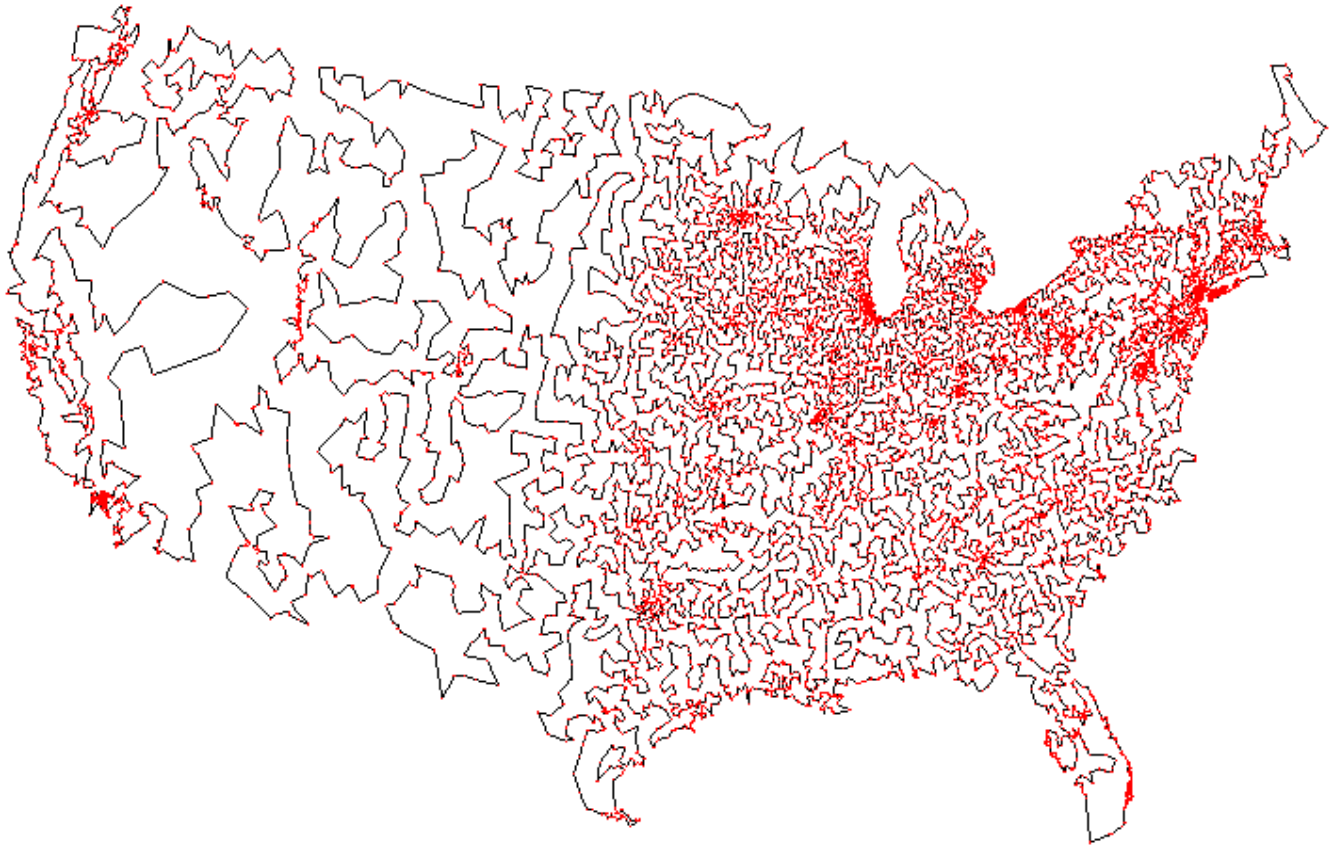
CSE 6140

PROJECT

metric Traveling Salesperson Problem

- TSP. Given a set of n cities and a pairwise distance function d(u, v), is there a tour of length $\leq$ D?



Optimal TSP tour
Reference: http://www.tsp.gatech.edu

Teams

- Randomly assigned
- Group of 3 (one has 4)
- 25 teams
- You have until Monday to make swaps
- Have to negotiate between each other and email with names of the students and their team Letter that are swapping
  - Both students should be cc'ed in the email
  - Reason for swaps in incompatible Programming Languages

Algorithms

- Greedy Construction Heuristic (Furthest Insertion)
- Approximation Algorithm (MST 2-approx)
- Local Search (2 variants)
  - Choose a method to select a starting solution
  - Choose a neighborhood (2-opt, 3-opt)
  - Choose a method to explore search space
    - Hill Climbing, Simulated Annealing, Iterated Local Search, Tabu Search, Genetic Alg
- Branch-and-Bound
  - How do you branch/expand on each node?
  - How do you bound on each node?
- Other methods?

Executable

- Parameters
  - -alg [BnB | Approx | Heur| LS1 | LS2]
  - -inst filename.tsp
  - -time cutoff (in minutes)
- Output:
  - SolFile <filename_method.sol> with 2 lines:
    - line1: quality of best solution found
    - line2: list of vertices on tour v1,v2,…,vn,v1
  - SolTrace <filename_method.trace> for BnB and LS
    - Each line is 2 numbers: timestamp quality
    - Record every time when a new better solution is found

Deliverables

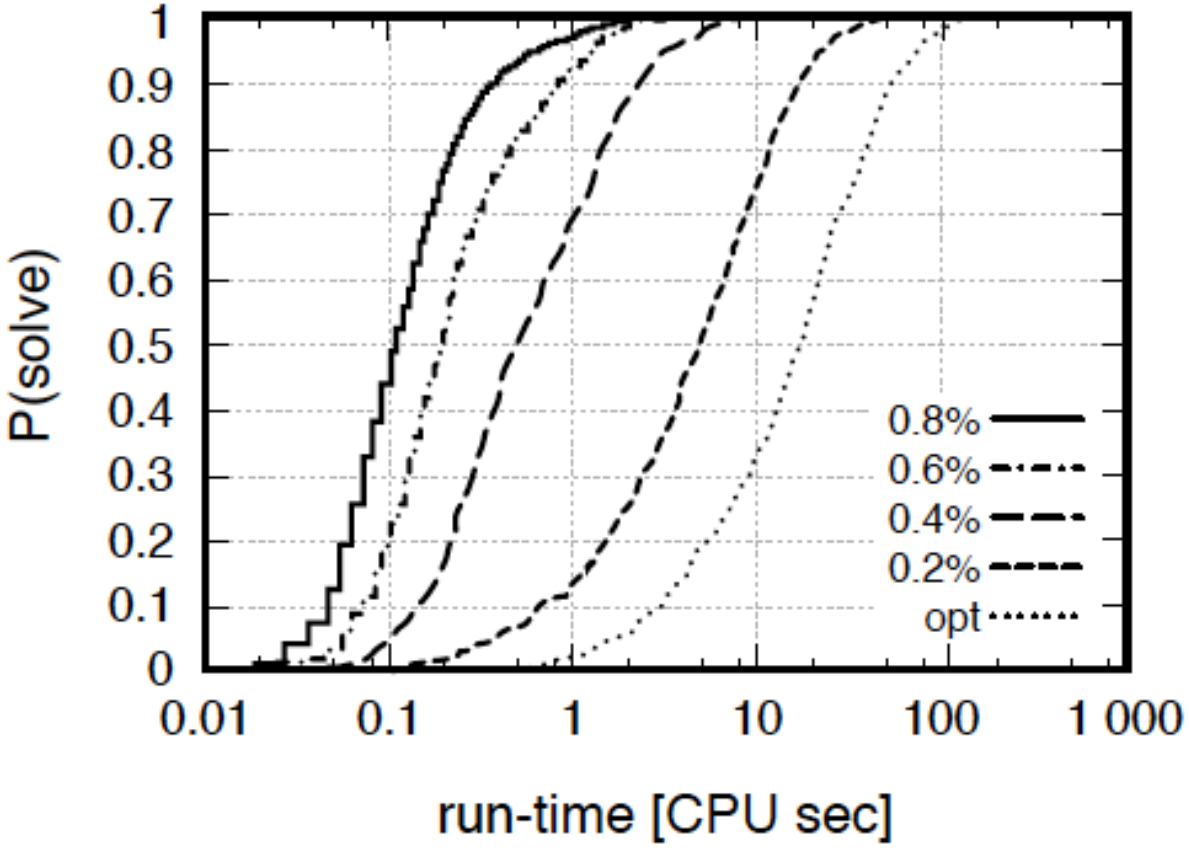
- Partial report/check in on Thur. Nov 20
  - You should have be able to parse input and produce output
  - You should have at least 2 approaches working by then
  - Report best quality within 10 mins for each benchmark instance
- Presentation last class Dec 4 with 5 mins per team
- Report (the bulk of your grade)
  - Written like a proper academic paper
  - Describes your approaches and choices, Data structures, Worst-case running time
  - For each instance, report results on all ALGs in terms of relative optimality gap and time with max 10 mins time cutoffs
  - Does the empirical running time match the worst-case complexity? How does the empirical relative error compare to the approximation guarantee? How do methods comapare?

Deliverables

- Local Search Approaches
  - Why did you make the choices that you did?
  - Report any sources of ideas you have used
  - Try to be creative – make your own variant of a known LS approach, don't just implement exactly the same
  - Generate QRTDs and SQDs for selected 3 instances
  - Boxplot for each instance with Mean, Median, Q1, Q3, etc

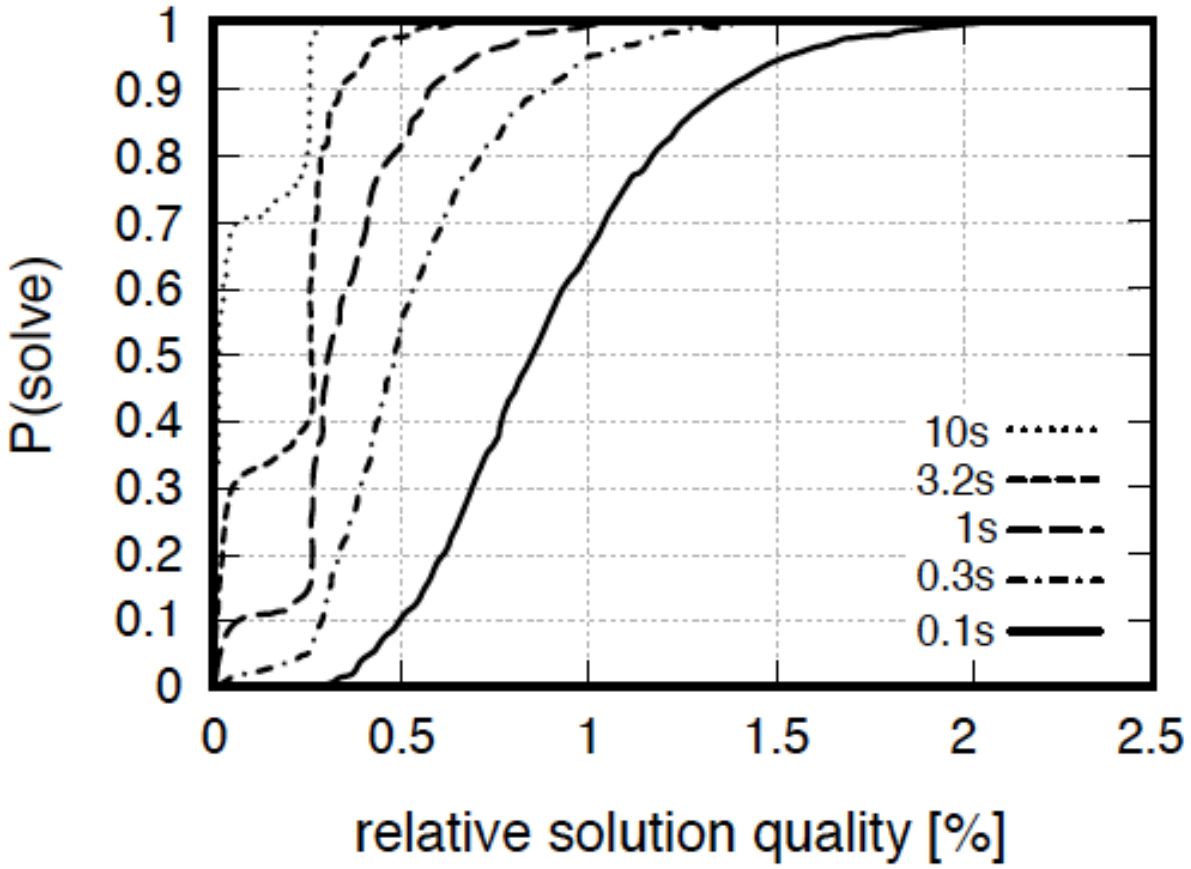
Georgia Tech

Qualified RTDs for various solution qualities:



Georgia Tech

Solution quality distributions for various run-times:



Local Search hints

- Implementation Optimizations:
  - Pre-compute the distance matrix
  - For each node, have a sorted list by distance of edges to the other nodes
  - Consider first-improvement Selection versus best-improvement
  - Consider only the K=5 nearest neighbors of a node for the 2-opt or 3-opt exchanges

- Enter your LS approaches in our Competition!!! – submit entries of your LS approaches each week to compete against other teams

Team Participation

- Each individual e-mail me a thoughtful and honest evaluation of the contributions of your group members, including yourself.
- For each individual, include a score from 1 to 10 indicating your evaluation of their work (10 meaning they were a valuable member of the team that made significant contributions to the project and were good to work with, 1 meaning they contributed none or very little and were difficult to work with).
- You may also include any clarifying comments, etc. (especially for low scores).
- If a person receives consistently low evaluations from peers, then their score will be proportionally decreased.

Example: TSP file EUC

NAME: kroA100

TYPE: TSP

COMMENT: 100-city problem A (Krolak/Felts/Nelson)

DIMENSION: 100

EDGE_WEIGHT_TYPE : EUC_2D

NODE_COORD_SECTION

1 1380 939

2 2848 96

3 3510 167

…

EOF

Example: TSP file GEO

NAME: burma14

TYPE: TSP

COMMENT: 14-Staedte in Burma (Zaw Win)

DIMENSION: 14

EDGE_WEIGHT_TYPE: GEO

EDGE_WEIGHT_FORMAT: FUNCTION

DISPLAY_DATA_TYPE: COORD_DISPLAY

NODE_COORD_SECTION

1 16.47 96.10

2 16.47 94.44

3 20.09 92.54

…

EOF

Branch-and-Bound algorithm

Georgia Tech

Branch-and-Bound(P) // Input: minimization problem P
01 F ← {(∅,P)} // Frontier set of configurations
02 B ← (+∞ , (∅,P)) // Best cost and solution
03 **while** F not empty **do**
04 *Choose* (X,Y) in F – the most ”promising” configuration
05 *Expand* (X,Y), by making a choice(es)
06 Let (X_1,Y_1), (X_2,Y_2), ..., (X_k,Y_k) be new configurations
07 **for** each new configuration (X_i,Y_i) **do**
08 ”*Check*” (X_i,Y_i)
09 **if** ”solution found” **then**
10 **if** cost(X_i) < B cost **then** // update upper bound
11 B ← (cost(X_i),(X_i,Y_i))
12 **if** not ”dead end” **then**
13 **if** *lb*(X_i) < B cost **then** //
14 F ← F U {(X_i,Y_i)} // else prune by lb
15 **return** B

TSP

- Partial solution: a path from a start node a to b, going through nodes T – (a,T,b)
- *Choose*: the partial assignment with smallest lower bound (most promising)
- *Expand*: add any edge from b to V-T-{a,b}

- Lower bounds?

Traveling Salesman Problem—Bounding Function 2 **Dynamic (2 Shortest Edges)**

Georgia Tech

- Given a partial solution (a,T,b)
- We have a path from a to b using vertices $T \subseteq V - \{a, b\}$

- A lower bound is the sum of:
  - The partial path we have
  - A lower bound on exiting a and b (half of their shortest edge **to a vertex in V-T-{a,b}**)
  - A lower bound on entering and exiting each of the remaining vertices (the sum of the two shortest adjacent edges **to nodes in V-T** divided by 2)

Traveling Salesman Problem—Bounding Function 3 **Dynamic (MST)**

Georgia Tech

- Given a partial solution (a,T,b)
- We have a path from a to b using vertices $T \subseteq V - \{a, b\}$

- A lower bound is the sum of:
  - The partial path we have
  - A lower bound on exiting a and b (their shortest edge **to a vertex in V-T-{a,b}**)
  - A lower bound on visiting the remaining nodes (The cost of the Minimum Spanning Tree for the subgraph over nodes in **V-T-{a,b}**)

2-approximation for TSP using MST

APPROX-TSP-TOUR(G)

Find a MST T;

Choose a vertex as root r;

return preorderTreeWalk(T, r);

- preorderTreeWalk(T,r)
  - depth first search in the tree, and output each node the first time that you enter it
  - Exactly the same order as the Eulerian tour and the shortcutting
- Note: There is a 1.5 approximation algorithm for metric TSP.
-

Greedy Construction Heuristic: Furthest Insertion

- starts from a single vertex, and then inserts new points into this partial tour one at a time until the tour is complete.
- The version of this heuristic that seems to work best is *furthest point* insertion: of all remaining points, insert the point *v* into a partial tour *T* such that
  - $\max_{v \in V \setminus T} \min_{i=1}^{|T|} (d(v, v_i) + d(v, v_{i+1})$
- The "min" ensures that we insert the vertex in the position that adds the smallest amount of distance to the tour,
- while the "max" ensures that we pick the worst such vertex first.
- This seems to work well because it "roughs out" a partial tour first before filling in details.
- Such tours are typically only 5% to 10% longer than optimal.

Local Search (LS)

- start from initial position
- iteratively move from current position to one of neighboring positions
- use evaluation function to choose among neighboring positions

Hill climbing

Georgia Tech

```
function Hill-Climbing(problem) returns a solution  state


current ← Make-Node(Initial-State[problem])
loop do
    next ← a highest-valued successor of current
    if  Value[next] < Value[current] then return current
    current ←next
end
```

SA Pseudo code

function Simulated-Annealing(*problem*, *schedule*) **returns** *solution state*

current ← Make-Node(Initial-State[*problem*])

for t ← 1 to *infinity*

T ← *schedule*[*t*] // T goes downwards.

if *T* = *0* **then return** *current*

next ← Random-Successor*(current)*

ΔE ← f-Value[*next*] - f-Value[*current*]

if $\Delta E > 0$ then *current* ← *next*

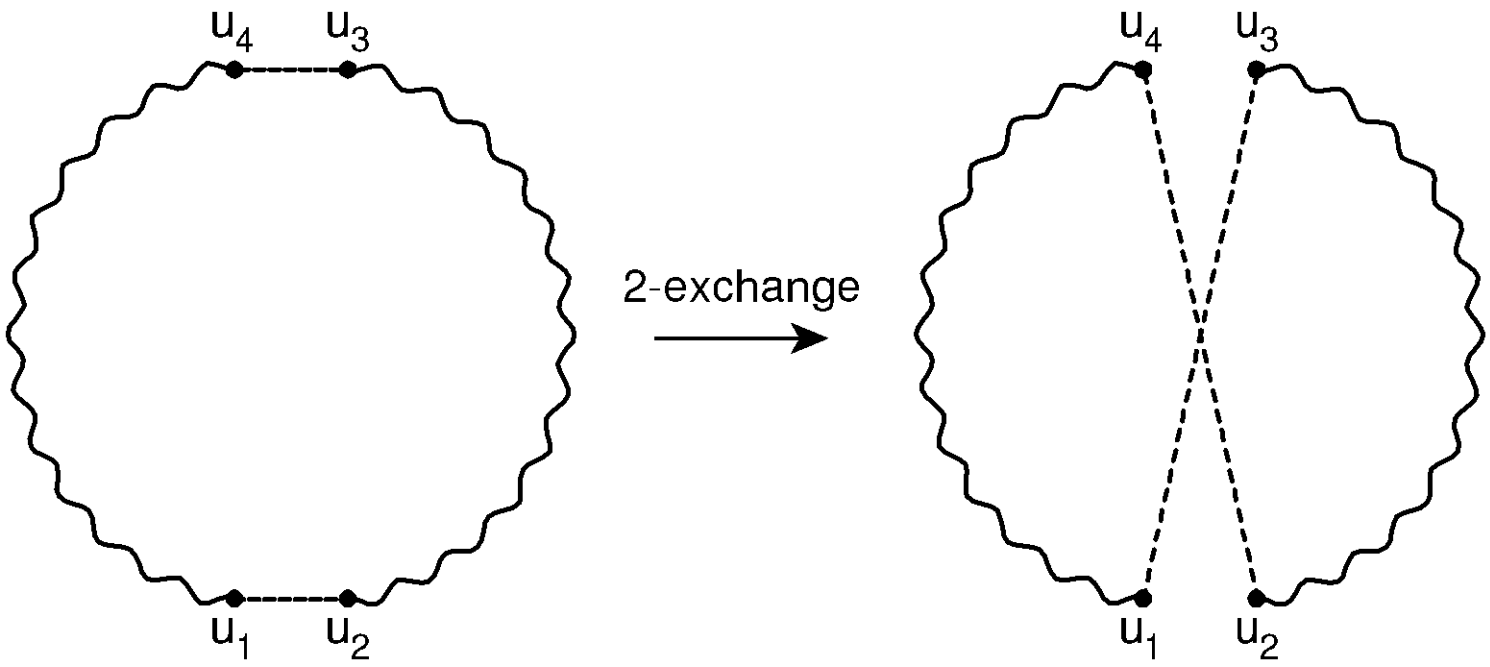
else *current* ←*next* with probability $e^{\Delta E/T}$

end

Iterated local search

- Generate initial candidate solution s
- Perform local search on s (for example iterative improvement starting from s)
- While termination condition not met
  - Set r=s
  - Perform perturbation on s
  - Perform local search on perturbed s
  - Based on acceptance criterion, keep s or revert to r

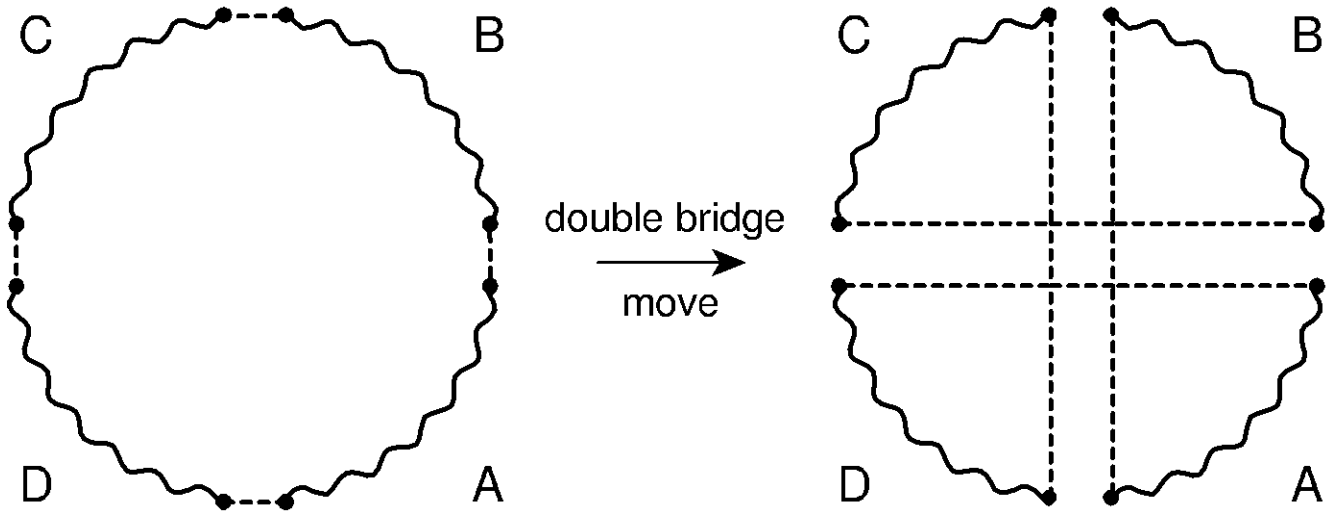
Symm. TSP --- search neighborhood



3-opt – delete 3 edges and reconnect fragments into 1 cycle
k-opt – delete k edges and reconnect fragments into 1 cycle

Iterated local search for TSP

- Perturbation: "double-bridge move" = 4-exchange step
- Cannot be directly reversed by 2-exchange moves



Nominate your TAs for an award

- Did you find one of the TAs for this class particularly helpful
- Show your appreciation by nominating them for Graduate TA award
- Deadline Nov 9
- Please visit http://cetl.gatech.edu/students/tas/awards/2015
- and share your favorite TA's name and why you think they should win this award.