TSP-3510: Two-opt algorithm

Hyatt Bao, Evan Nelson

April 19, 2019

Our choice of algorithm for this project was two-opt with some random route creation in order to solve the local minima problem. Two-opt works by going through the given route, swapping the selected edge with all edges after it one by one, calculating the distance, and replacing the best route if the new route is better. We chose two-opt because it has a very simple implementation, and when compared against Genetic algorithms or greedy it was considerably faster and more effective at finding smaller distances consistently. Below, the pseudocode is described.

```
repeat until no improvement is made:
  startagain:
  bestdistance = calculateTotalDistance(existingroute)
  for ($i = 1; i<numberofnodeseligibletobeswapped - 1; i++$):
    for($k = i + 1; k<numberofnodeseligibletobeswapped; k++$):
      newroute = 2optSwap(existingroute, i, k)
      newdistance = calculateTotalDistance(newroute)
      if ($newdistance<bestdistance$):
        existingroute = newroute
        bestdistance = newdistance
        goto startagain
  main():
    parse given arguments into input txt, output txt, timelimit
    open input txt and read, line by line, the city id's into the graph
    compute the distances between each city and store it in the pairmatrix
    start initial route in sequential order - in the order of the ID's.
    distance = distance(graph, initial route, timelimit);
    while $time<timelimit - 10$:
      tempdist = distance
      temproute = initialroute
      randomly shuffle the temproute nodes
      newroute = twoOpt(graph, temproute, timelimit, timestarted);
      newdist = distance(graph, newroute, timelimit)
      if (newdist was computed):
        tempdist = newdist
      if $tempdist<distance$:
        swap best distance and best route with the new route and distance
    open output txt file
    write the cost/distance of the final route, followed by the ID's of the city followed in the
path.
```

One of the flaws of two-opt is its inability to get out of the local minima once it converges, as even if there is a lower minima it would likely not be able to find it if it only accepts better paths than it had before and the routes similar to it have a worse distance. As such, we elected to do a random shuffling of the paths to utilize more exploration while still maintaining our exploitation ratio; unlike simulated annealing, it will not accept worse routes, but the algorithm in main will run two-opt on randomly shuffled paths to see if there is a better minima than the one we currently have. Two opt itself does not know the previous best route, only the best in the region that it is currently searching, so we maintain an overall best route and distance outside of it and run two-opt on each of the randomly shuffled paths. In this way, it is possible to explore while still utilizing the quick property of two-opt to find an approximation to the optimal path, as it will only accept better paths than it found before, but it will restart at different points to explore the full breadth and avoid getting trapped in a certain minima.