HEURISTICS GUILD

Contest Write-up

Karthik Kashyap

EE23B030™

All of the codes referenced in this application can be found on my GitHub. The Greedy algorithm was mostly thought of and implemented by me, but for the Genetic Algorithm implementation, I have used the help of ChatGPT.

§2 Heuristics

5. *(Contest Write-up)* We will organise a two-day contest where each of you is expected to solve the question and send a write-up on your solution. Feel free to use any software to make the write-up, just make it clear and easy to read.

For the contest itself, I tried referring to a few papers I found related to The Node Weight Dependent Traveling Salesperson Problem, but in the end before I implemented this I ended up implementing a relatively greedy approach that I thought of based on scores of points. Description on the algorithm is after the codes Description.

```
'
        v[i] = v1;
    }
    cin >> s >> t1 >> t2;

    double gold_total = 0;
    for (auto i : v)
        gold_total += i;

    vector<bool> visited(n, false);
    vector<int> tour;

    int best_start = 0;
    double min_start_dist = 1e18;
    for (int i = 0; i < n; i++)
    {
        double d = euclidean(x[i], y[i], 0, 0) / v[i];
        if (d < min_start_dist)
        {
            min_start_dist = d;
            best_start = i;
        }
    }
    int current = best_start;
```

```
'
    visited[current] = true;
    tour.push_back(current);
    double time_elapsed = min_start_dist / s;

    while (tour.size() < n)
    {
        double best_score = -1e18;
        int next = -1;

        for (int i = 0; i < n; ++i)
        {
            if (!visited[i])
            {
                double dist = euclidean(x[current], y[current], x[i], y[i]);
                if (dist == 0)
                    dist = 1e-6;

                double score;
                if (time_elapsed <= t1)
                    score = (double)v[i] / dist;
                else if (time_elapsed <= t2)
                    score = (10.0 / dist + 1.0 * (double)v[i] / dist) / 11;
                else
                    score = 1 / dist;

                if (score > best_score)
                {
                    best_score = score;
                    next = i;
                }
            }
        }

        double dist = euclidean(x[current], y[current], x[next], y[next]);
        time_elapsed += dist / s;

        visited[next] = true;
        tour.push_back(next);
        current = next;
    }

    current = tour[0];
    double gold_acc = v[current];
    time_elapsed = euclidean(x[tour[0]], y[tour[0]], 0, 0) / s;

    for (int i = 1; i < n; i++)
```

Explanation of algorithm:

Start by choosing a point with the smallest $\frac{distance}{weight}$ value or highest value of $\frac{weight}{distance}$. This is done to ensure that the point is not too far away as well as has more amount of gold in it. Applying this for all the consequent unvisited points should yield a good solution.

There's one more catch to this: Time. Once time reaches $T_{realise}$, the nodes with the higher weights are not preferred as they are the ones to get evacuated. So we keep track of time after each node travel, and after the time reaches, $T_{realise}$, we instead choose points with highest value of $\frac{1}{distance}$ rather than $\frac{weight}{distance}$. So as to not completely choose nodes distance wise, I have a preference ratio of $10:1$ to each of these factors.

Changes from the contest final submission:

The contest submission was relatively messy, with several parts of the code not having any impact. I tried adding slight randomness in choosing the points like how it is done in Simulated Annealing, but then later on removed that aspect as this needs time to run, and $3s$ is not enough time to do this, hence went with this Greedy Approach. But I never removed the parts of the code. Also, the initial point was chosen based on just distance, not sure why I missed this. Changing this to take into account weight as well changed the score to 7^{th} position in the contest. Here is the refined code.

I agree, not a great algo. Thought of Simulated Annealing and Genetic Algorithm but as the time limit was just $3s$ and I wasn't sure about the constraints of the input, I decided to not implement them.

Later on, after the contest I tried implementing genetic algorithm into it. It runs well for the initial test cases but for the larger ones it gives TLE as expected with GA. The total score hence is low, but I wanted the approach to be out there. Here is the code for it. Also to mention, the code is not completely written by me. I have used the help of ChatGPT in adding the GA on top of the greedy approach that I initially had. I didn't want to do this, but I didn't want to spend too much time on this, I just wanted to see how the result would turn out.