The prize-collecting traveling salesman and related problems

Written by: Czirják Lilla
Supervisor: Dr. Király Tamás
December, 2024

1 Introduction

The Traveling Salesman Problem (TSP) is a cornerstone in the field of combinatorial optimization. The objective of this problem is to determine the shortest Hamiltonian cycle in a complete graph with n nodes, where the edges have metric weights. As one of the most well-known NP-hard problems, TSP has been studied extensively, not only due to its theoretical significance but also because of its practical applications in logistics, transportation, and network design.

Numerous versions of the Traveling Salesman Problem have been proposed. This essay focuses on the prize-collecting variant, where, alongside the metric edge weights, each vertex of the graph is assigned a positive weight representing the prize or penalty the salesman forfeits by skipping a city. In this version, the goal is not to visit all vertices, so we are not seeking a Hamiltonian cycle. Instead, we aim to find a cycle of arbitrary size that starts at a given root vertex, minimizing the sum of the tour length and the penalties paid for missed nodes.

In this semester I continued working on a heuristic algorithm for finding a solution to the problem. Moreover, I implemented the linear programming relaxation of the PCTSP, described below, in order to get a lower bound for the optimal value, and be able to evaluate the results of the heuristic algorithm.

2 The prize-collecting traveling salesman problem

Definition 1. *Be given a complete graph $G = (V,\ E)$, a root $r \in V$, $c_e \geq 0\ \forall e \in E$ metric edge lengths, and vertex weights $\pi_v \geq 0\ \forall v \in V \backslash \{r\}$.*
The prize-collecting traveling salesman problem is to find a cycle $C = (V_C,\ E_C)$ in G, so that $r \in V_C$, and $\sum_{e \in E_C} + \sum_{v \in V \backslash V_C}$ is minimal.

2.1 The LP relaxation

The linear programming relaxation of the problem can be formulated as follows:

$$min \sum_{e \in E} c_e x_e + \sum_{v \in V} \pi_v (1 - y_v)$$

$$
\begin{aligned}
x(\delta(v)) &= 2y_v \quad \forall v \in V \backslash \{r\} \\
x(\delta(r)) &\leq 2 \\
x(\delta(S)) &\geq 2y_v \quad \forall S \subseteq V \backslash \{r\},\ v \in S \\
y_r &= 1 \\
x_e &\geq 0 \quad \forall e \in E \\
y_v &\geq 0 \quad \forall v \in V
\end{aligned}
$$

Here for all $e \in E$, c_e is the length of the edge e, and for all $v \in V$, π_v is the amount of penalty that has to be paid for leaving out the vertex v. r stands for the root of the tour, which must be included in the solution.

Each node and edge has a corresponding variable in the relaxation (x_e for edge e, and y_v for node v), which represents the extent to which the node or edge is visited by the fractional solution.

Given a subset $S \subseteq V$, let $\delta(S) := \{e \in E : \ |e \cap S| = 1\}$, i.e. the set of edges exiting the node set S, and let $\delta(v) := \delta(\{v\})$. Besides, let $x(\delta(S)) \ := \ \sum_{e \in \delta(S)} x_e$.

We require the extent to which a vertex is visited to be equal to half of the sum of the visitation levels of the edges incident to that vertex. This follows naturally from the fact that in an integer solution, if a vertex is included in the tour, then there have to be exactly two edges incident to it that are included in the tour.

3 Implementation of the LP

I implemented the linear programming relaxation of the Prize-Collecting Traveling Salesman Problem, described above, in Python programming language, using the PuLP library and the built-in PULP-CBC-CMD solver.

However, the number of variables in the LP is quadratic, while the number of constraints is exponential in the size of the node set. Namely, the number of variables equals to $|V| + |E| = |V| + \frac{|V| \cdot (|V|-1)}{2}$, which makes the program grow large quickly when increasing the size of the graph. Besides, the linear program includes one or more constraints for every subset $S \subseteq V \backslash \{r\}$. More precisely, it contains a constraint for each (S, v) subset-node pairs, where $v \in S$. This makes the program too large to solve even for a small instance.

In my implementation, I first create the model leaving out all the set constraints of the form $x(\delta(S)) \geq 2y_v$, where $|S| \geq 2$. I fixed the number of nodes to be 100, which is large enough to provide evaluable results but still manageable to handle. After solving this model, I add the constraints for the sets where the condition is violated in the solution. In the case when the solution consists of a disjoint graph, I consider the connected components and add the corresponding constraints to the model, as there are no outgoing edges, thus the conditions for these sets must be violated. In case the solution is a connected graph, I look for the minimal cuts between the root vertex r and each of the other vertices v, and check whether the condition holds for the set containing v in the minimal cut. Then, I extend the LP model with these additional constraints if needed. I use the solver again

to solve the model, then I repeat this step, while there are no more constraints to add, if the program terminates in a reasonable amount of time. In practice, for some instances this is still unrealizable, but we still gain a lower bound for the optimal value.

4 The heuristic algorithm

Solving the Traveling Salesman Problem exactly, and so solving the Prize-Collecting TSP is computationally infeasible for large instances due to the factorial growth of possible solutions. However,a variety of approximation algorithms have been developed to tackle it.

Heuristic algorithms provide an essential approach in handling the PCTSP by offering near-optimal solutions in a reasonable amount of time. These methods, which do not guarantee an exact solution, focus on finding good-enough solutions quickly by making decisions based on a simplified search process.

In my implementation, I operate with the following algorithm.

Given an n-vertex complete graph with metric edge lengths, a positive weight assigned to each vertex, and a fixed root node, we begin with the root as a one-point tour. Vertices are then added one by one to the existing tour. In each iteration, the goal is to find the best possible insertion by considering all the remaining vertices. Specifically, we examine each outlying vertex and attempt to insert it between every pair of adjacent vertices in the current tour, selecting the insertion that results in the smallest objective value, but forcing the algorithm to increase the size of the tour by one node, even if it causes a momentarily worse solution. Then, we try to execute twice in a row the following edge-swapping, then the node-dropping step, in case they improve the current solution.

Edge-swapping step:
For each pair of edges $(v_1v_2,\ u_1u_2)$ in the given tour (where the edge v_1v_2 is passed through sooner than u_1u_2), we check whether the solution can be improved as follows. The edges v_1v_2 and u_1u_2 are deleted from the solution and replaced by the edges v_1u_1 and v_2u_2. The path $v_2 - u_1$ is replaced by the path $u_1 - v_2$ (walked through from the opposite direction). We repeat this until the objective value cannot be decreased by this method.

Node-dropping step:
For a given tour, we examine whether deleting one vertex can decrease the objective value, and if so, we delete the "worst" node from the solution, repeating this while it can make an improvement.

After these steps, we proceed in the current iteration with the tour gained before deleting any nodes, providing the termination of the algorithm in $n-1$ iterations, where n is the number of nodes, specifically, $n = 100$ in our case. We keep track of the best found tour, which is returned at the end of the algorithm.

5 Evaluation of the heuristic algorithm

5.1 Setting parameters

I implemented the algorithm on pseudorandom graphs generated from seeds using the NumPy Python package. Specifically, the number of nodes is fixed at $n = 100$, and each node is assigned a pseudorandom prize, which is an integer between 1 and 100. The edge lengths are required to satisfy the metric properties, so I use the ceiling of the Euclidean distances between vertices, which are placed within a bounded area. The coordinates of the vertices are generated uniformly at random within the range [0, field size], where the field size was set to 700.

5.2 Some experimental results

In the following data table (5.2) we can see a comparison of the result of the heuristic algorithm and the lower bound for the optimal solution found by solving the LP. It contains values for some instances of the problem with parameters described above.

The images 5.2 and 5.2 provide an example of the tours found by the two algorithms.

5.3 Codes

The codes are available at the following links:
Heuristic algorithm: https://colab.research.google.com/drive/1yovYIn38DYfjaRy6PLfgnwocDGbDg_w5?usp
LP relaxation: https://colab.research.google.com/drive/1niv_Pf0NYNi498QvWv9Bhy2GApyaP81c?usp=sha

seed	value of LP	value of heuristic	difference in values	solution size of LP	tour size of heuristic	difference in tour sizes
0	4564	4683	119	46	26	-20
1	4336	4664	328	52	63	11
2	4696	4925	229	42	1	-41
3	4295	4595	300	50	40	-10
4	4299,5	4580	280,5	64	70	6
5	4142	4660	518	50	51	1
6	4368	4711	343	61	3	-58
7	4568	4929	361	58	1	-57
8	4494	4658	164	45	52	7
9	4319	4493	174	64	43	-21
10	4198,75	4361	162,25	74	21	-53
11	4332,986	4578	245,014	70	29	-41
12	4553,348	5014	460,6518	94	55	-39
13	4267	4613	346	41	47	6
14	4740,5	5016	275,5	59	45	-14
15	4611,5	4879	267,5	69	66	-3
16	4091	4264	173	30	15	-15
17	4222	4380	158	40	21	-19
18	4277,875	4581	303,125	85	62	-23
19	4455	4760	305	59	29	-30
20	4334	4651	317	59	56	-3
21	4315	4451	136	60	16	-44
22	3957,652	4154	196,3485	77	29	-48
23	4482	4602	120	51	19	-32
24	4332	4887	555	59	1	-58
25	4765	5159	394	62	70	8
26	4386	4992	606	54	73	19
27	4541,5	4899	357,5	68	28	-40
28	4662	5089	427	57	77	20
29	4095	4397	302	71	37	-34

Figure 1: Table of values for field size=700

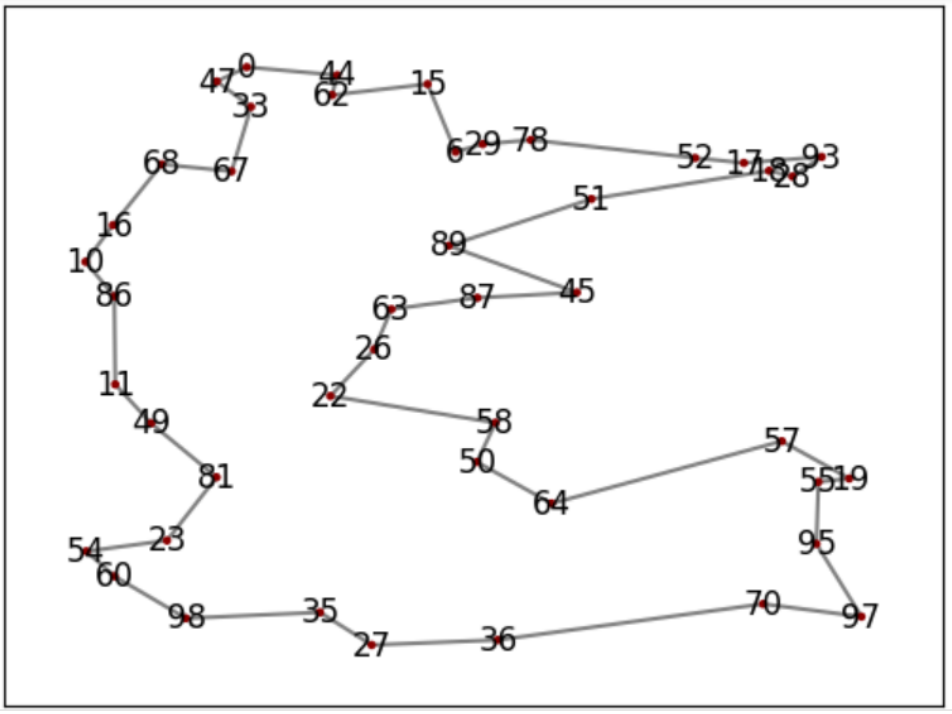

Figure 2: The tour found by the linear program for seed=8, field size=700

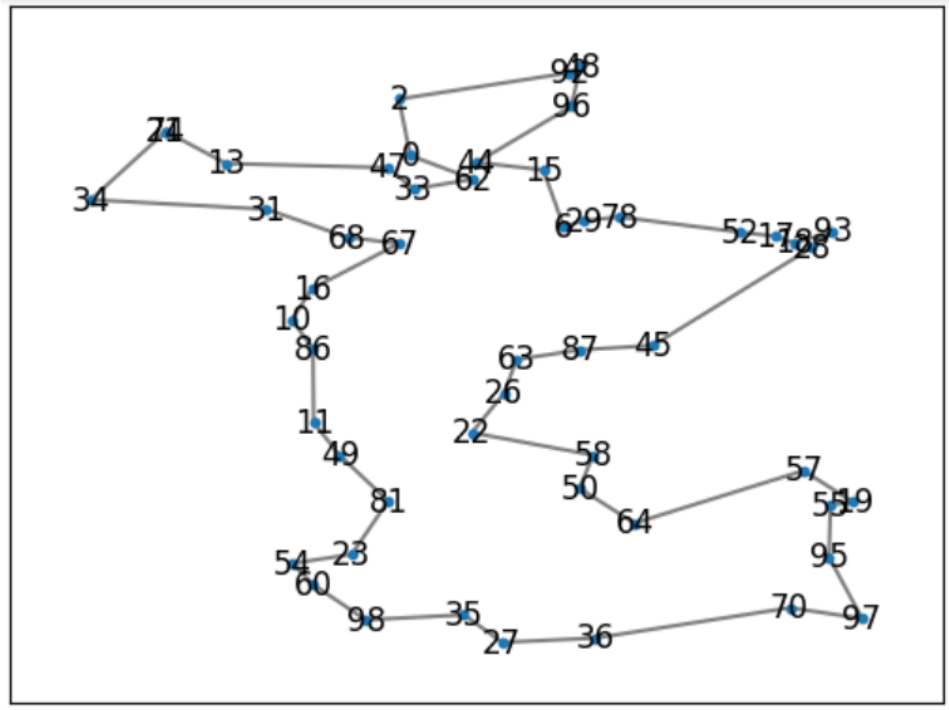

Figure 3: The tour found by the heuristic algorithm for seed=8, field size=700

References

[1] Blauth, Klein, Nägele: A Better-Than-1.6-Approximation for Prize-Collecting TSP (2023)

[2] Goemans, Williamson: A general approximation technique for constrained forest problems (1995)

[3] Ausiello, Bonifaci, Leonardi, Marchetti-Spaccamela: Prize-Collecting Traveling Salesman and Related Problems

[4] Ahamdi, Gholami, Hajiaghayi, Jabbarzade, Mahdavi: 2-Approximation for Prize-Collecting Steiner Forest