Shadaj Laddad

Parth Patel

Brian Yu

A Modified Single Commodity Flow Formulation for Solving Drive the TAs Home

The Single Commodity Flow(SCF) formulation is useful for modeling the Travelling Salesman problem(TSP) as a linear program. We found that it is possible to utilize elements of the SCF formulation to set up an accurate linear program to solve Drive the TAs Home(DTH) with optimal solutions for a vast array of inputs.

In the original formulation for TSP, we would model visiting the vertices by a flow, which is set to the number of vertices in the graph. This flow would begin as a single flow starting from the starting vertex. Excluding the starting vertex, our constraints would dictate that flow into each vertex equals the flow out plus one. This implies that at each vertex, one unit of flow must be dropped, or in other words it forces visitation of all vertices. To facilitate this constraint, it must be true that no subtours exist. Thus, we would add a constraint such that if flow exists across an edge, the travelling salesman must travel across that edge. Additionally, the number of times a vertex is entered equals the times a vertex is exited, which is set to 1. This ensures the salesman comes back to the starting vertex. With the objective function of minimizing the length of the tour the salesman takes, the optimal solution to this LP would yield a solution to the TSP.

Our algorithm was remarkably similar to the formulation above, with the same principles applying to DTH. The main difference is that instead of having the flow equal to the total number of vertices, it is equal to the number of TAs. Now, instead of having flow being subtracted by 1 at every vertex, we enforced the constraint that each unit of flow(a TA) must be dropped off exactly once at one vertex. Once again, excluding the starting vertex, the flow into any location is equal to flow out plus the number of units of flow(TAs) dropped at that location. Additionally, we modified the objective function to minimize the cost of the car's path plus the shortest path distances from the point at which a TA is dropped off to their home. This yielded optimal solutions for the vast majority of student inputs. We got even better results when the flow constraints were loosened to be non-integer values, which can be done without resulting in invalid results because the change in flow is always integer and the initial and final flows are required to be integers.

In the process of coming up with an algorithm, we tried some other approaches, primarily the Multi-Commodity Flow(MCF) formulation and the Miller-Tucker-Zemlin(MTZ) formulation. The main distinguishing feature of the MTZ is that it eliminates subtours by keeping track of the order in which vertices are visited rather than modeling flow. The MCF is very similar to the SCF in that it models using flow, with the main difference being we model each TA as its own separate flow, hence the name.

For computation we ran on the hive machines for approximately 36 hours and ran on Google Cloud Platform(GCP) for about 24 hours. We ended up using 50 dollars in free trial GCP credit. Thanks to our formulation, the vast majority of problems could be solved in a couple of

minutes on our own laptops, which enabled us to solve inputs without depending too heavily on external compute providers.

Overall, this project strengthened our knowledge on linear programming. It was interesting applying the theoretical background from lectures and discussions to a real life program in which we could see the magic of linear programming come together to solve an applicable problem on a large scale!