IND E 599 Integer Programming Project Report

Soheil Keshavarz

March 15, 2025

Abstract

This report describes the tasks done for IND E 599 course project [1] using the Gurobi solver in Python API. First, the codes will be explained, followed by different outputs, and in the end, we conclude with some remarks regarding the comparison between different formulations of the traveler salesman problem. To read the complete log of each solution as well as final optimal values (x_{ij}'s, u_i's, and f's), please refer to folder `logs` in the provided repository.

Task 1

Please refer to the provided code in `INDE599_project.ipynb` file for more details. First, we fix `r2` and `r3` as values of `r` in the second and third formulations. This value is the same for both formulations and all scenarios to maintain consistency between different scenarios. Then, the method below for reading data in a directory was written to avoid hard-coding anything in the code such as the data file name.

```
def read_data(dir='data'):
    data = dict()
    files = os.listdir(dir)  # Lists all files and folders
    # Filter only files (exclude directories)
    files = [f for f in files if os.path.isfile(os.path.join(dir, f))]
    for f in files:
        name = f.split('.')
        if name[1] == 'txt':
            name = name[0]
            # reading the file
            with open(os.path.join(dir, f), "r") as ff:
                lines = ff.readlines()

            # Extract number of nodes
            num_nodes = int(lines[0].strip())
            # Load distance matrix
            distance_matrix = np.loadtxt(lines[1:], dtype=int)
            data[name] = distance_matrix
    return data
```

Next, in a modular way, functions for each of the three formulations were implemented to be later called for different scenarios. For example, the code block below shows how Formulation 2 (MTZ) was coded. Note that by inputting `params` dictionary into `run_f2` function we can make changes to the parameters of the Gurobi model as a part of task 3.

```
def get_f2(data:np.array):
    n = data.shape[0]
    # similar to f0
    m = gp.Model('f2')
    x = m.addVars(n, n, vtype=GRB.BINARY, name='x')  # 1e
    m.addConstrs((gp.quicksum(x[i,j] for j in range(n))==1 for i in range(n)),
                 name='1b')
    m.addConstrs((gp.quicksum(x[i,j] for i in range(n))==1 for j in range(n)),
                 name='1c')
    m.addConstrs((x[i, i] == 0 for i in range(n)),
```

[1] The problem statement, codes, data, and outputs (logs) can be found in https://github.com/BlueSoheil99/UW_INDE599_project.

```
                 name='1d')
    # new constrains for f2
    u = m.addVars(n, vtype=GRB.INTEGER, lb=1, name='u')
    r = r2

    m.addConstrs((u[i]-u[j]+1<=n*(1-x[i,j]) for i in range(n) for j in range(n) if i
        != r and j != r), name='3a')
    m.addConstr(u[r] == 1, name='3b')
    m.addConstrs((u[i]<=n for i in range(n) if i != r), name='3c1')
    m.addConstrs((u[i]>=2 for i in range(n) if i != r), name='3c2')  # added lower
        bound when defining 'u', do I need this now?

    m.setObjective(gp.quicksum(data[i,j]*x[i,j]
                               for i in range(n)
                               for j in range(n)),
                   GRB.MINIMIZE)
    return m


def run_f2(logfileName, params, datasets):
    for name,data in datasets.items():
        with gp.Env() as env:
            with gp.Model(env=env) as model2:
                try:
                    m2 = get_f2(data)
                    m2.setParam("LogFile", f'logs/{logfileName}_{name}.log')
                    for key, value in params.items():
                        m2.setParam(key, value)
                    m2.optimize()
                    print_results(m2)
                except gp.GurobiError as e:
                    print('Error code ' + str(e.errno) + ': ' + str(e))
                except AttributeError:
                    print('Encountered an attribute error')
```

Task 2

Table 1 shows the results of running Formulation 0 on the provided four instances. All objective values from this formulation are lower than the benchmark objected values provided by the instructor which is due to the relaxation in this formulation and neglecting subtours.

	bays29	dantzig42	pr76	rat99
Solution time (s)	0.02	0.01	0.03	0.04
Objective value	1764	532	77119	1089

Table 1: Output results of Formulation 0.

Task 3

To run task 3, we were asked to work with some of Gurobi parameters such as `MIPGap`, `TimeLimit`, `Presolve`, and `Cuts`. The results for these runs can be seen in tables 2 to 5.

By running the code and comparing the results of Table 2, which shows results with default settings, with the provided optimized objective values, we see that both formulations 2 and 3 are implemented correctly. Therefore, we will not show objective values in later tables.

	bays29		dantzig42		pr76		rat99	
	F2	F3	F2	F3	F2	F3	F2	F3
Solution time (s)	0.37	2.21	2.44	14.85	255.61	1214.45	15.41	1107.07
Objective value	2020	2020	699	699	108159	108159	1211	1211
Nodes explored	756	1	4946	1	155739	11133	5293	31
Cuts added	124	46	212	1	1794	60	558	5
Root LP relaxation bound	1774.3	2013.5	535.5	697.0	77663.4	105120	1090.9	1206.0

Table 2: Output results of formulations 2 and 3 with default settings (task 3a).

	bays29		dantzig42		pr76		rat99	
	F2	F3	F2	F3	F2	F3	F2	F3
Solution time (s)	1.14	5.27	24.18	45.97	551.43	-	92.28	1053.44
Best UB	-	-	-	-	-	105898	-	-
Best LB	-	-	-	-	-	105218	-	-
Nodes explored	1110	1	22621	12	116495	34	4055	22
Cuts added	312	1	892	8	3952	8	563	3
Root LP relaxation bound	1774.0	2013.5	535.4	697.0	77656.2	105120	1090.9	1206.0

Table 3: Output results of formulations 2 and 3 with `Presolve` turned off (task 3b).

	bays29		dantzig42		pr76		rat99	
	F2	F3	F2	F3	F2	F3	F2	F3
Solution time (s)	1.91	8.34	243.55	28.85	-	-	-	714.98
Best UB	-	-	-	-	110477	109910	1211	-
Best LB	-	-	-	-	91096	105219	1179	-
Nodes explored	5533	1	267589	1	1122100	73	752416	44
Cuts added	-	-	-	-	-	-	-	-
Root LP relaxation bound	1774.4	2013.5	535.4	697.0	77663.4	105120	1090.9	1206.0

Table 4: Output results of formulations 2 and 3 with `Cuts` turned off (task 3c).

	bays29		dantzig42		pr76		rat99	
	F2	F3	F2	F3	F2	F3	F2	F3
Solution time (s)	5.21	5.86	240.16	34.92	-	-	-	665.26
Best UB	-	-	-	-	108159	108467	1211	-
Best LB	-	-	-	-	95802	105231	1178	-
Nodes explored	8642	20	347530	25	863212	33	547986	93
Cuts added	-	-	-	-	-	-	-	-
Root LP relaxation bound	1774.0	2013.5	535.4	697.0	77656.2.4	105120	1090.9	1206.0

Table 5: Output results of formulations 2 and 3 with both `Presolve Cuts` turned off (task 3d).

Task 4

4-a

Based on the project description on the strength of MCF formulation versus the cut formulation and also the big-M notation in the MTZ formulation, it was hypothesised that formulation 3 is stronger than formulation 2. This was tested in task 3 of this project, and we see that the MCF formulation tends to find the optimal solution by exploring and adding much fewer nodes and cuts which indicates a relatively tighter search space. Moreover, based on table 5 where the formulations are compared without presolving and cuts to the x-space, we see a better performance from formulation 3 regarding solution time or upper/lower bounds , although having time complexity of $\Omega(n^3)$.

4-b

By comparing the results of formulations 3 and 2 in all settings, we can see that the stronger formulation (3) drastically affects the number of explored nodes. In this regard, the number of explored nodes and added cuts in F3 is always much smaller than these numbers in F2 for the same network which indicates tightness of the solution set in F3.

4-c

Since the stronger formulation has a three dimensional variable (f_{ij}^v) and requires a time complexity of $\Omega(n^3)$, it will perform worse than a formulation with the same solution set. In tables 2 and 3, formulation 3 is outperformed by formulation 2 which was running on a near-optimal solution set with the help of additional cuts.

4-d

Due to randomness in each time solving an optimization problem, it is hard to directly compare tables 2 and 3 to analyze the effect of `Presolve` parameter. For example, we can see that `rat99` network has a shorter solution time when `Presolve` is off which should not be the case if in both cases exact same parameters were used. For a better comparison, we should have fixed the random seed in our model. However, due to time limitation, this was not done.

Nevertheless, `presolve` reduces the number of cuts added and simplifies the problem based on the definition of conditions and constraints. While this might not change the complexity of a model, it can hugely impact the computation needed for an optimization problem, for example by reducing the dimension of the search space (fixing variables), if we do not define constraints of a problem efficiently.

4-e

The same rationale from the previous part can be applied here where results from table 2, 3, and 4 need to be compared carefully.

First, eliminating cuts seems to have a higher impact on the performance versus presolve in most cases as it increases the solution time or causes time limit error. Second, it has a higher impact on formulation 2 as formulation 3 has a tighter search space and is not as dependent on additional cuts as MTZ formulation. Additionally, we can see that in table 2, unlike table 4, formulation 2 outperforms formulation 3 which clearly shows how good cuts can help us overcome the weakness of our formulations when solving a problem.

4-f

Table 6 shows the results from Concorde benchmark which used additional cuts. For the sake of comparison, we compare the default results of our implemented algorithms in table 3 with table 6. From this comparison, we can clearly see the superiority of Concorde where its linear complexity and tight space come in advantage.

	bays29	dantzig42	pr76	rat99
Running time (s)	0.04	0.09	0.60	0.40
search nodes	1	1	1	1

Table 6: Concorde benchmarks for given networks.