# Assignment - Integer Linear Programming (BDS)

M. Leitner

May 23, 2025

In this assignment, you will consider a given combinatorial optimization problem (COP) and develop and (empirically) compare different (mixed) integer linear programming formulations for it. Depending on your own level and confidence (and how much you want to challenge yourself) you can either work on the single-vehicle variant or the multi-vehicle variant which is slightly more challenging. The following section first defines the single-vehicle variant as well as the extensions to the multi-vehicle case.

## Problem definition

Both problem variants consider a complete (directed) graph $G = (V, A)$ with node set $V$ and arc set $A$. The node set $V = \{1, \dots, n\}$ is partitioned into the depot node 1 and the remaining set of customer nodes $N = V \setminus \{1\}$. Customer nodes are grouped into $K$ subsets $S_1, \dots, S_K$ called clusters such that $\bigcup_{k=1}^{K} S_k = N$. Note that these subsets are not necessarily disjoint, i.e., a node may be contained in more than one cluster. A non-negative score $p_k$ is associated to every cluster $k = 1, \dots, K$ which is collected only if all the customers of the cluster $S_k$ are visited (not necessarily sequentially). Parameter $t_{ij} \geq 0$ indicates the time required to traverse arc $(i, j) \in A$, i.e., to travel from $i$ to $j$. These travel times satisfy the triangle inequality.

**Single-vehicle variant.** The goal of the single-vehicle variant is to identify a tour starting and ending at the depot 1 that maximizes the total score obtained from visiting customer nodes. The total travel time of the tour may not exceed the maximum travel time $t_{\max}$.

**Multi-vehicle variant.** In the multi-vehicle variant, $m \in \mathbb{N}$ identical vehicles are available at the depot 1. The goal is to identify a set of at most $m$ tours each starting at the depot 1 that maximize the total score obtained from visiting all nodes of clusters. Notice that the score of a cluster is also collected if its customers are visited by different vehicles. Furthermore, none of the paths may exceed the maximum travel time $t_{\max}$.

## Your Tasks

1. Formulate the optimization problem you choose to work on as a (mixed) integer linear programming problem. Thereby, develop at least two versions of your formulation where one is using the idea of Miller-Tucker-Zemlin (MTZ) constraints and the other one is flow-based. To obtain full credits, you have to derive, implement, and test at least three versions (MTZ, single-commodity flow, multi-commodity flow).
2. If you decide to work on the multi-vehicle variant, try to avoid the use of any variables that contain a vehicle-index in your MTZ and single-commodity flow formulations. It can also be interesting to compare such a formulation with a corresponding one including vehicle-indexed variables (which you will likely find easier to model).

3. Implement your formulations. While I recommend to either use Julia/JuMP or Python/Gurobipy in combination with Gurobi, you can also use a different tool / programming language assuming that you will be able to explain your formulations and their implementation to me. You also need to be able to run your code during the oral exam on request (e.g., in case I do have additional questions or doubts).

4. Solve your formulations for (some of) the instances provided (which are based on the TSPlib) and for each formulation. Some instances / formulations may take too long in which case you should report the results after a chosen timelimit of, e.g., 15 minutes). It is very likely that not all benchmark instances can be solved (optimally) by all formulations. Nevertheless, analyzing and reporting optimality gaps after reaching the time limit provides important insights also for these instances.

5. The instances provided are for the single-vehicle variant. Thus, in case you work on the multi-vehicle variant, you need to choose (and vary) the number of vehicles and also experiment with different values for $t_{\max}$. I recommend to start with $m = 1$ and then experiment with different values of $m$ and $t_{\max}$, e.g., according to a (meaningful) rule that you identify by performing preliminary experiments. Report results for these different values in your report (see below) including information about the number of used vehicles in the optimal solution. In general, it can be interesting to experiment with different values of $t_{\max}$ also for a given number of vehicles (or when working on the single-vehicle variant) to analyze whether its value impacts the performance of your formulations.

6. Create a short report including information about the problem variant you did choose, your formulations including a descriptions of them, and result tables comparing the results obtained by them (solution values, gaps after reaching the time limit, runtimes). Other observations as well as a short interpretation of them are of course welcome too.

It can always be interesting and beneficial to think about variants of your formulations or enhancements by means of additional, strengthening inequalities. If you consider such additions, it is a good idea to analyze their impact (which can be done empirically and/or theoretically).

**Deadline**

- Hand in your implementation and report on Canvas until **Friday, June 27 (end of day)**.

**Remarks and Hints**

- This assignment as well as the instances are used for the first time. Thus, ask / inform me if something seems weird / wrong.
- First design models, then implement!
- Check if final report exactly includes the implemented formulations!
- Do not use non-linear constraints (even if some are possible in CPLEX/Gurobi)!
- Some people find it easier to model tours starting and ending at depot 1 as paths from a starting depot 1 to an end-depot $n + 1$. This is of course allowed, but make sure to appropriately extend the instances (in your code) and report your model accordingly!

- Performing a check whether a solution is feasible is very important (to find mistakes). Thus, I strongly recommend to extract the solution (i.e., variable) values obtained from the ILP solver and create a code that re-creates the corresponding path(s) and checks whether they are indeed feasible.
- Your final grade will depend on the correctness (and quality) of the formulations developed and implemented and (the clarity of) your report.

### Format of test instances

The test instances start with a line providing basic information about the number of nodes ($|V|$) , the number of clusters ($K$) and the value for $t_{\max}$. The second line contains the string "NODE_COORD_SECTION" which is followed by one line for each node each consisting of three (integer) values defining the node number (between 1 and $|V|$), the X and Y-coordinate of that node. After specifying all nodes a line containing the string "SET_SECTION" follows. Subsequently, there are $K$ lines each containing the cluster number (i.e., an integer between 1 and $K$), the score of this cluster, and the list of nodes contained in this cluster. The (symmetric) travel time $t_{ij}$ between two nodes $i$ and $j$ is defined as the Euclidean distance between them rounded up to the next integer, i.e., equal to $t_{ij} = \lceil \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \rceil$ if the $x_\ell$ and $y_\ell$ are the x- and y-coordinates of node $\ell \in V$, respectively.

**Example:**

```
5 2 10
NODE_COORD_SECTION
1 1 5
2 2 3
3 4 4
4 2 2
5 6 4
SET_SECTION
1 10 2 3 5
2 8 2 3 4
```

In this example, an instance with 5 nodes, 2 clusters and a maximum travel time of 10 is defined in which cluster 1 has a score of 10 and contains nodes 2, 3, and 5 while cluster 2 has a score of 8 and contains nodes 2, 3 and 4.