SDP for SRFLP

$$\min \quad K - \sum_{i<j} \frac{c_{ij}}{2} \left[\sum_{k<i} l_k X_{ki,kj} - \sum_{i<k<j} l_k X_{ik,kj} + \sum_{i<k} l_k X_{ik,jk} \right] \tag{1}$$

$$s.t. \quad X_{ij,jk} - X_{ij,ik} - X_{ik,jk} = -1 \quad \text{for all triples} i<j<k \tag{2}$$

$$diag(X) = e \tag{3}$$

$$rank(X) = 1 \tag{4}$$

$$X \succeq 0 \tag{5}$$

The constraints (3) is the equivalent to $X_{ij,ij} \in \{-1, 1\}$. The constraint (4) ensures that the matrix X has rank 1, which is part of the optimality condition for SDP. The constraint (5) ensures that the matrix X is positive semidefinite.

Solving the indexing problem

Assume we want the first part of the index of a matrix X for a pair (a, b) and lets assume all pairs (a, b) where $a < b$ are ordered the following:

$\{(1,2);(1,3);(1,4);\ldots;(1,n);(2,3);(2,4)\ldots(2,n);\ldots;(n-1,n)\}$

Then with help of the Gauss summation we get the following equation:

$$n(a-1) - \frac{a^2+a}{2} + (b-a) - 1 \tag{6}$$

To obtain the index starting with zero.

$$\begin{bmatrix} 1 & y_{1,2} & \cdots & y_{1,n} \\ y_{1,2} & y_{1,2}y_{1,2} & \cdots & y_{1,2}y_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ y_{1,n} & y_{1,n}y_{1,2} & \cdots & y_{1,n}y_{1,n} \end{bmatrix}$$

Abstract

The Single Row Facility Location Problem (SRFLP) is a combinatorial optimization problem that arises in the context of locating facilities in a network. The problem is to find the optimal location of a single facility on a line network, such that the total transportation cost is minimized. For this problem semi definite programming relaxations have proven to be effective. In this paper we build on this approach and apply it to the Multi-Objective Single Row Facility Location Problem (MOSRFLP). We present a semi definite programming framework with for the MOSRFLP and compare it to state-of-the-art integer programming formulation used in literature for the single objective SRFLP.

SDP Branch and Bound

I should start by figuring out how to do the branch and bound for the SDP as for the multi objective problem we might use lexico-graphical ordering. This would make it a lot easier to implement.

At the moment I can do the SDP relaxation. For the branch and bound I would need to implement a heuristic to get a feasible solution. Furthermore, I would need to think of an efficient way to branch and to prune and especially how to store the branching tree.

In other papers, there are five phases. The Preprocessing, the bounding, the prunding, the reduction and the branching.

The Preprocessing might not be necessary in our case, because we our problem cannot be not feasible, as there is always some way to order the facilities.

The Bounding is basically the heuristic to get a feasible solution and a upper bound on the problem. The better the heuristic the better the bound.

In the pruning phase, we check if the current solution is better than the best solution found so far. If not we prune the branch. For this check again all the reasons for pruning. (Hint: Write a function, which gets a b and B tree and a node and checks if the node can be pruned or not)

For the branching, the question is which variable to choose from. I would at first recommend to not choose variables where the value is already fixed. Need to read further literature on this because maybe putting some facilities in the middle or at the end might be a good idea.

Additionally I need to transform the solution into a permutation such that it make sense. Maybe already translate the relaxation into a permutation, to find some patterns.

Branch and bound

- Solve the relaxation of the problem
  - Check if the relaxation is infeasible
  - Check if the relaxation is integer (rank 1)
  - If none of the above, find the element with smallest absolute value in the first row and branch on it. (-1, +1)