Applied Operations Research - Fixed Charge Network Flow

Anders G. Kristensen & Karl P. Nørgaard

2023-10-24

In this project we consider the Fixed Charge Network Flow Problem (FCNFP). More specifically, we consider and compare different methods of solving the problem for different sizes of instances.

Formulation and initial solution

Consider the following formulation of the FCNFP:
Let V, A, be set of vertices and arcs for a directed graph. Define the root, $r \in V$ and let the demand $b_r < 0$ for the root node. Now, for all other nodes we have $b_v \geq 0$ as well as unit flow costs c_{ij} for each arc $(ij) \in A$, and fixed costs q_{ij} for the use of arc $(i,j) \in A$. The problem is to find feasible flow that minimizes the sum of all the flows and fixed costs. Define y_{ij} as the flow in arc (ij) and x_{ij} be an indicator of whether arc (i,j) is used. Let $V^+(i) = \{j : (i,j) \in A\}$, $V^-(i) = \{j : (j,i) \in A\}$. Without loss of generality we always assign the root to the root with index 1. Using the data generation class `FCNFP`, we can readily construct problem instances and represent them visually with the aid of the `NetworkX` library [3].

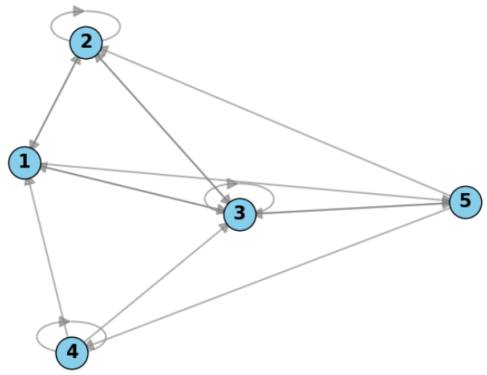

Figure 1: An instance of the FCNFP where $|V| = 5$.

Considering the problem's characteristics, an initial mathematical formulation of the FCNFP is given by:

$$\min \sum_{(i,j)\in A} c_{ij}y_{ij} + q_{ij}x_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j \in V^-(i)} y_{ji} - \sum_{j \in V^+(i)} y_{ij} = b_i \qquad i \in V \tag{2}$$

$$y_{ij} \leq |b_r| x_{ij} \qquad (i,j) \in A \tag{3}$$

$$y_{ij} \geq 0, x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{4}$$

We implement a solver for the the problem using the `Gurobi` API in Python [2]. This is done in the class `FCNFPSolver` located in the attached `fixed_charge_network_flow.ipynb`; the class has methods that invokes Gurobi's built-in solver method for the entire problem directly.

Alternative formulation of the problem

The aforementioned FCNFP can be formulated otherwise. A popular alternative formulation is:
Let $T = \{v \in V | b_v > 0\}$ be the set of terminals - that is vertices with a positive demand. One can then treat the flow to a specific destination $t \in T$ as a distinct commodity and define w_{ij}^t as the flow along arc (i,j) destined for $t \in T$. Such a reinterpretation is motivated by a desire to enforce stricter bounds on the decision variables x_{ij}, which in the previous formulation is downwards bounded by $x_{ij} \geq \frac{y_{ij}}{|b_r|}$ whereas in this formulation, the bound on x_{ij} is $x_{ij} \geq \max_{t \in T} \frac{w_{ij}^t}{b_t}$. The bound is weakly sharper and in most cases significantly more computationally sound. In this setting the FCNFP can be be formulated as:

$$\min \sum_{(i,j) \in A} c_{ij} \sum_{t \in T} w_{ij}^t + q_{ij} x_{ij} \tag{5}$$

$$\text{s.t.} \sum_{j \in V^-(r)} -w_{jr}^t \sum_{j \in V^+(r)} w_{rj}^t = -b_t \qquad t \in T \tag{6}$$

$$\sum_{j \in V^-(i)} -w_{ji}^t \sum_{j \in V^+(i)} w_{ij}^t = 0 \qquad i \in V \backslash \{r\}, t \in T, t \neq i \tag{7}$$

$$\sum_{j \in V^-(r)} -w_{jt}^t \sum_{j \in V^+(r)} w_{tj}^t = b_t \qquad t \in T \tag{8}$$

$$w_{ij}^t \leq b_t x_{ij} \qquad (i,j) \in A, t \in T \tag{9}$$

$$w_{ij}^t \geq 0, x_{ij} \in \{0,1\} \tag{10}$$

which can also be solved using the `Gurobi` API.

Benders' Decomposistion

Benders' decomposition is yet another way of reformulating an existing problem with the intention of making complicating variables less problematic in practical applications by splitting the problem into stages. In very broad terms, Benders' Decomposition Algorithm works by initially fixing the complicating variables and then solving the remaining problem

repeatedly, checking for feasibility and optimality as it goes along. The found solution is then optimal for the initial problem. To illustrate how the algorithm works, consider a generic problem of the form:

$$\begin{aligned}
&\min c^T x + h^T y && h \in \mathbb{R}^p, c \in \mathbb{R}^n && (11)\\
&\text{s.t. } Ax = b && A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m && (12)\\
&\quad Gx + Hy = d && G \in \mathbb{R}^{m_2 \times n} H \in \mathbb{R}^{m_2 \times p}, d \in \mathbb{R}^{m_2} && (13)\\
&\quad x \in \mathbb{R}^n_+, y \in \mathbb{R}^p_+ && && (14)
\end{aligned}$$

and assume that x is a complicating variable. Recall that the core principle of the algorithm is to fix the complicating variables and then solving for the less complicated variables. Define $Q(x) = \min_{y \in \mathbb{R}^p_+} \{h^t y | Hy = d - Gx\}$, that is a function to optimize the easy part of the problem for a given set of complicated variables. To ensure feasibility, that is a choice of x for which $Q(x)$ is well-defined, we introduce the set $K_x = \{x | \exists y \in \mathbb{R}^p_+ : Hy = d - Gx\}$, i.e. the set of x for which there exists a y such that $Q(x)$ is defined - the set of x for which there exists a feasible solution to the initial problem. Rewriting equations 11-14 keeping the recently introduced definitions in mind yields:

$$\begin{aligned}
&\min c^T x + \varphi && (15)\\
&\textbf{s.t. } Ax = b && (16)\\
&\quad x \in K_x && (17)\\
&\quad x \in \mathbb{R}^n_+ && (18)\\
&\quad \varphi \geq Q(x) && (19)
\end{aligned}$$

Where 17 ensures feasibility and 19 and 15 in combination constitute the optimality condition $\varphi = Q(x)$. Define the master problem (MP) as:

$$\begin{aligned}
&\min c^T x + \varphi && (20)\\
&\textbf{s.t. } Ax = b && (21)\\
&\quad x \in \mathbb{R}^p_+ && (22)
\end{aligned}$$

As explained earlier, the strategy is then to solve the MP iteratively, whilst checking for feasibility and optimality. In the simplest of terms, the algorithms solves the master problem, then checks whether $x \in K_x$, then checks whether $\varphi = Q(x)$, if both of these conditions are met the problem is solved, if either of the two conditions are not met, a constraint is added to the MP: A feasibility- or optimality cut respectively. In order to add these cuts in practice, an analytical expression of Q and K_x is required. For that purpose, define

$$F(x) = \min_{w^+, w^-, y} \{\mathbf{1}^T w^+ + \mathbf{1}^T w^- | Hy + w^+ - w^- = d - Gx, w^+, w^- \in \mathbb{R}^{m_2}_+, y \in \mathbb{R}^p_+\}.$$

If $F(x) > 0$ then $Q(x)$ is not well-defined, i.e. $x \notin K_x$. Let $F^D(x)$ denote the associated dual problem. Strong duality yields that $x \notin K_x \Rightarrow 0 < F^D(x)$. It can also be proven that

inserting a different x into $Q(x)$ does not change the feasibility of $Q(x)$ implying that for fixed x the *feasibilitycut* $F^D(x) \leq 0$ can be added to the MP. Optimality cuts are added in a similar fashion: Let $Q^D(x)$ denote the dual of $Q(x)$. It can be shown for some pair (x, φ) that $\varphi < Q(x) \Rightarrow \varphi < Q^D(x)$. In that case the *optimalitycut* $\varphi \geq Q^D(x)$ is added to the master problem.

Applying Benders' Decomposistion

In this section, we briefly illustrate how applying Benders' Decomposition can be done in the context of the FCNFP. The exact formulations of the master problem and the feasibility - and optimality subproblems naturally depend on which initial formulation is being considered. We limit ourselves to the first formulation as the steps in formulating the MP and the subproblems are practically identical and are easily translated to the alternative formulation. Recall the formulation 1-4 of the FCNFP

$$\min \sum_{(i,j)\in A} c_{ij}y_{ij} + q_{ij}x_{ij} \tag{23}$$

$$\textbf{s.t.} \sum_{j\in V^-(i)} y_{ji} - \sum_{j\in V^+(i)} y_{ij} = b_i \qquad i \in V \tag{24}$$

$$y_{ij} \leq |b_r|x_{ij} \qquad (i,j) \in A \tag{25}$$

$$y_{ij} \geq 0, x_{ij} \in \{0,1\} \qquad (i,j) \in A \tag{26}$$

Here, we consider the problem of choosing the arcs to open to be the complicated one. The MP then becomes:

$$\min \sum_{(i,j)\in A} q_{ij}x_{ij} + \varphi \tag{27}$$

$$\textbf{s.t.} x_{ij} \in \{0,1\} \tag{28}$$

The feasibility subproblem becomes:

$$\min v_1 + v_2 \tag{29}$$

$$\textbf{s.t.} \sum_{j\in V^-(i)} y_{ji} + v_{1_{ji}} - (\sum_{j\in V^+(i)} y_{ij} + v_{2_{ij}}) = b_i \qquad i \in V \tag{30}$$

$$y_{ij} - v_{2_{ij}} \leq x_{ij}|b_r| \qquad (i,j) \in A \tag{31}$$

We then retrieve the objective value along with the duals for each constraint, here denoted by d_{30} and d_{31} respectively. The duals can be obtained using Gurobi's built-in `.Pi`-function. The feasibility cuts are then added to the MP as needed, that is; we check if the objective value of the feasibility subproblem is equal to zero, if not we add the constraint

$$\sum_{i\in V} d_{30}b_i + \sum_{(i,j)\in A} d_{31}x_{ij}|b_r| \leq 0$$

to the MP. Similarly for the optimality subproblem we have:

$$\min \sum_{(i,j)\in A} y_{ij}c_{ij} \tag{32}$$

$$\textbf{s.t.} \sum_{j\in V^-(i)} y_{ji} - (\sum_{j\in V^+(i)} y_{ij}) = b_i \qquad i \in V \tag{33}$$

$$y_{ij} \leq x_{ij}|b_r| \qquad (i,j) \in A \tag{34}$$

$$\tag{35}$$

Once again retrieving the objective value and the duals for each constraint here denoted by d_{33} and d_{34} respectively. We then check if the objective value exceeds $\varphi + \varepsilon$ for some, relatively small, $\varepsilon > 0$. If that is the case we add the constraint

$$\sum_{i\in V} d_{33}b_i + \sum_{(i,j)\in A} d_{34}x_{ij}|b_r| \leq \varphi$$

to the MP. If neither feasibility nor optimality is violated, the problem is solved and the values for each of the choice variables are obtained along with the value of the objective function.

Logical constraints

As we show later the Benders implementation performs poorly even for instances with a small number of *n_nodes*. To bolster the Benders' methods we utilize various logical constraints to try and simplify the problem. In this, we also consider implications of the way the data is generated in the provided `FCNFP` class. To be exact we use that $c_{ij} \in (20, 40)$, $q_{ij} \in (200, 400)$ for $(i, j) \in A$, along with $b_t \in \{50, 51, \ldots, 75\}$, for $t \in V\backslash\{1\}$. As motivation briefly consider figure 1. Clearly, there are numerous arcs in this graph that are redundant in the problem. Due to their inherent characteristics, we can a priori identify arcs that will not be activated; these can be disregarded without changing the objective. Yet, the solver does not know this, so to optimize computational efficiency we implement a method in the `FCNFP` class, `reduce_data`. This method eliminates arcs based on the reasoning in this section and eventually reduces the dimension of the problem significantly for a fixed size of *n_nodes*.

In addition, several constraints can be added to the master problem in advance to minimize the number of feasibility- and optimality cuts. Note that the reasoning in the reduction of the problem is universal, whereas some constraint from one formulation might not be applicable in the other.

Reducing the problem

Naturally, since any flow- and fixed cost is non-negative, there is no reason to consider arcs of type $(i, i) \in A, i \in V$. Additionally, the root node is by construction the only non-terminal node. Thus we needn't examine arcs going to the root; that is arcs of type

$(i, 1) \in A, i \in V$. Also, rather simple calculations can show that due to the lower bound on the demand of 50, we should always open up a flow in arcs directly from the root to a directly connected node, provided the flow cost between them is less than 36. As a direct consequence, we can overlook any arcs leading into such nodes apart from the arc from the root. Another implication of the way the data is generated is that if a path of length m from the root to a node exists, then we do not need to consider paths of lengths strictly longer than $m + 1$ that share the first $m - 1$ arcs. This is easily seen by direct comparison of worst- and best case in terms of flow- and fixed cost between two such paths of length m and $m + 2$ respectively. In particular, when a node is connected to the root by an arc directly, then any other arc going into that node from nodes not directly connected to the root themselves, can be disregarded.
To further improve our reduction, we focus on nodes connected directly to the root, where one or both is connected to the other. Analyzing the flow for such nodes $i \in V$ and $j \in V$ that without loss of generality is connected by the arc $(i, j) \in A$ it is clear that if

$$c_{i1} + c_{ij} > \frac{q_{1j} - q_{ij}}{b_j} + c_{1j} \tag{36}$$

then we can disregard arc (i, j). The network in 1 can be reduced to:

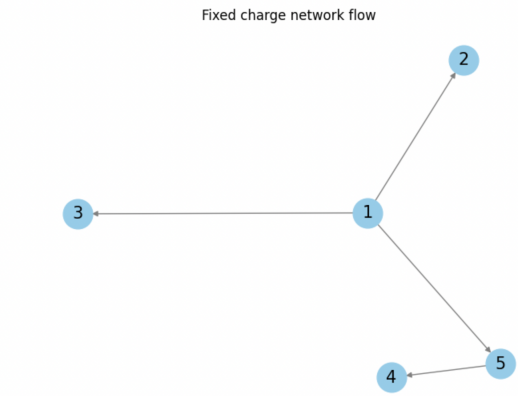

Figure 2: Figure 1 after applying data reduction.

Note that the extend to which a given problem can be reduced is clearly highly dependent on the parameter values for that specific problem. It is however independent of the formulation of the problem.

Adding constraints to the master problem

Providing the master problem with additional constraints can drastically reduce the number of required cuts. Though to begin with we introduce and compare constraints aimed at improving the optimality cuts by introducing a necessary lower bound on cost related to the flow between nodes, φ. A trivial such bound is the flow cost from the arc with the lowest flow cost among all arcs times the total demand. However, in most cases this bound is of course not very tight, it is not difficult to find a better bound: First, all flow has to

go through some arc from the root to a node connected directly to the root. Regardless of how this is done, the cost related to the flow of this is lower-bounded by $\min(c_{1j})|b_r|$ for $j \in \{v \in V | (1, v) \in A\}$, or the previous bound. Yet, for many cases we have nodes that are not directly connected to the root. The total cost related to the flow into these notes are analogously $\min(c_{ij})|b_l|$ for $i \in \{v \in V | (1, v) \in A\}$ and $j \in \{v \in V | (1, v) \notin A\}$. Where b_l is the total demand for notes are distance two to the root. Though, this strengthened method misses out on a contribution from the nodes with shortest distance to the root larger than 2, these are by construction quite rare and constitute a relatively insignificant contribution.

This bound in turn can be improved even further. As the demand for any non-root node always is strictly positive, we must send flow into a node through some path of arcs. For each node the cheapest way this can be done, considering all possible paths from the root to the node in question, multiplied with the demand in that node is a lower bound for the cost related to the flow into that node. Finding the lowest flow cost is done with the help of the implementation of Dijkstra's algorithm in `Networkz`[3]. Summing the contribution from all nodes then yields our best initial lower bound on φ. This is our proposed method; and it would, in fact, yield the solution, were it not for the part the fixed cost plays in the network. We illustrate the quality of the lower bound by generating 50 instances for values $n\_nodes \in \{15, 30, 50, 75, 100, 125, 150\}$ compute the lower bound and get the optimal objective with `FCNFPSolver` and then calculate their proportion

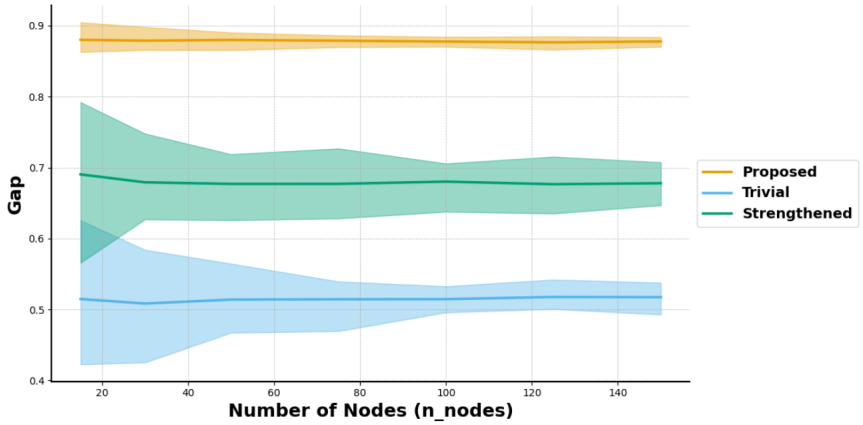

Figure 3: Median-, worst-, and best proportion of lower bound to optimal objective stratified after method

Having in mind that the objective has the addition of the fixed cost, whereas φ only approximates cost related to the flow, the proposal must yield a rather tight lower bound. In addition to the tighter bound, note that the method also gives a much lower difference between the worst- and best cases. This granularity, of course, comes with an increased computational cost. In the same experiment we illustrate the computation time of the methods
Though as we will see this computation time is on a much lower order of magnitude compared to the overall run-times and is only done once in the master problem. Thus the differences shown in figure 4 is negligible.

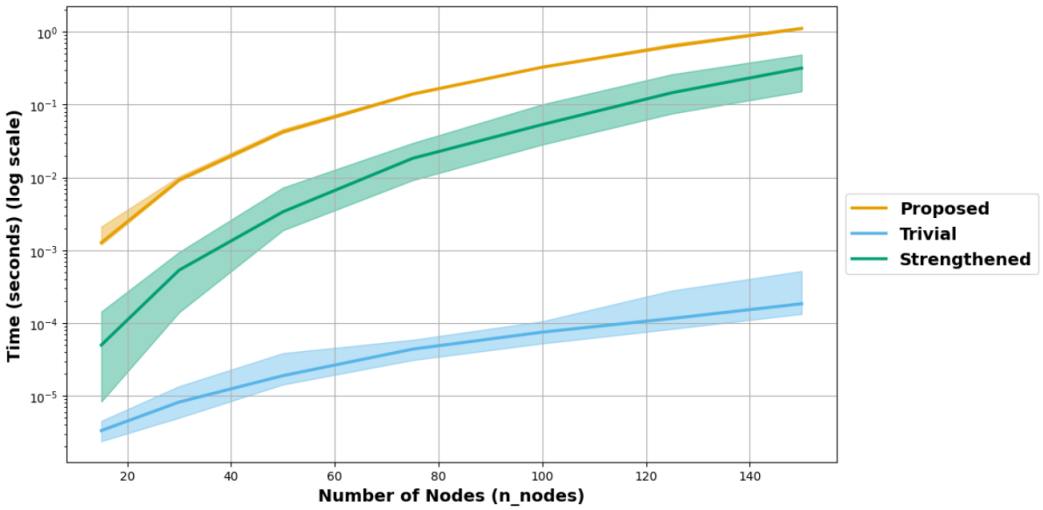

Figure 4: Median-, worst-, and best computation times stratified after method

Returning to the adding of constraints to the master problem, we add the a priori constraint based on the previous observation that a flow between the root and a directly connected node is always activated if the flow cost is less than 36. By construction, if for some nodes $i, j \in V$ both arcs $(i, j), (j, i) \in A$, then at optimality at most one will be active. Invoking similar reasoning and recalling that all nodes $v \in V \setminus \{1\}$ has strictly positive demand we have exactly one arc going into each node must be active for all v. Finally note, this also means that the total number of active arcs at optimality is exactly $n\_nodes - 1$.

Heuristic

As a final solution method we utilize heuristics. We name our heuristic: "First-best" and it is essentially just the solution corresponding to using the flows given by the proposed bound on φ in the previous section; we open the arcs along these paths and pay the fixed price regardsless of what it is. This of course gives an minimal objective, which is weakly larger than the true value. We make an experiment that creates and solves 20 random instances for each $n\_nodes \in \{10, 20, 30, 50, 75, 100, 125, 175, 225, 275, 300\}$ and computes the worst-, best- and median runtime and -ratio between optimal objective for both methods. The results are

As figure 5 is on a log-scale, we see that the ratio of the runtimes initially is quite large, decreases for medium sized problems and grows again for larger problems. Thus the heuristic becomes much more computationally sound for larger instances. However, as is clear in figure 6, the simplicity of the heuristic also results in larger errors on the objective.

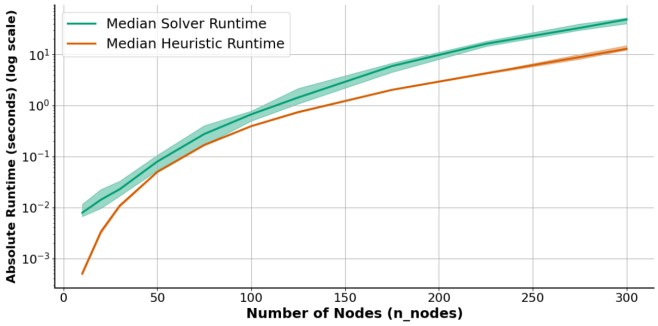

Figure 5: Run-times as a function of number of nodes stratified after method.

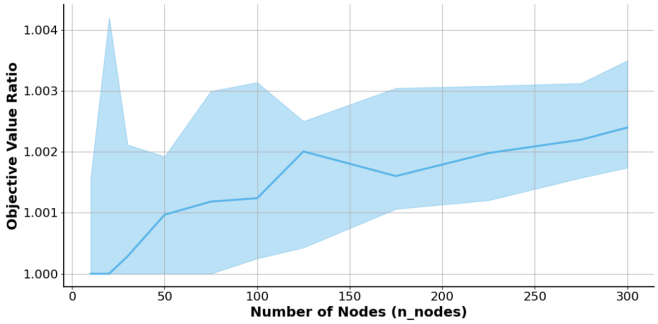

Figure 6: Ratio between objective value from heuristic and solver

Results

In this section we examine the results of various implementations. We highlight strengths and weaknesses of implementing each of the proposed methods of improvement into either formulation of the FCNFP. Throughout the section, the version of Benders' Decomposition associated with the formulation given by equations 1-4 will be referred to as the "original formulation" and the version of Benders' Decomposition associated with the formulation given by 5-10 will be referred to as the "split-variable formulation". When comparing objective values, it is noticeable that the objective values provided by running either implementation of Bender's Decomposition at times underestimates the objective value. This is more pronounced in the original formulation and happens increasingly as the number of nodes increases until plateauing at approximately 45 nodes with an objective-ratio of about .99. This is likely due to the weak bounds on x_{ij} in 3. When the number of nodes increases as does total demand implying that for sufficiently many nodes $x_{ij} << \frac{y_{ij}}{b_r}$ implying $x_{ij}q_{ij}$ likely underestimates q_{ij} [1]. The problem is less pronounced when switching to the split-variable formulation, however the number of variables increases exponentially with the number of nodes causing the program to run significantly slower than the original formulation. The original formulation is overall preferable with regards to performing a meaningful timed experiment as the split-variable formulation's runtime by far exceeds the point where it can be considered practically useful (and even feasible) when the number of nodes supersedes 20. Therefore we present a comparison of the Benders' Decomposition

applied to the original formulation with the gurobi solver.

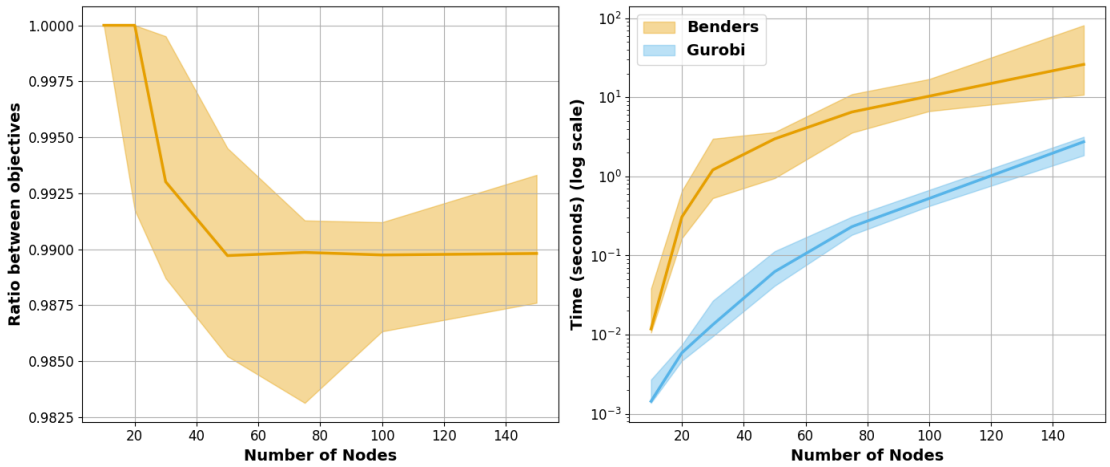

Figure 7: Comparison of runtime and ratio of objective value

It is evident from the computation times that a direct comparison between the two methods for a fixed time will not yield anything interesting. There is almost an order of magnitude between the computation times for each size of n_nodes. Therefore we finish with the following remarks: The heuristic method is definitely the fastest in comparison to the exact solver. Also, the error it makes is relatively small even for larger instances. Numerical issues related to the nature of the LP relaxation of the initial formulation, makes it difficult to properly compare the solver to the Benders decomposition in this formulation. On the other hand, as noted, the Benders Decomposition had running-times so slow in relation to the solver that a comparison is redundant.

References

[1] Lecture Notes: Writing Good Formulations.

[2] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023.

[3] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.