IE716: Integer Programming **Jan-April 2023**

End-term Project Report: Plotting Branch and bound tree using Gephi

Team Name: Team RDS *Team Member: 21i190003, 21i190006, 21i190010*

1 Introduction

Branch and bound is a technique used in optimization problems to find the optimal solution by systematically exploring the search space. It involves the following steps:

- Divide the problem into smaller subproblems (branches).
- Solve each subproblem individually, either by brute force or using some heuristics.
- Determine a lower bound for the optimal solution for each subproblem.
- If the lower bound for a subproblem is worse than the current best solution, prune that subproblem and move on to the next.
- If the lower bound for a subproblem is better than the current best solution, explore that subproblem further by repeating steps 1-4 recursively.

This process continues until all subproblems have been explored, and the best solution has been found. The key idea behind branch and bound is to eliminate subproblems that cannot contain the optimal solution, which reduces the overall search space and speeds up the search process.

The Gephi tool is a popular software tool for visualizing and analyzing complex networks, including branch and bound trees. The tool allows users to adjust the layout and styling of the tree to make it easier to understand and interpret.Here the branch and bound tree plotted using Gephi tool provides a powerful visual representation of the search process, helping users to better understand and optimize the algorithm's performance.

2 Methods and Approaches

For plotting the branch and bound tree, we have used the tree data generated by minotaur solver.

2.1 Minotaur: Toolkit

Minotaur is an open-source software for solving Mixed-Integer Nonlinear Optimization (MINLO) problems.It includes solvers for both convex and nonconvex problems.

It has following solvers in the toolkit:

- mbnb: nonlinear optimization based branch-and-bound for convex MINLPs

- mqg: LP/NLP based branch-and-cut for convex MINLP
- mqgpar: Shared-memory parallel version of LP/NLP based branch-and-cut for convex MINLP
- mglob: Global optimization solver for quadratically constrained problems.
- multistart: Multistart heuristic for nonconvex MINLP

We have used the solver 'mbnb' since we have used convex MINLP problems. The tree data file saved from the solver, was used to generate the data into the format that is suitable for gephi.

2.2 Gephi:

Gephi is a tool to explore and understand graphs. It has the flexibility to plot and visualize both static and dynamic graphs. It can visualize the networks up to 100,000 nodes and 1,000,000 edges. We can visualize how a network evolves over time by manipulating the embedded timeline which can be set during the visualization into the gephi itself. This enabling timeline is available only in the trees for which data contains timestamps for edges and nodes or the time intervals for the edges and nodes. It can read data from spreadsheets of .csv format with nodes and edges data.

3 Data Extraction Details

The data was stored into .txt file using the -vbcfile option from mbnb solver. The vbc file generated contains the following details of the tree:

- The sequential details of the happening in the branch and tree.
- The time stamp of each node's generation.
- The details of the children of each node.
- The status of nodes in the form of labels. which can be written as follows:
  - 4: The node is unsolved, open
  - 2: The node is incumbent (The current best)
  - 11: the node's problem is infeasible.
  - 9: The node has been solved.
  - 8: The node is currently being solved.
  - 13: The subtree is infeasible.
  - 6: Node is suboptimal.

We ignored the visualization of status of each node in gephi tree, and wrote a python code, which takes the solver generated .txt vbcfile as input and returns two csv files named node_data_InstanceName and edge_data_InstanceName. Edge_Data file contains the column named Source Node, Target Node and Type of edge (Directed or Undirected). And Node_Data contains 3 columns named Node_ID, Node_Label and the TimseStamps for which that node was available and must be visible in the tree. This timestamp was taken as natural numbers starting from one. For example The first node would be visible in the tree for all the time stamps corresponding to all nodes. This timestamp has nothing to do with the one given in the vbc file. And it has been taken in this way for the simplicity. Node_Data and Edge_Data files for an instance syn20 have been shown in the below figure-1 & 2.

Id	Label	Timestamp
0	v0	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
1	v1	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
2	v2	[2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
3	v3	[3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
4	v4	[4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
5	v5	[5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
6	v6	[6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
7	v7	[7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
8	v8	[8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
9	v9	[9, 10, 11, 12, 13, 14, 15, 16, 17, 18]
10	v10	[10, 11, 12, 13, 14, 15, 16, 17, 18]
11	v11	[11, 12, 13, 14, 15, 16, 17, 18]
12	v12	[12, 13, 14, 15, 16, 17, 18]
13	v13	[13, 14, 15, 16, 17, 18]
14	v14	[14, 15, 16, 17, 18]
15	v15	[15, 16, 17, 18]
16	v16	[16, 17, 18]
17	v17	[17, 18]

Figure 1: Node data table for syn20 instance

Source	Target	Type
0	1	directed
1	2	directed
1	3	directed
2	4	directed
2	5	directed
4	6	directed
4	7	directed
6	8	directed
6	9	directed
8	10	directed
8	11	directed
10	12	directed
10	13	directed
12	14	directed
12	15	directed
14	16	directed
14	17	directed

Figure 2: Edge data table for syn20 instance

4 Experiments & Results

We plotted graphs for syn30, syn40 instances from MINLIPLIB Library and also for a water network problem that is convex. The MINLPLIB Syn30m instances are a subset of the Syn30m problems that are formulated as mixed-integer nonlinear programs (MINLPs) with linear constraints. These instances have several properties that are given as:

- Nonlinearity: The objective function and some of the constraints are nonlinear, which makes them difficult to optimize using traditional optimization methods.
- Mixed-integer variables: The MINLPLIB Syn30m instances have a combination of continuous and integer variables, which makes them more complex to solve than problems with only continuous variables.
- Global optimization: The MINLPLIB Syn30m instances have multiple local optima, and finding the global optimum is challenging.
- Linear constraints: The constraints in the MINLPLIB Syn30m instances are linear, but they are still difficult to satisfy due to the complexity of the objective function and the mixed-integer variables.
- Dimensionality: Some of the MINLPLIB Syn30m instances have a high number of decision variables, which makes them more difficult to solve due to the curse of dimensionality.

Some of the properties of syn40m that are different from syn30m are given as:

- Global optimization: The Syn40m instances have multiple local optima, and finding the global optimum is challenging.
- High dimensionality: Some of the Syn40m instances have a high number of decision variables, which makes them more difficult to solve due to the curse of dimensionality.

Hence the MINLPLIB Syn30m and Syn40m instances are a useful benchmark for testing and comparing the performance of MINLP solvers on challenging optimization problems with mixed-integer variables, nonlinearity, and global optima.

- Plot for syn30 Instance:

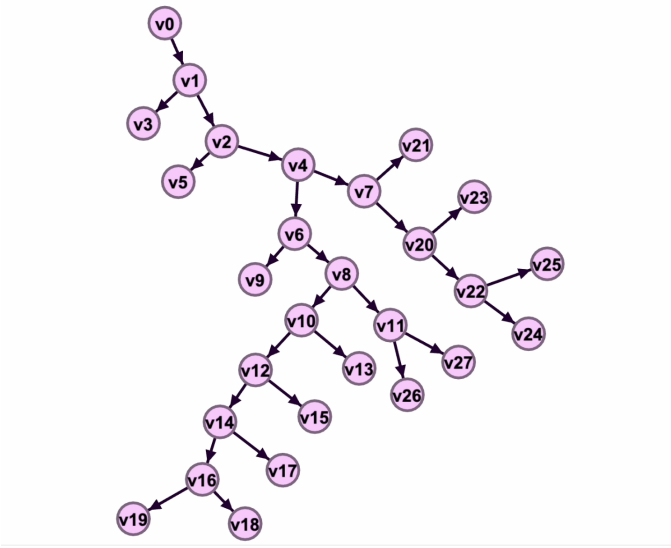

Figure 3: Branch and bound tree for Syn30m instance

- Plot for syn40m instance:

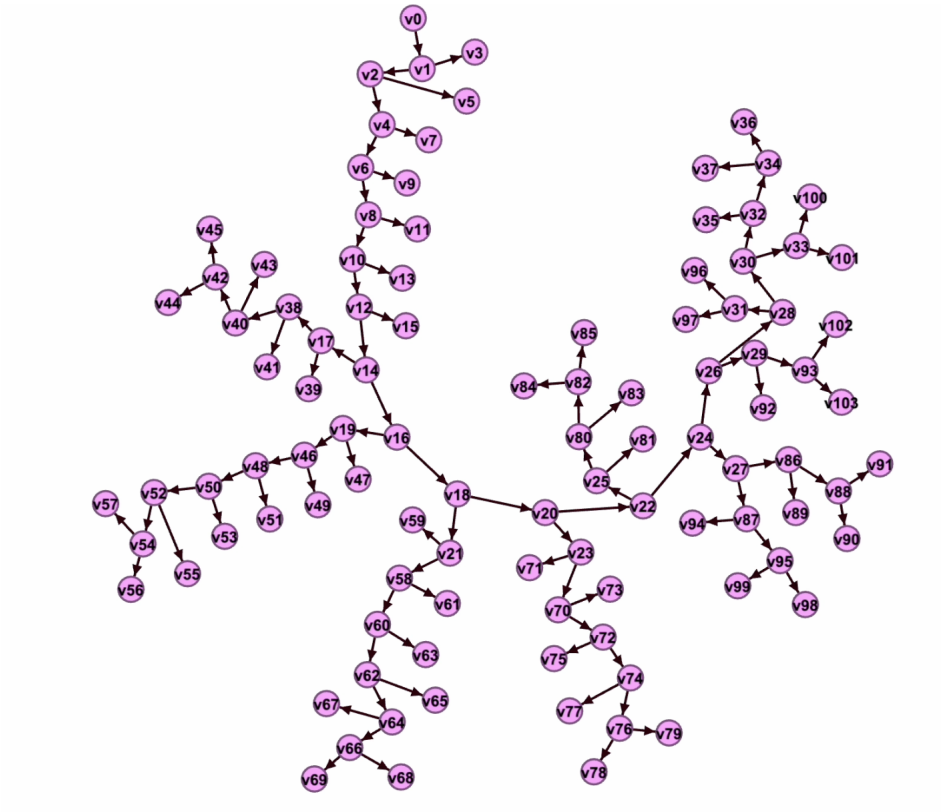

Figure 4: Branch and bound tree for Syn40m instance

- Plot for Water network instance:

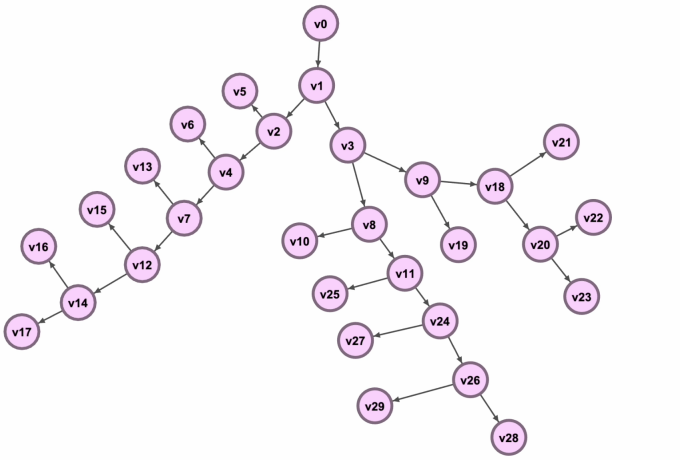

Figure 5: Branch and bound tree for Water network instance

5 Future Work

The further work that can be extended in the future:

- We have used a gap of 1 unit of time between the generation of consecutive nodes but further it can be replaced by time intervals.
- The real-time status of a node can be shown using color labeling.
- We can also explore the Gephi use in the concurrent appearance of the children nodes that can be enabled by making slight changes in timestamp data.
- The Gephi use can also be explored in assigning the weights to the nodes in the sense of their importance and hierarchy in a tree.

6 Attached Files:

- Each folder named syn30, syn40 and water consists of four files. The .txt file is the original vbc file generated from minotaur solver, two .csv files are the data tables generated after using python code on to the .txt file. And the other file is the clip for the visualization of tree.
- Python code file .ipynb notebook has been attached that was used for data extraction.
- The report has been submitted.

Bibliography

1. `https://github.com/gephi/gephi.git`
2. `https://docs.gephi.org/`
3. `https://stackoverflow.com/questions/tagged/gephi`
4. `/home/amahajan/instances/minlplibconvex/` Directory