UNIVERISTY OF MISSOURI, COLUMBIA

Computational Optimization Exam Answers

Nguyen Ph. Nguyen

December 1st, 2014

Abstract

This paper presents method to apply computer in Computational Optimization. The concept here is combination of functional software for sub processes in a whole workflow. The tasks done in sub processes are from preparing data, models and configuration to solving problem, and final evaluation. This can benefits research project in productivity and support experiment design.

# Introduction of research workflow in Computational Optimization

Computational Optimization applies computers to solve optimization problems. These problems can be supply chain optimization, transportation planning problem, or energy efficiency problem that are usually large and complicated. Expected results are not only solutions but also performance. Researchers need to formulate problems, collect/process data, build models, run and evaluate results. This work usually requires serial and/or parallel combination of many tasks in many software programs.

Each program has specific function (data processing data analyzing, solving …), is written in some language (Java, Python, C++ …), and run in some platform (Windows, Linux, OsX). One alternative for this is using API to directly connect between 2 programs (such as r2py …). The advantages of this method is faster speed. However, when the number of program increases, communication in this network become too complicated and heavy. We need a flexible solution that separates data part and program part in our job. Each program will work with database to exchange information with other programs via database. A workflow will control and monitor all these activities: generating, collecting, storing, analyzing and visualizing data and results. That’s the automating research workflow.

It also uses configuration (.ini) file to allow users to flexibly change working scenarios or configuration. With this kind of workflow, automation, handling large amount of data with high accuracy, and easy reproduction, flexibly changing scenarios are advantages to increase research productivity.

One important thing is automating workflow supports to do computational experiment in research process. Workflow helps to generate, collect, storing, analyze and visualize data and results. So now researchers can fully focus on their reasoning work, easily change, or combine or reproduce study scenarios, or simulate to find the best solution.

# Typical research workflow in Computational Optimization

So the tasks are ordered in a workflow to be automatically implemented by an orchestration. There are 2 types of task in workflow: independent task and dependant task. Independent tasks can run simultaneously, while dependant tasks must run in order. A typical combination of workflow is diamond workflow, shown in Figure 1. Task 2 and Task 3 are independent, and both of them depend on Task 1.



Figure 1: Diamond workflow

A specialized case of diamond workflow in Computational Optimization is shown in Figure 2. In this workflow, Task 1, 2, and 3 are for preparation. Data are collected, cleaned, and stored in csv file for database (SQLite) and passed to workflow.



Figure 2: Typical workflow in Computational Optimization

Real automating workflow begins from Task A, reads configuration file (.ini) to initialize path (data, program), environment variable, and reads data. After orchestration is ready, Task A will pass data and models to solvers. The solvers here can be Gurobi, Cplex, Microsoft Foundation Solver, OSi,. Then, Task B1, B2, and B3 are running solver with different method for problem 1. They are independent each other and can run in parallel. But they depends on task A, and task D depend on them. Solutions and logging information (running time) are recoded to database. After getting solution and running time from 3 methods, workflow will activate task D, call R to analyse data. R script file run analysing scenario to do experiment, find significant results as well as visualize results.

This workflow is implemented in Python, SQLite, R as details. ([see Appendix](#_Appendix))

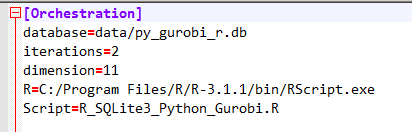
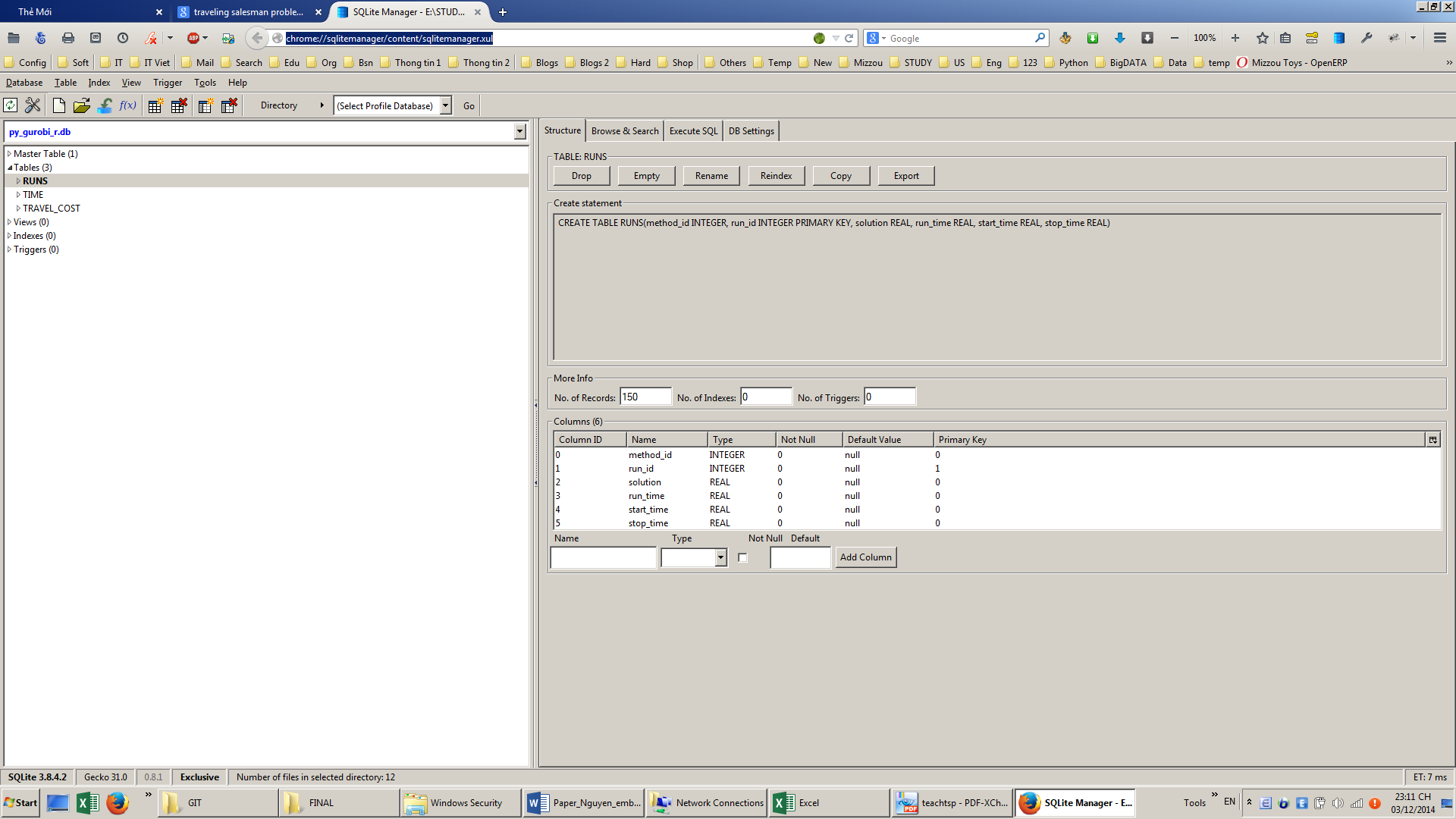


Figure 3: Configuration file

# Data Collection

Input data for optimization problem in this project are collected from Python random generator. Number of cities that travelling sales man will reach are set in configuration (dimension parameter). Value ranges are [1, 10] for time(cost), early time is 0, and late time from [30,100]. Input data are stored in sqllite database. Result data also record in this database for analysing. Workflow will run 50 iterations (set in configuration file), and refresh dataset after each iteration.



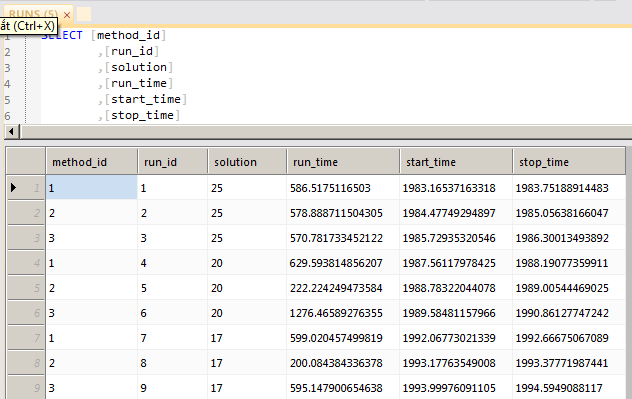


Figure 4: SQLite database

# Traveling Salesman Problem with Time Windows

The travelling salesman problem (TSP) try to find the shortest possible path among a group of cities that salesman visits each city exactly once and returns to the origin city.

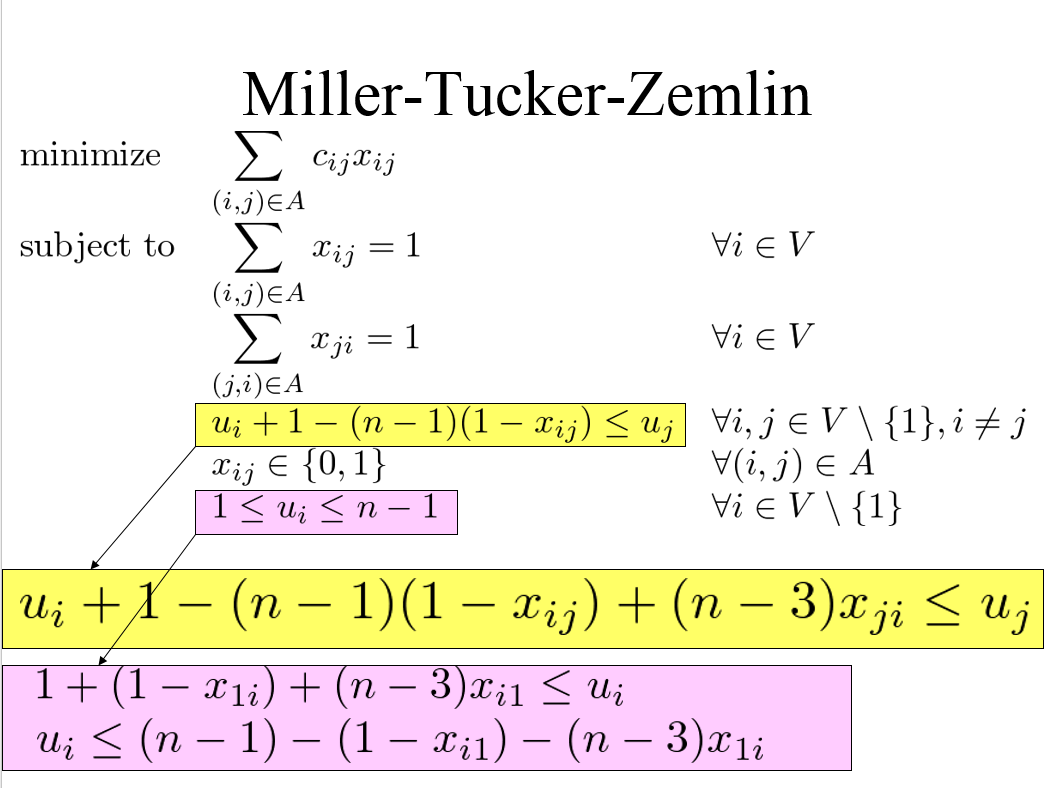


For i = 1, ..., n, let

ui be an artificial variable (sequence number in which city i visited)

cij is the time (distance) from city i to city j

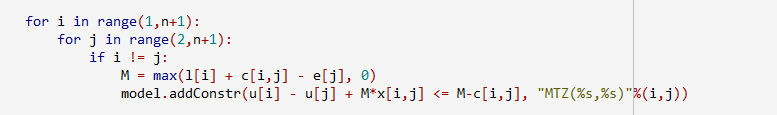
The model is formulated as following [1]:



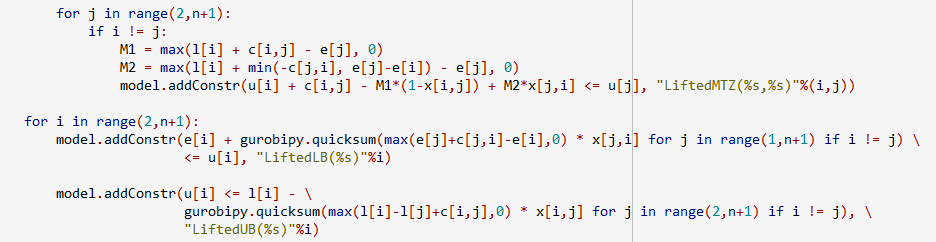
2 last contraints exclude subtours, they force uj ≥ ui + 1,when xij = 1

Time Windows constraints:

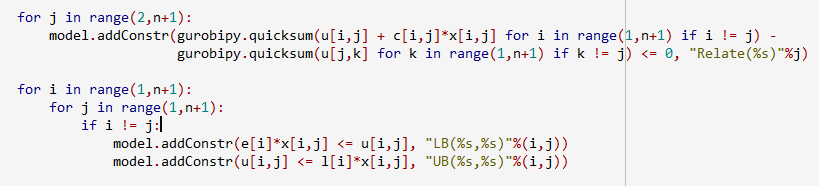
Model 1:



Model 2:



Model 3:



# Result and Evaluation

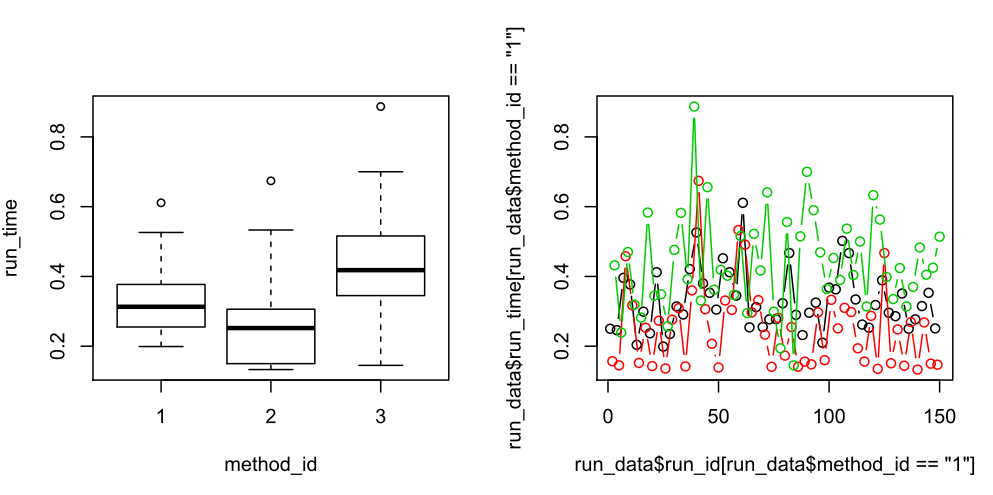


Figure 5a: Analysing result

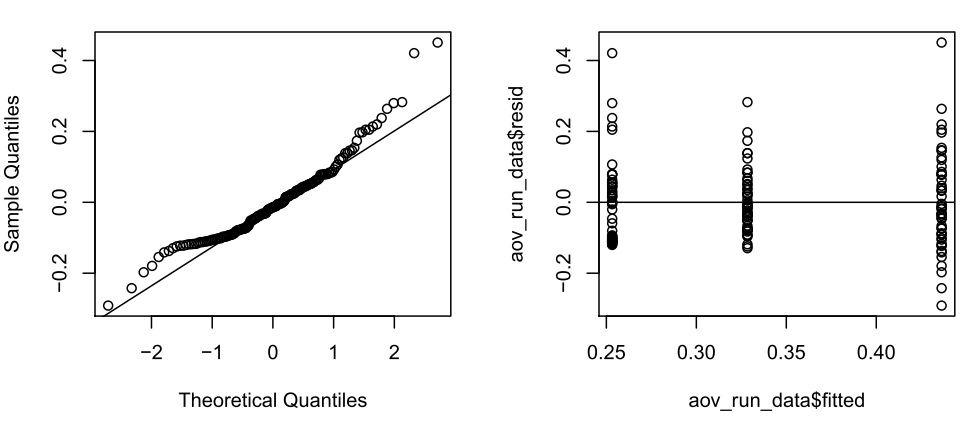
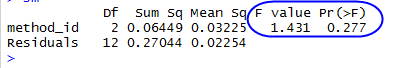


Figure 5b: QQ norm plot



From series chart, it looks like method 2 gives smaller solving time. However ANOVA result tells us that there is no significant difference between solving time of 3 methods. Also, for small problem, solving time is below 1s. This can be affect with large proportion of noise from computer operation. So the result is not reliable.

# Conclusion

.

# References

|  |  |
| --- | --- |
| [1] | Mikio Kubo, Joao Pedro Pedroso, Masakazu Muramatsu, Abdur Rais, *Mathematical: Solving problems using Gurobi and Python,* 2012 |
| [2] | Kate Anderson, *Managing & Sharing Your Research Data*, MU Libraries, 2014. |
| [3] | University of Florida Data Lifecycle Management: http://ufdc.ufl.edu/IR00000801/00001 |

# Appendix

File structure of workflow

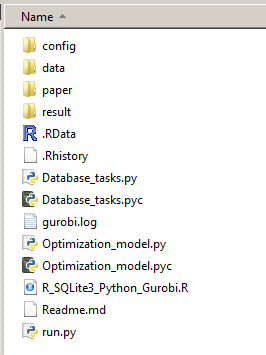


Figure 6: Project file structure

At first, we need to import following packages to Python running environment.

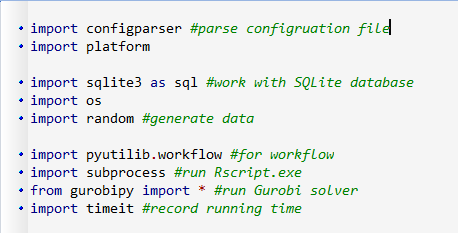


Figure 7: Imported packages

And define workflow.

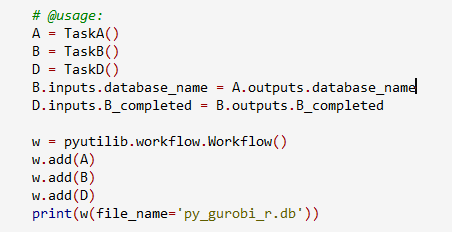


Figure 8: Workflow

Task A will open database, and configuration file.

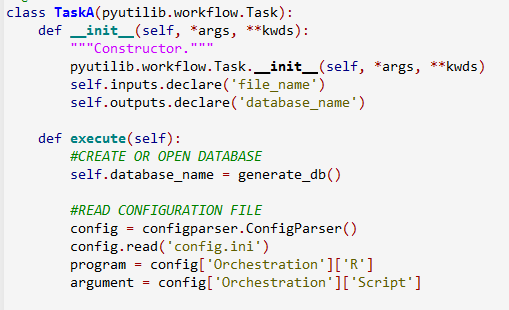


Figure 9: Task A

Generate database and insert data.

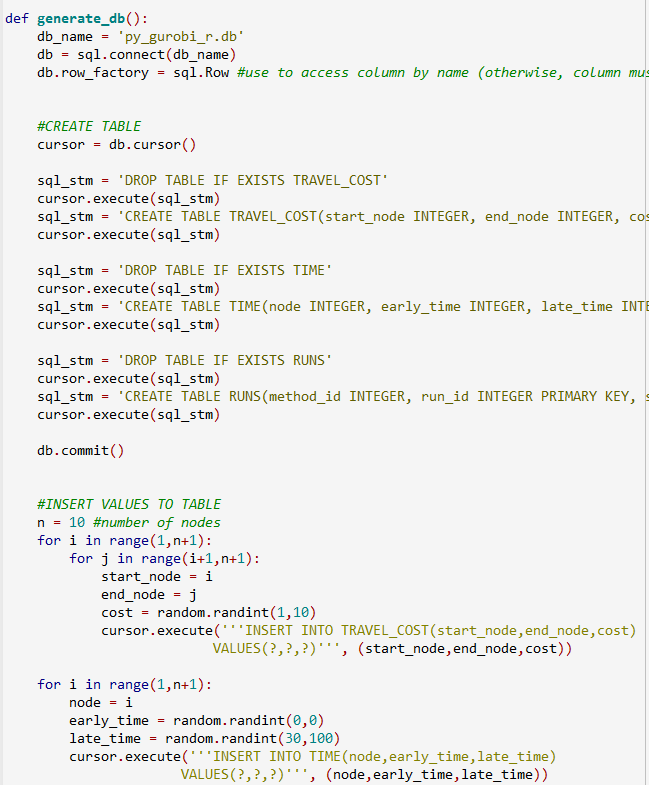


Figure 10: Open SQLite database and generate data

Task B will solve problem 1

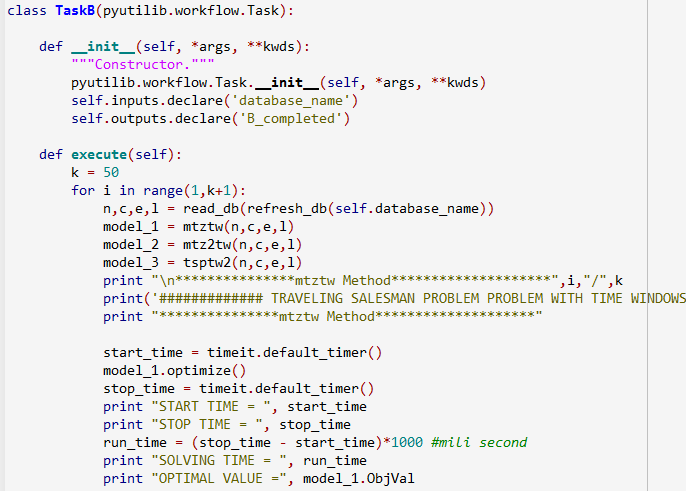


Figure 11: Task B for problem 1

Model for task B

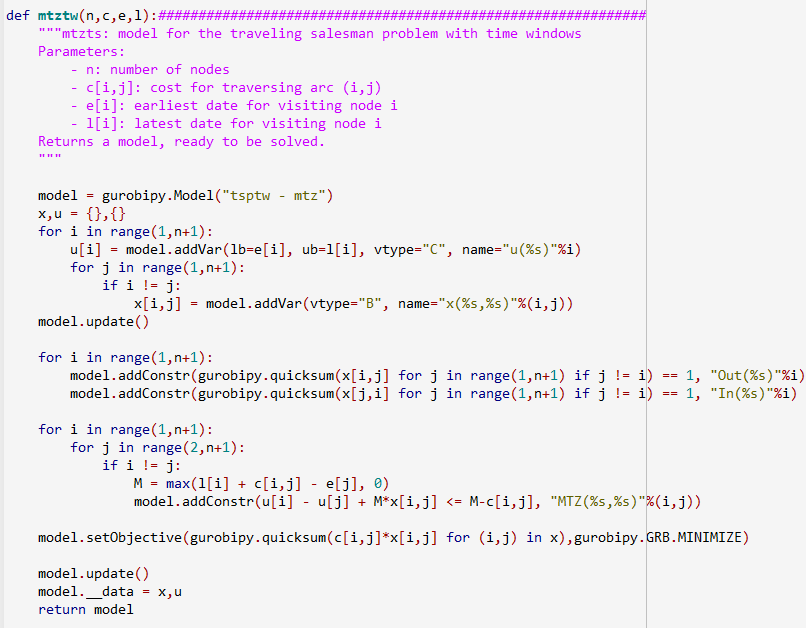


Figure 12: Model of task B

Task D will call R to analyse and visualize results.

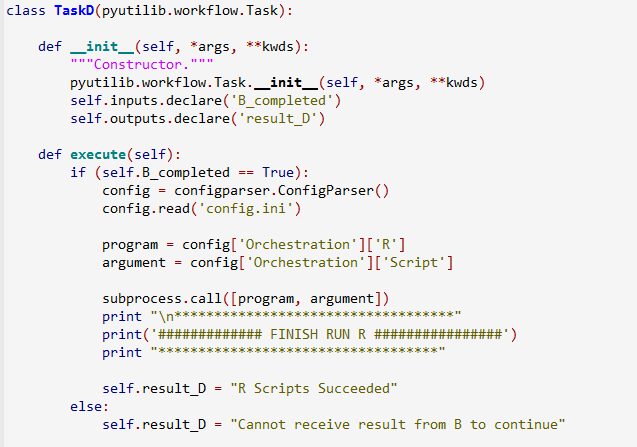


Figure 13: Task D will run R script

R Script

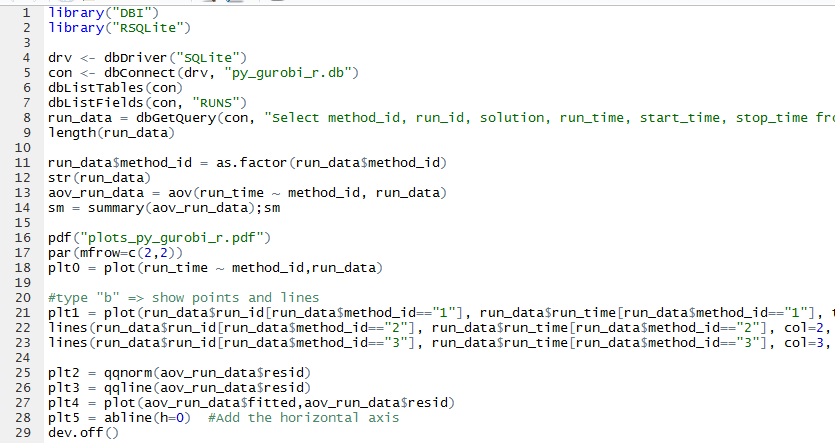


Figure 14: R script for analysing data