Final Project: File server allocation

Erik Saule

Submission deadline: Reading day (December 4th)

1 Preliminaries

This project is an individual project. Each student is expected to complete it on his/her own without relying on external resources. External resources include friends, students from other years or universities and the web. In case of doubt about what is and is not acceptable, consult the code of student academic integrity at `http://legal.uncc.edu/policies/up-407` or me.

The expected time to complete this assignment in tens of hours. I estimate that it will take me about a day to complete this assignment. Implementing these algorithms can be tricky. Finding the correct algorithms can be difficult. To be safe, I suggest you plan at least 3 days of work. But probably you should take a first crack at it as soon as possible so that you get a clear picture of what needs to be done.

2 Introduction

This problem is linked to the allocation of files on content distribution networks (CDN). In this problem we assume that each file can only be placed on a single server in the CDN. Each file will cause a bandwidth requirement on the server it is allocated to. The purpose of the problem is to find an allocation of the files on the server so as to minimize the bandwidth required by the server that requires the most bandwidth.

Formally, the File Server Allocation (FSA) problem is defined by n files and m servers. Each file i has a bandwidth requirement $b_i > 0$. The purpose of the problem is to choose for each file i a server $S(i) \in \{1, \ldots, m\}$ on which file i will be allocated. The total bandwidth requirement for server j is $B(j) = \sum_{i|S(i)=j} b_i$. The problem is to minimize $\max_j B(j)$.

The purpose of the project is to provide both algorithmic solutions (algorithms, proofs, complexity, properties) AND implementation (code and evaluation) to obtain a higher understanding of the problem.

3 Brute Force

Design a brute force algorithm for the FSA problem. That is to say an algorithm that tries all the possible solutions for the problems and pick the best one.

4 Dynamic Programming

Design a dynamic programming algorithm for the problem. The key to designing the dynamic programming algorithm is to find the correct suboptimality property (optimal structure) in the problem. It is non trivial to exploit an optimal substructure in this problem.

A way to think about the problem is to transform the optimization problem into a decision problem. That is to say, instead of trying to find the minimal $\max_j B(j)$, to try to decide whether there is a solution such that all machines have a $B(j) \leq B$, where B is a parameter of the algorithm.

A suggested way to design the algorithm follows (note that this is not the only way to design it):

- Show how to solve the optimization problem assuming a blackbox algorithm for solving the decision problem.
- Find a sub-optimality property on the decision problem (i.e., if this solution is feasible with this property, then that (smaller) solution is feasible with that property). Hint: encode the problem using m values that gives a budget in bandwidth for each server and one value that gives the number of files.
- Deduce the dynamic programming formula.
- Write the algorithm (either top-down or bottom-up).
- Reconstruct the solution.

5 Heuristic

Design a heuristic algorithm for this problem. That is to say, design an algorithm that is not proven to be optimal but that should run faster than the previous two algorithms.

6 Expected work

Two things are expected as an output of the programming assignment. Runnable code that can solve the problem AND a detailed report.

The grades will be split essentially 33% for each of the following categories: algorithm design, code and evaluation.

6.1 Algorithm design

This goes into the report.

Provide the design for the 3 algorithms. That is to say, the report should include at least:

- The algorithm itself (not code). To be clear on what level of details is expected: This should be clear enough so that a second year undergraduate student can simply read the explanation and be able to write the code that outputs the solution. In other word, give complete pseudo code.
- An analysis of the time complexity of the algorithm.
- An analysis of the memory complexity of the algorithm.
- For the dynamic programming:
  - What problem is actually addressed by the dynamic programming (you might be using the decision version). And how does that relate to the FSA problem.
  - What is the sub-optimal structure leveraged by the dynamic programming, and the proof that this structure is correct.
  - The dynamic programming formula.
  - The complexity of the algorithm.

6.2 Code

The code should be written in Java, C or C++. It should be easy to read. And it should compile and run according to common practice.

For easy testing, make one program per algorithm amd write the codes so that they take three parameters from the command line: the path to a file containing the bandwidth requirements b_i, the number of tasks n, and the number of servers m. The file should contain the bandwidth requirement one per line. The file might be much longer than that n. Only use the first n lines of the file.

Some instances will be provided. But feel free to create your own.

The codes should output the solution to the problem. The time it took to compute the solution and the value of the objective ($\max_j B(j)$) for this solution.

6.3 Evaluation

This also goes in the report.

The idea is to evaluate the quality of the different algorithms and in which conditions they perform well or not. To do that you should run the codes varying the number of files, the number of servers and the type of instances that you. Two types of analysis are expected.

First, you should verify the quality of the solution. The brute force and dynamic programming should return solutions with the same objective value $\max_j B(j)$. But the heuristic might return something different. How high is the price of running the heuristic?

Second, you should investigate the time that it takes to run the instances. Don't run any code for more than a time budget (two hours per run seems reasonable). Some questions you should answer:

- Which algorithm is the fastest?
- Which optimal algorithm is the fastest? in which conditions?
- Does the runtimes match your expectation?
- What is the largest problems that can be run within a time budget? (as a function of n, m, the type of instances and the algorithm used.) Note that some algorithm might reach OOM.

7 Extra Credit

- Design and test BOTH a top-down dynamic programming algorithm and a bottom-up one.
- Design an optimal algorithm inspired by the dynamic programming formulation with much higher efficiency.
- Prove a meaningful guarantee for the quality of the solution generated by the heuristic.
- Additional insight about the problem.