Optimal P2P network

Nikita Popov, Danil Kharasov, Yaroslav Kishchenko

December 22, 2015

Problem description

Suppose we have a network of N computers, and one of them (or several) has a file that we want to distribute to every other peer. The sooner all the computers will have the file the better, and this is the optimization task we want to solve.

Our base model for this task will be continuous, and later we will discretize the problem. Connections between peers can be modelled via graph: vertices of the graph will represent the computers in our network, and weights of the edges is the bandwidth between the pair of the computers. Let the adjacency matrix of this graph be B. Besides the bandwidth of pairs we also have connection speed of every computer to the network. This values will be assigned to vertices. Let u_i be maximum upload speed for the computer i, and d_i be the maximum download speed for the computer i.

The last part of our model (and the beginning of discretization) is the file description. All we need to now is file's length, we will name it L. We will split the file in K parts and will deliver the parts along the net. Discretizing time and recomputing the bandwidth and all upload and download restrictions in terms of amount of parts (which will be integers),

$$\hat{b}_{ij} = b_{ij}\frac{dt}{dL} \tag{1}$$

where $dL = \frac{L}{K}, dt$ – time step, we now ready to formulate the problem.

Let x_{ikt} be a variable that is equals to 1 when computer i has a part k at the moment of time t and to 0 otherwise, and y_{ijkt} be a variable that is equals to 1 when part k was transferred from computer i to computer j at moment of time t. In terms of x and y we want to minimize the following thing:

$$\max_i \min\{t : \sum_k x_{ikt} = K\} \to \min_{x,y}. \tag{2}$$

In other words, we want to minimize the time when all computers in network will have all parts of the file. Now let's formulate the restrictions on the variables x, y:

$$\sum_k y_{ijkt} \leqslant b_{ij} \tag{3}$$

$$\sum_{ik} y_{ijkt} \leqslant d_j \tag{4}$$

$$\sum_{jk} y_{ijkt} \leqslant u_i \tag{5}$$

$$\forall t' \geqslant t \; x_{ikt'} \geqslant x_{ikt} \tag{6}$$

$$y_{ijkt} \leqslant x_{ikt} \tag{7}$$

$$y_{ijkt} \leqslant x_{jk,t+1} \tag{8}$$

$$\sum_{i,t} y_{ijkt} = 1 \tag{9}$$

$$\forall i \in S_0 \; x_{ik0} = 1, \forall i \notin S_0 \; x_{ik0} = 0 \tag{10}$$

$$x_{ikt}, y_{ijkt} \in \{0, 1\}. \tag{11}$$

The description of every inequality is the following:

- Equation (3) is the bound on the amount of parts that can be transferred within connection between i and j.
- Equations (4) and (5) restrict the uploads and downloads for computer i.
- Equations (6) forbid the disappearance of the part k from computer i.

- Equations (7) state that part k can be downloaded from computer i if and only if this part has been downloaded to the computer i before.
- Equation (8) make the piece k available on computer j in next time step.
- Equations (9) forces the file piece to be downloaded from only one source in only one moment of time.
- Equations (10) imply that computers S_0 have the whole file at the beginning.
- Equations (11) requires the x and y to be binary.

Every restriction is linear in terms of x and y. To finalize this problem as a linear programming problem, we need to reduce the objective to a linear form. We use the following expression:

$$\max_i \min\{t : \sum_k x_{ikt} = K\} \to \min_{x,y} \Rightarrow \max_i \sum_t \gamma_t \Big(K - \sum_k x_{ikt}\Big) \to \min_{x,y}, \tag{12}$$

where γ_t are just increasing weights to force optimizer switch x's to ones as soon as possible.

First of all, it is not the same objective, but it's close: when computer i downloaded file, the expression $K - \sum_k x_{ikt}$ tends to zero, so the sooner the the computer download the file the better, but not in the same way better. Still, it's close. And the part $\sum_t \gamma_t K$ is constant, so we will minimize $\max_i \sum_{kt} x_{ikt}$.

This is the final problem:

$$\min_{x,y,\xi} \xi$$

$$s.t. \quad \begin{aligned} \sum_k y_{ijkt} &\leqslant b_{ij} \\ \sum_{ik} y_{ijkt} &\leqslant d_j \\ \sum_{jk} y_{ijkt} &\leqslant u_i \\ x_{ikt'} &\geqslant x_{ikt} \quad \forall t' \geqslant t \\ y_{ijkt} &\leqslant x_{ikt} \\ y_{ijkt} &\leqslant x_{jk,t+1} \\ \sum_{i,t} y_{ijkt} &= 1 \\ \forall i \in S_0 \;\; x_{ik0} &= 1 \\ \forall i \notin S_0 \;\; x_{ik0} &= 0 \\ x_{ikt}, y_{ijkt} &\in \{0,1\} \\ -\sum_{kt} x_{ikt} &\leqslant \xi \quad \forall i \end{aligned}$$

Variable ξ imitates the maximum.

Data generation

We generate data as following: first, we randomly generate positions on a plane for each computer and it's connection type (ADSL, Ethernet or Dial-Up). Then, we create a bandwidth matrix as following:

$$b_{ij} = \frac{C}{\|r_i - r_j\|} \cdot (0.5 + R[0; 0.5]), \tag{13}$$

where C is normalization constant, $R[a,b]$ - uniform distribution on $[a,b]$. Uniform distribution simulates connection jams between computers.

Applied methods

We tried Simplex method on relaxed problem with depth-first search of integer solution, convex optimization methods via CVXPY and greedy algorithm.

Greedy algorithm

The first approach was to implement any kind of greedy algorithm. In our case we start from the fastest connections and distribute them first, and later start distributing in descending by speed connection order.

CVXPY

Unfortunately, we haven't managed to make it work - optimizer failed every our attempt stating that our problem is infeasible. But code is still in the attachments.

Simplex method

The most promising method turns out to be Simplex method, but not without limitations - matrix of restrictions is huge and doesn't fit into memory for large parameters. But still, we achieved some results.

Time comparison

3,5,5	16s	163ms
4, 5, 5	32s	310ms
5, 5, 5	-	252ms
10, 10, 10	-	310ms
100, 100, 100	-	

Table 1: Time comparison of Simplex algorithm and greedy algorithm

Results visualisation

Down below reader can find visualisation for implemented algorithms. Plot means the following: axis x is for file pieces, axis y - computers. If the square is red, then this piece is downloaded on chosen computer. There are several time stamps for different situations.

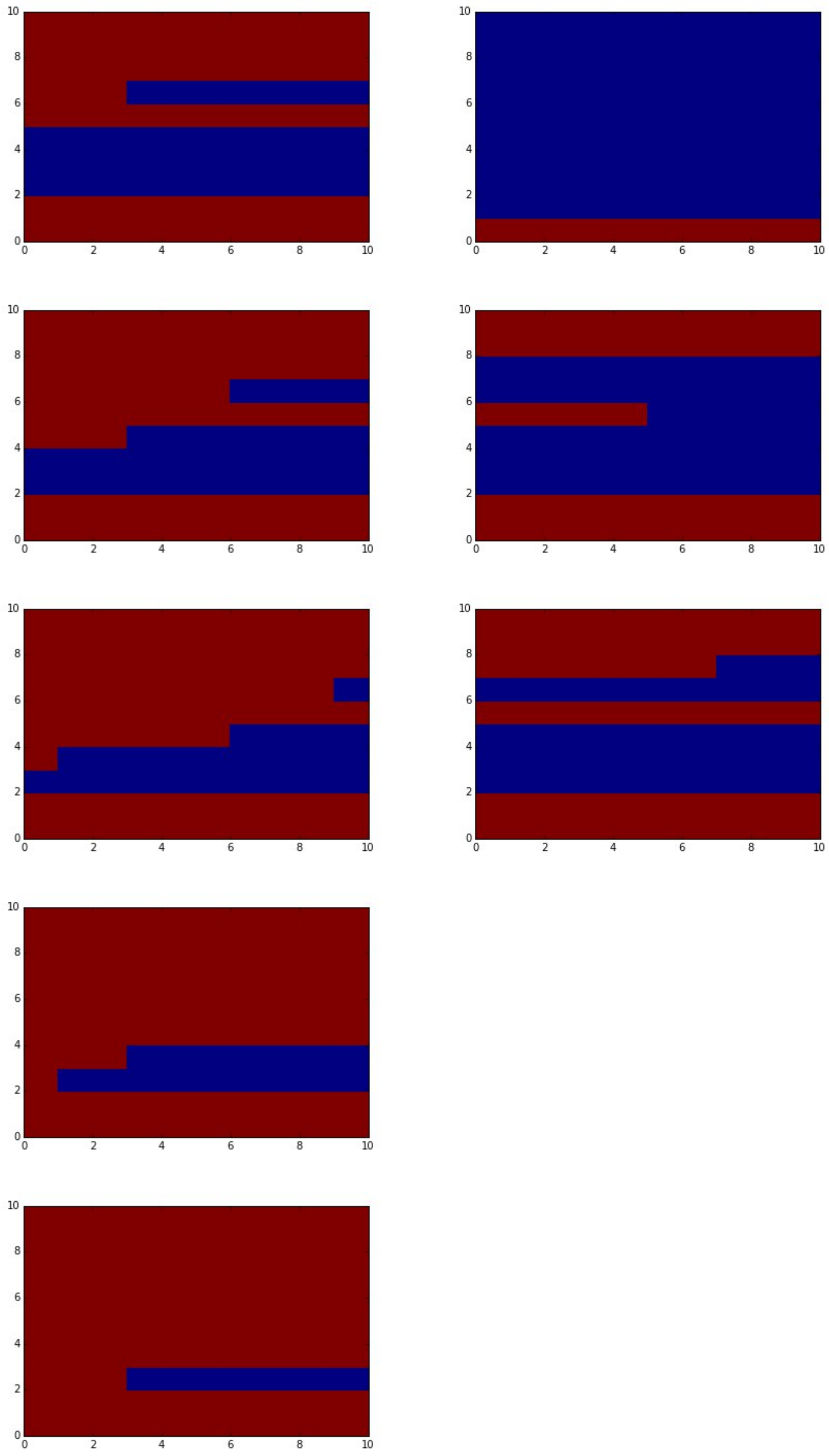

Figure 1: Simulation of greedy algorithm with $N = 10$ computers, 10 file pieces for two situations

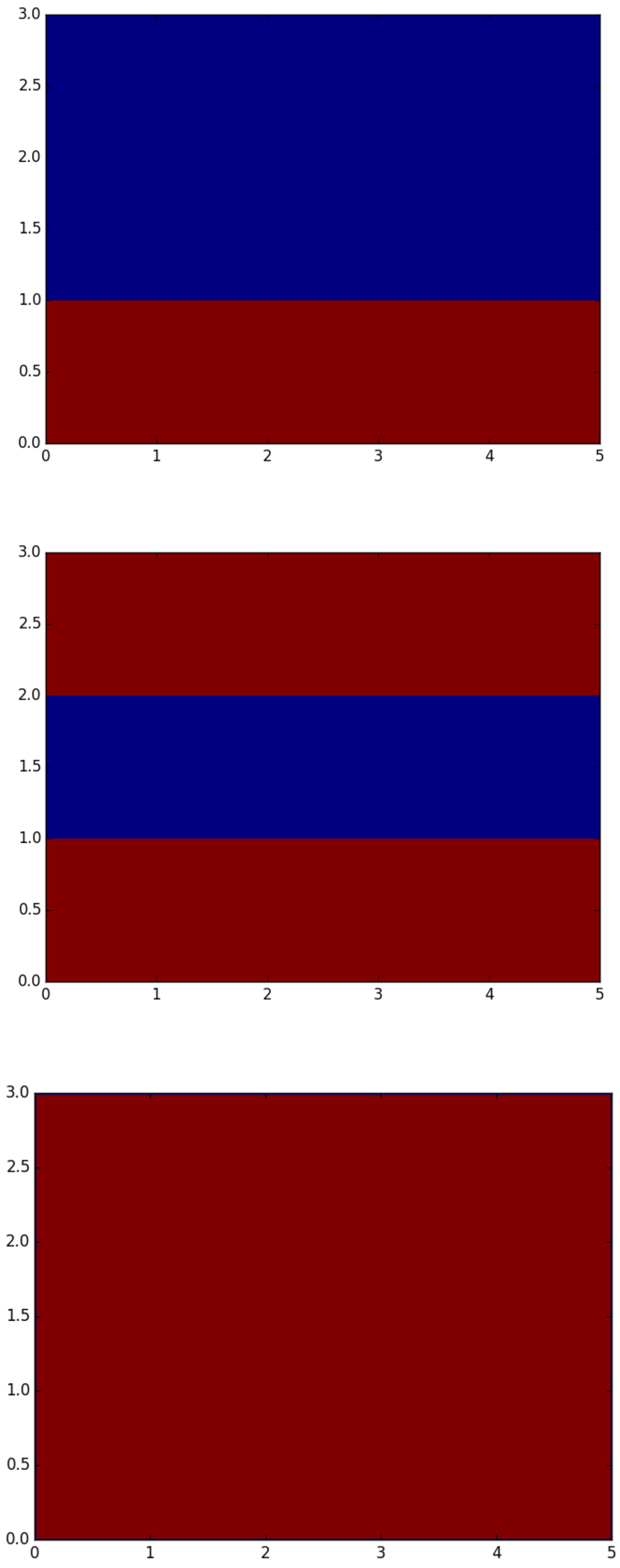

Figure 2: Simulation of Simplex method with $N = 3$ computers, 5 file pieces