Midterm Report: Solving Network Resource Allocation Using Linear Programming

Yongli Peng
1901210088@pku.edu.cn

May 12, 2020

1 Background

The network resource allocation problem can be formulated mathematically as an nonlinear, min-max optimization problem:

$$\begin{aligned} \min_{\mathbf{x}} \max_{l} & \frac{\mathbf{R}[l]\mathbf{x}}{c_l} \\ \text{s.t.} \quad & \mathbf{R}\mathbf{x} \le \mathbf{c} \\ & \|\mathbf{x}_k\|_1 = d_k \\ & \mathbf{x} \ge \mathbf{0}, \end{aligned}$$

where $\mathbf{R} = (\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_K)$ is the routing matrix denoting whether certain path transverses certain link, each submatrix $\mathbf{R}_k$ contains all the P_k paths for the flow k, $\mathbf{R}[l]$ exhibits all the paths transversing the link l, $\mathbf{c} = (c_1, c_2, \ldots, c_L)$ is the link capacity vector encoding the capacity for each link l, d_k is the bandwidth demand for flow k and $\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K)$ is the rate allocation and path selection vector we desire to obatin.

This problem can be transformed into a linear programming (LP) problem using an auxiliary variable:

$$\begin{aligned} \min_{\mathbf{x},t} \quad & t \\ \text{s.t.} \quad & \mathbf{R}\mathbf{x} \le \mathbf{c} \\ & tc_l - \mathbf{R}[l]\mathbf{x} \ge 0 \\ & 1^\top \mathbf{x}_k = d_k \\ & \mathbf{x} \ge \mathbf{0}, t \ge 0. \end{aligned} \tag{1}$$

Here we choose the 2nd task: Learn to Pivot. We want to use simplex method to solve the above LP, where a well-designed pivot rule is essential. To be precise, we need an appropriate strategy to choose the next vertex in each iteration of the simplex method. In practical the Dantzig rule and the steepest edge rule are used most frequently. Other rules, like Bland's rule and greatest improvement rule, have also been proposed. But there's no a canonical principle when coping with real problems. Recently some authors Varun, Onur, Ankit, and Andrew (2020) use deep value-based reinforcement learning to learn a strategy from data on choosing between Dantzig and steepest edge at each iteration. But the simulation sample size is too small to convince people using it in practice. We here want to apply this method in our LP (1) and compare its performance with methods using purly Dantzig or steepest edge. If possible, we can also try to generalize this method and try to learn a totally new pivot strategy. A naive way is to add other pivot rule to the action space. But I think maybe there's a more innovative way to merge reinforce learning with the pivot learning, which can give us a data-driven and problem specific pivot rule less dependent on our knowledge of simplex method.

2 Work we have done

We first need to determine which dataset we might use.
We might choose at least two groups of instances in the JLF problem as our test data. A small random generated data might also be used (where we just generate all the necessary parameters at random). But the instances in the JLF problem are for the multicommodity flow (MCF) problem and they are in the node-arc formulation, so we need to convert them into arc-path formulation to fit into our problems. Refer to James (2010), we have learned how to implement such a transformation and downloaded the dataset. We also learned how the parameters in (1) corresponds parameters in MCF problems. The transformation is:

$$
\begin{array}{l}
\text{Min } \sum_{(i,j)\in A} \sum_k c_{ij}^k x_{ij}^k \\
\sum_j x_{ij}^k - \sum_j x_{ji}^k = \begin{cases} d_k & \text{if } i = s_k \\ -d_k & \text{if } i \in t_k \\ 0 & \text{otherwise} \end{cases} \\
\sum_k x_{ij}^k \le u_{ij} \quad \text{for all } (i,j) \in A \\
x_{ij}^k \ge 0 \quad \forall (i,j) \in A, k \in K
\end{array}
\quad \Rightarrow \quad
\begin{array}{ll}
\text{Min} & \sum_k \sum_{P \in P^k} c^k(P) f(P) \\
& \sum_k \sum_{P \in P^k} \delta_{ij}(P) f(P) \le u_{ij} \quad \text{for all } (i,j) \in A \\
& \sum_{P \in P^k} f(P) = d^k \quad \text{for } k = 1 \text{ to } K \\
& f(P) \ge 0 \quad \text{for } P \in \bigcup_{k=1}^K P^k,
\end{array}
$$

where $c^k(P)/f(P)$ is the cost/rate allocation on the path P. The dataset of the latter problem offer all the parameters we need and we just need to replace the cost objective function with our maximal objective function. In this way we can get test data from MCF dataset.

As for the methodology, the concrete experiment has not been taken and we are still reading the paper. But we can introduce the basic idea on how to apply RL into the simplex method:

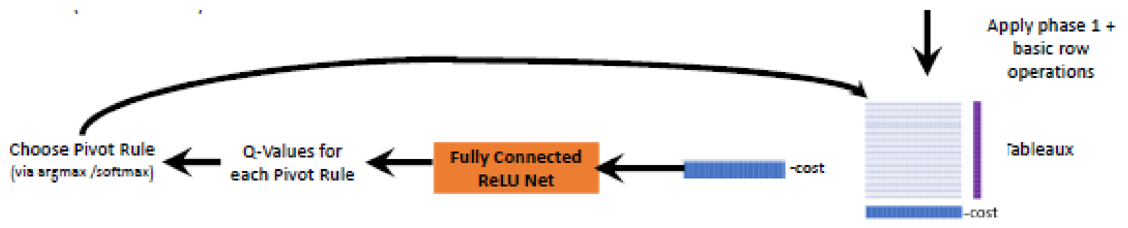
Apply phase 1 + basic row operations
Tableaux
-cost
-cost
Fully Connected ReLU Net
Q-Values for each Pivot Rule
Choose Pivot Rule (via argmax /softmax)

Figure 1: Basic workflow of the DEEPSIMPLEX method

After we find an initial extreme point to begin with, we use RL to find an appropriate strategy between Dantzig and steepest edge. Dantzig will compute the reduced costs for all combinations and choose the best one whereas the steepest edge will normalize the reduced cost with the norm of their corresponding updated constraint matrix columns and chooses the best one. The action space is $\{0, 1\}$, $a_t = 0$ if the Dantzig is selected and $a_t = 1$ if the steepest edge is chosen. The state space is the set of possible simplex tableaux, containing all the information we need to select the next point. Since the steepest edge is computationally more expensive, we will penalize more for it in the reward than the Dantzig. The relative weight w in the reward, according to the article, can be chosen using the ratio of the cpu time of the simplex using purely the Dantzig or the steepest edge. More formally, the reward is:

$$R(s_t, a_t) = \begin{cases} 0 & t > T \text{ or } \ell' = \ell^* \\ 1 - \frac{1}{T} & t \leq T, a_t = 0, \text{ and } \ell'(s_t) > \ell(s_t, a_t) = \ell^* \\ -\frac{1}{T} & t \leq T, a_t = 0, \text{ and } \ell(s_t, a_t) > \ell^* \\ 1 - \frac{1+w}{T} & t \leq T, a_t = 1, \text{ and } \ell'(s_t) > \ell(s_t, a_t) = \ell^* \\ -\frac{1+w}{T} & t \leq T, a_t = 1, \text{ and } \ell(s_t, a_t) > \ell^*, \end{cases}$$

where T is the maximum number of unweighted iterations, $\ell'(s_t), \ell(s_t, a_t), \ell^*$ are the objective values before the action, after the action and the optimal value.

And we may also exploit the architecture in the original paper: 8 fully connected layers, where each layer has a width of 128 with ReLU activation. The tanh activation is applied at the output.

3 Future work

In the following days, we will first implement the simplex method solving this LP (1) using purely the Dantzig rule or the steepest edge rule. Then we can determine the relative weight in the reward function and implement DEEPSIMPLEX. The accuracy as well as the cpu time will be compared among these methods.

Moreover we might try the following idea using RL to learn the pivot: let the action space be all the adjacent points (all the adjacent points with reduced costs $\bar{c}_j < 0$) and we want to learn a transition probability from the present point based on the present state. We expect this probability will only concern the reduced costs before and after the action. This might provide a data-driven and problem specific pivot rule, but how to design the reward function is important and I have no idea yet.

References

James, O. (2010). *Lecture notes: Network optimization, multicommodity flows 2.* Retrieved from `http://engineering.purdue.edu/~mark/puthesis`

Varun, S., Onur, T., Ankit, B. P., & Andrew, J. S. (2020). Deepsimplex: Reinforcement learning of pivot rules improves the efficienty of simplex algorithm in solving linear programming problems.