ON THE BOTTLENECK STRUCTURE OF POSITIVE LINEAR PROGRAMMING

Jordi Ros-Giralt (giralt@reservoir.com), Harper Langston, Aditya Gudibanda, Richard Lethin — Reservoir Labs

SIAM Workshop on Network Science 2019
May 22–23 · Snowbird

Summary

Positive linear programming (PLP), also known as packing and covering linear programs, are an important class of problems frequently found in fields such as network science, operations research, or economics. In this work we demonstrate that all PLP problems can be represented using a network structure, revealing new key insights that lead to new polynomial time algorithms.

Problem Motivation

It is well-known that many traditional network problems—e.g., max-flow, max-min or network utility maximization [1]—can be solved using general linear programming (LP) algorithms. In this essay, we show that the reverse is also true for all *positive* LP problems [4]. For this class of problems, each variable can be interpreted as a flow while each inequality constraint corresponds to a bottleneck link in the corresponding network. This approach reveals new insights on the bottleneck structure of the optimization problem that general approaches such as the simplex or interior-point methods are not able to capture, which can be exploited to design fast algorithms.

Solving PLP Problems Using Networks

A positive linear programming (PLP), also known as packing or covering LP, takes the form $max\{c^\top x : Ax \leq b\}$, where $c \in \mathbb{R}^m_{\geq 0}$, $b \in \mathbb{R}^n_{\geq 0}$, and $A \in \mathbb{R}^{m\times n}_{\geq 0}$. We illustrate how to obtain the network representation of a PLP problem using the Klee-Minty cube as an example:

Example 1. *Klee-Minty cube network structure.* Assume the 3-dimensional Klee-Minty Cube problem [3]: $x \in \mathbb{R}^3$, $c = [4, 2, 1]$, $A = [1, 0, 0; 4, 1, 0; 8, 4, 1]$ and $b = [5, 25, 125]$. We construct an equivalent network with $m = 3$ flows (paths in a graph) and $n = 3$ links (vertices in a graph) as follows. Each variable $x_i, 1 \leq i \leq m$, is represented as the transmission rate of a flow that loops through a link l_j, $1 \leq j \leq n$, as many times as the coefficient $A_{j,i}$ indicates, where each link l_j has a capacity b_j. Fig. 1 shows the network structure of the 3-dimensional Klee-Minty Cube. The original linear programming problem is equivalent to identifying the set of flows $\{x_1, x_2, x_3\}$ in this network such that $c^\top \cdot x$ is maximal.

Our key insight is that by using the network interpretation we can leverage well-known results on the bottleneck structure of communication networks to design new incremental paths within the polytope $Ax \leq b$.

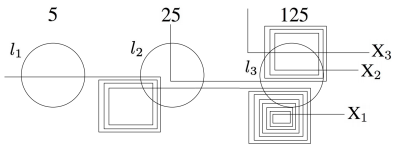

Figure 1: Klee-Minty cube network structure ($m = n = 3$).

Let N be a network constructed from a given PLP problem as shown in Example 1. Assume $N(\Delta)$ is an *extended* network formed by adding a *virtual link* to each flow x_i with a capacity $x_i - \Delta_i$, where $\Delta = [\Delta_1, ..., \Delta_m]$ and $\Delta_i \geq 0$. (Note that by construction, $N([0, ..., 0]) = N$.) In the field of communication networks such virtual links are commonly referred as *traffic shapers* [1], since the value of Δ_i enforces a reduction on the flow transmitted through x_i. A first main contribution of our work is given by the following lemma:

Lemma 1. *Incremental paths.* Let $x^{mm}(N) \in \mathbb{R}^m$ be the max-min solution to network N (see [1]) and let x be an arbitrary feasible solution of network N—i.e., an element in the polytope of the corresponding PLP problem. If $c^\top \cdot x > c^\top \cdot x^{mm}(N)$ for some feasible $x \in \mathbb{R}^m$, then there exists a traffic shaper vector Δ such that $c^\top \cdot x^{mm}(N(\Delta)) > c^\top \cdot x^{mm}(N)$.

We demonstrate that the traffic shapers introduced in Lemma 1 provide incremental paths towards the optimal solution. We denote the direction provided by these incremental paths as *flow gradients*. Under the classic theory of fairness (e.g., [2]), our approach can also be interpreted as a method that travels within the set of Pareto efficient solutions starting at a point that is fair but highly costly, and then moving along the path of the flow gradients to gradually sacrifice fairness until the cost is minimized—thus maximizing $c^\top \cdot x$.

We also demonstrate an efficient method to compute flow gradients through a Pareto-efficient feasible path:

Lemma 2. *Flow gradients.* Using a generalization of the *constrained precedence graph* introduced in [5]—which we call the *flow gradient graph*—flow gradients can be computed in polynomial time.

Leveraging the above two lemmas, the general structure of our proposed algorithm—which we call the *flow gradient algorithm*—is as follows:

1. Set $N^{(0)} = N$, $x^{(0)} = x^{mm}(N^{(0)})$, $i = 1$;
2. While $\Delta^{(i)} \neq \mathbf{0}$ on network $N^{(i-1)}$:
   Set $N^{(i)} = N^{(i-1)}(\Delta^{(i)})$, $x^{(i)} = x^{mm}(N^{(i)})$, $i = i+1$;

Lemma 3. *Complexity.* The flow gradient graph converges to the exact optimal solution in $\mathbf{poly}(m, n)$ steps.

Proofs of Lemma 1, 2 and 3. Omitted for the sake of brevity, see our extended paper https://goo.gl/6gubJi.

Example 2. *Solving the Klee-Minty cube.* To construct the flow gradient graph of a network, we extend the *constraint precedence graph* introduced in [5] in two ways: (1) we include not only the bottleneck links but also the flows as vertices in the graph and (2) we add weights in each edge of the graph according to the coefficients of the matrix A, revealing how the traffic shaping of a flow affects the performance of the rest of the flows. Consider Fig. 2 as an example, which illustrates the execution of the proposed algorithm to resolve the Klee-Minty cube in Example 1. Fig. 2-a presents the initial flow gradient graph, showing that flows x_1, x_2 and x_3 are bottlenecked at links l_1, l_2, l_3, respectively. In our work we show that the bottleneck information provided by the graph reduces the complexity of searching for incremental paths to polynomial time. In the first step (Fig. 2-a), we have $c^\top \cdot x^0 = c^\top \cdot x^{mm}(N) = [4, 2, 1] \cdot [5, 5, 65] = 95$. The algorithm finds a flow gradient by traffic shaping (reducing the value of) flow x_2 with a slope of 3, which we denote as $\nabla_{x_2}(N) = 3$. Applying this gradient takes us to the graph in Fig. 2-b, with $c^\top \cdot x^1 = c^\top \cdot x^{mm}(N([0, 5, 0])) = [4, 2, 1] \cdot [5, 0, 85] = 105$. Next, in Fig. 2-c the algorithm finds a flow gradient by traffic shaping flow x_1 with a slope of 7, $\nabla_{x_1}(N([0, 5, 0])) = 7$. Applying this gradient we obtain the graph in Fig. 2-d and a total value of $c^\top \cdot x^2 = c^\top \cdot x^{mm}(N([5, 5, 0])) = [4, 2, 1] \cdot [0, 0, 125] = 125$. At this point we find no more flow gradients, thus concluding from Lemma 1 that the obtained solution $x = [0, 0, 125]$ is optimal. This is indeed the well-known solution of the Klee-Minty cube for $m = n = 3$. Note that while general purpose algorithms such as simplex would take $2^D = 8$ iterations to resolve this problem (as many as vertices), by exploiting its bottleneck properties, the flow gradient algorithm resolves it in just 2 iterations. Fig. 3 presents the generalized bottleneck structure of the cube for $m = n = D$.

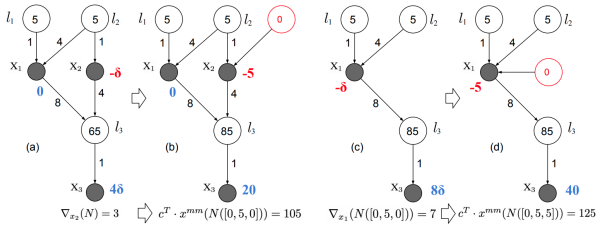

Figure 2: Algorithm steps for Example 2.

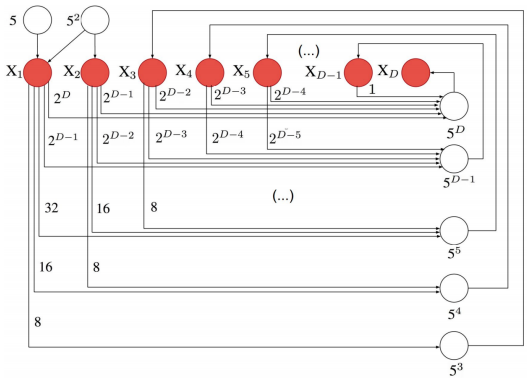

Figure 3: Bottleneck structure of the Klee-Minty Cube.

Discussion

In our talk we will introduce in detail the flow gradient algorithm and demonstrate its polynomial time performance for positive linear programming problems.

References

[1] D. Bertsekas and R. Gallager. *Data Networks (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[2] D. Bertsimas, V. F. Farias, and N. Trichakis. The price of fairness. *Oper. Res.*, 59(1):17–31, Jan. 2011.

[3] V. Klee and G. J. Minty. How good is the simplex algorithm? In O. Shisha, editor, *Inequalities*, volume III, pages 159–175. Academic Press, New York, 1972.

[4] M. Luby and N. Nisan. A parallel approximation algorithm for positive linear programming. In *Proceedings of the Twenty-fifth Annual ACM Symposium on Theory of Computing*, STOC '93, pages 448–457, New York, NY, USA, 1993. ACM.

[5] J. Ros and W. K. Tsai. A lexicographic optimization framework to the flow control problem. *IEEE Transactions on Information Theory*, 56(6):2875–2886, June 2010.