Evaluation of Optimization Methods for Network Bottleneck Diagnosis

Alina Beygelzimer, Jeff Kephart, Irina Rish
IBM T.J. Watson Research Center
19 Skyline Drive, Hawthorne, NY 10532
{*beygel,kephart,rish*}*@us.ibm.com*

Abstract

We consider the problem of localizing network performance bottlenecks and evaluate how various optimization techniques developed for reconstructing link delays perform on this decision problem. We provide some practical suggestions on which approach to use in different scenarios.

Introduction. Timely localization of performance bottlenecks (i.e., links responsible for end-to-end slowdowns) remains one of the top-priority problems in management of distributed computer systems and networks. Since monitoring every single network component often becomes too expensive or infeasible, inference approaches that estimate network characteristics from available measurements (*network tomography* [2]) become increasingly popular.

Despite its similarity to the standard network tomography problem of reconstructing link delays from end-to-end delays, the bottleneck localization is a different problem since we are only interested in identifying the culprit links in order to take corrective actions (i.e., classification or ranking problem) rather than accurately reconstructing all real-valued link delays (regression problem), which is an overkill in our scenario. It turns out that a good solution to the above regression problem may not necessarily be a good solution to the corresponding classification or ranking problem, since due to noise and underdetermined nature of the problem the solution is always an approximation to the ground truth, biased by the choice of the objective function. An interesting question that we investigate here empirically is which optimization approaches (or a combination of them) perform best for the k-worst bottleneck identification problem. We observe that the state-of-art network tomography techniques that solve an L2-norm (sum-squared distance) minimization problems [3] (or, similarly, impose particular priors in probabilistic delay models [2]) may result into less accurate diagnosis than a simple greedy strategy that searches for k bottlenecks directly, when the number of bottlenecks is quite small; however, the situation is reversed when there are many bottlenecks.

Problem Formulation. We assume that $\mathbf{y} \in R^m$ is an observed vector of available end-to-end measurements (e.g., end-to-end delays), $\mathbf{x} \in R^n$ is an unobserved vector of link delays, and $\mathbf{D}$ is a *routing matrix*, also called *dependency matrix*, where $d_{ij} = 1$ if the end-to-end test i goes through the link j, and 0 otherwise. The model is $\mathbf{y} = \mathbf{D}\mathbf{x} + \epsilon$ where ϵ is noise in observations, such as other possible (hidden) causes of delay and/or potential nonlinear effects.[1]

A simple approach to this problem would be just to search for a vector x that best "explains" the delays y in the sense of minimizing some distance $d(y, Dx)$. Particularly, using (squared) L_2-norm (L_p norm is denoted $\|x\|_p$), or sum-squared loss, yields

$$\min_x \|y - Dx\|_2^2,$$

which is equivalent to *ordinary least squares (OLS)* regression where the matrix D is viewed as a collection of datapoints, and x as unknown coefficients. The solution to OLS problem is given in a closed form as $x = (D'D)^{-1}D'y$, where D' denotes the transpose of D.

Another variation on this problem is to add a constraint $x \geq 0$ since link delays are always non-negative. Next, as mentioned above, we can add a regularization term that puts an additional constrained on x, such as bounded L_1 or L_2 norm. For example, the L_1-regularized problem with positivity constraint will look like

$$\min_x \|y - Dx\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \delta, x \geq 0.$$

While the above methods apply in general, the bottleneck identification problem has a specific property that x is expected to be sparse (there are hopefully only a few significant link delays), in which case we may want to search for an (approximate) sparse solution directly:

$$\min_x \|x\|_0 \quad \text{subject to} \quad \|y - Dx\|_2 \leq \delta,$$

for some $\delta > 0$, where the L_0 norm $\|x\|_0$ is just the number of non-zero components in x. However, such formula-

[1] Since the number of tests, m, it typically much smaller than the number of components, n, the problem of reconstructing x is underdetermined, so there is no unique solution, and various *regularization* approaches, such as priors [2] or L_p-norm constraints on $\mathbf{x}$ [2] are typically used.

Fourth International Conference on Autonomic Computing (ICAC'07)
0-7695-2779-5/07 $20.00 © 2007 IEEE

tion requires combinatorial optimization, making the problem computationally intractable. A common solution is to convexify the objective by replacing the L_0-norm with the L_1-norm. The exact version becomes a linear programming (LP), while the approximate version—a quadratic programming (QP) problem with linear inequality constraints. Both can be solved using standard techniques such as interior-point methods or active-set methods. The approximate version

$$\min_x \|y - Dx\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \delta.$$

is essentially the Lasso regression method used in machine learning [4]. In the signal processing community, the approach is known as *Basis Pursuit*.

Greedy Approaches. Another way to avoid the computational hardness of L_0 formulations is to build up solutions greedily, adding one coefficient at a time. Suppose that the matrix has normalized columns, $\|D_i\|_2 = 1$ for all $1 \leq i \leq n$. Starting with the residual $r^{(0)} = y$ and an approximation $\hat{y}^{(0)} = 0$, the algorithm picks a column i_k ($k = 1, \ldots$) maximizing $|\langle r^{(k-1)}, D_i \rangle|$ over all columns i. The approximation is updated using $\hat{y}^{(k)} = \hat{y}^{(k-1)} + x_{i_k} D_{i_k}$, where the coefficient x_{i_k} is fitted using least squares minimization. The new residual is set to $r^{(k)} = r^{(k-1)} - x_{i_k} D_{i_k}$. This variant is called *stagewise regression* in statistics, and is the main variant we use here. An alternative is to minimize the L_2 to the residual instead of maximizing the dot product. Notice that the coefficients are not recomputed at every iteration here.

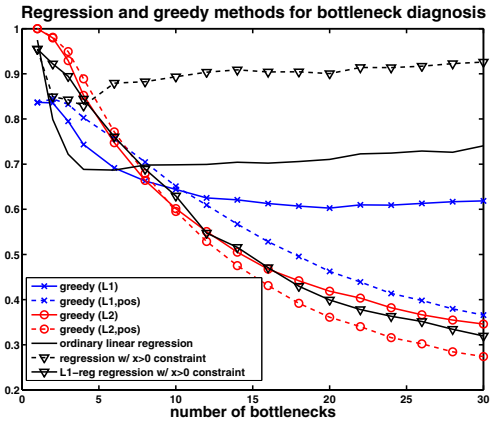

Figure 1. Probability that a bottleneck is successfully found, plotted versus the number of bottlenecks k, for least-squares optimization and greedy approaches.

Experiments. In the attempt to generate realistic dependency matrices, we started with snapshots of the Gnutella Network, maintained by Limewire.org.

To build a dependency matrix, we built breadth first search trees from a small number of randomly selected sources. Assuming that each discovered path is a potential test, we ran a greedy test selection algorithm for isolating single distinguished nodes [1]. The resulting matrix had roughly 200 nodes and 100 tests.

For each number of bottlenecks k, we picked k random nodes (among the ones with at least 6 tests going through them). Each of the selected nodes was assigned a random delay in the interval $[0, 10\text{K}]$. The delays on other nodes were set to 0. End-to-end delays were corrupted by an additive Gaussian noise with mean 0 and standard deviation $\sigma \in \{0.5, 1\}$), scaled by a factor of 100.

Experimental results. Figure 1 demonstrates empirical results for all optimization problems considered above. We plot the probability that a bottleneck is successfully localized versus the number of bottlenecks. We compare first the least-squares regression methods: ordinary regression (OLS), regression with positivity constraint $x \geq 0$, and with both the L_1-norm regularization and the positivity constraint. Note that for small number of bottlenecks, the L_1 regularization with positivity constraint $x \geq 0$ outperforms the OLS regressions, with or without the positivity constraint, although for larger number of bottlenecks the situation is reversed. Also, it looks like the positivity constraint hurts unregularized regression (OLS versus OLS with positivity constraint). Next, we experimented with several greedy algorithms: greedy with L_1- and L_2-norm, with and without $x \geq 0$ constraints. When the number of bottlenecks is sufficiently small, the L_2 variant performs better, but once the number of bottlenecks is about 10, the L_1 variant starts winning[2]. Overall, we can suggest the following winning combination of the above methods: when the number of bottlenecks is quite small, which is typically the case in practice (e.g., less than 5-6 in our experiments), it is best to use the greedy method with L_2 norm (adding the positivity constraint may help just a little), while for larger number of bottlenecks, OLS with the positivity constraint should be used. Of course, a more extensive empirical evaluation on various networks is required to provide a stronger support for our observations.

References

[1] M. Brodie, I. Rish, and S. Ma. Optimizing probe selection for fault localization. In *Distributed Systems Operation and Management*, 2001.

[2] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499–517, 2004.

[3] H. Song, L. Qiu, and Y. Zhang. NetQuest: A Flexible Framework for Large-Scale Network Measurement. In *ACM SIGMETRICS-06*, 2006.

[4] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society, Series B*, 58(1):267–288, 1996.

[2]Note that the positivity constraint in both versions becomes harmful as the number of bottlenecks increases, which can be due to the fact that the coefficients obtained at earlier steps of the greedy are not readjusted and the algorithm may "over-blame" the components choosen at earlier steps.