NETWORK PROBE SELECTION PROBLEM

PROJECT REPORT FOR CS357

Presented by

Kunal Gupta (150001015)

Bhor Verma (150001005)

Computer Science and Engineering

3rd Year

Under the Guidance of

Dr. Kapil Ahuja

Department of Computer Science and Engineering

Indian Institute of Technology Indore

Autumn 2017

CONTENTS

INTRODUCTION

A defining characteristic of today's information age is our reliance on data centers. The scale and complexity of these data centers have been increasing rapidly. This in turn is limiting our ability to understand and control the data center operations.

Availability of good quality monitoring data is a vital need for management of today's data centers. However, effective use of monitoring tools demands an understanding of the monitoring requirements that system administrators most often lack. Instead of a well-defined process of defining a monitoring strategy, system administrators adopt a manual and intuition-based approach.

A quintessential requirement for development of automated analytics-led solutions for understanding of the as-is state of these systems is the availability of good quality monitoring data. It is very important to systematically deploy a monitoring framework to capture behavioral characteristics of all compute, communication, and storage components. Monitoring metrics need to be collected at various layers ranging from the hardware metrics such as CPU and memory utilization, to operating system and virtual machine layers, to database and application layers, among others. The good news is that a lot of monitoring tools exist that can monitor a wide variety of system components. Most of these tools are capable of monitoring a large range of metrics and can be configured as per needs.

However, the bad news is that the quality of the monitored data is often a suspect. Effective use of the monitoring tools demands an understanding of monitoring requirements that system administrators most often lack. Instead of a well-defined process of defining monitoring strategy, system administrators adopt an ad-hoc, manual, and intuition-based approach. This leads to inconsistent and inadequate data collection and retention policies.

For this project, we propose a solution to adapt the monitoring level of a system component using probes[1].

[1] Probes refer to test transactions that are sent in the network. The success and failure of these probes depend on the success and failure of the network components used by the probes. An example of a simple probing technique is to ping all nearby nodes and check the response time and packet loss.

MATHEMATICAL FORMULATION OF THE PROBLEM

PROBLEM STATEMENT

The key idea is to send probes in the network, infer the system state from the probe results, and adjust the monitoring of system components based on the inferred system state. The key objective of the problem is:

1. Probe selection - how to select the right set of probes such that criticality of components can be correctly estimated and appropriate recommendations can be provided for each component.

ASSUMPTIONS

Network is like an undirected graph, with each node in graph denoting a node in the network.

We are given:

- List of probes. (We can select out of these)
- List of nodes monitored by each node.
- Minimum number of probes by which each node has to be monitored. (For data consistency check).

NOTATIONS

Let the set of all probes be $P$and p represent any probe in P. N is the total number of probes.

Then,

- $nodes(p)$ represents all the nodes that p can monitor,
- $C(p)$ is the cost associated with probe p, where cost is defined as length of the probe = $|nodes(p)|$,
- and c is the minimum number of probes that monitor each node.

COST FUNCTION

Cost is defined as the length of the probe where length is the total number of nodes that can be monitored by any given probe. Probes with low cost are preferred.

CONDITIONS

Following are the set of conditions which need to be fulfilled for our objective:

- ***Condition 1*** *– The average number of probes probing each node:*
  $$\frac{\left(\sum_{p\in P_{sel}}|nodes(p)|\right)}{N}$$
  should be maximized.
- ***Condition 2*** – The average length of a probe:
  $$\frac{\left(\sum_{p\in P_{sel}}|nodes(p)|\right)}{|P_{sel}|}$$ should be minimized.
- ***Condition 3*** – Every node must have at least one probe passing through it.
  $$\bigcup_{p\in P_{sel}}\left(\left(nodes(p)\right)\right) = N$$
- ***Condition 4*** – In order to minimize the probe traffic, the number of selected probes:
  $|P_{sel}|$ should be minimized.

OBJECTIVE FUNCTION

We propose the following linear program that includes the four stated objectives in the form of constraints and objective functions:

$$min \sum_{p\in P} x_p \times C(p)$$

$$s.t. \sum_{p\in P}\left(x_p \times p_n\right) \geq c \quad \forall n \in N$$

$$x_p \in \{0,1\} \quad \forall p \in P$$

In the formulation, x_p, is a binary variable indicating whether probe p is selected (x_p = 1) or not (x_p = 0). *C(p) = nodes(p)*. p_n is the variable indicating the presence of node n in the path of probe p. p_n is 1 if node $n \in$ *nodes(p)* and 0, otherwise.

In other words,

$$\mathrm{x_p} = \begin{cases} 1, & probe\ p\ is\ selected \\ 0, & otherwise \end{cases}$$
$$\mathrm{p_n} = \begin{cases} 1, & n\ \in nodes(p) \\ 0, & otherwise \end{cases}$$

The objective function tries to minimize the cost of all selected probes. It satisfies the third and the fourth requirements of the minimum length and minimum number of probes, respectively. The constraint enforces that each node n in probe p should get probed at least c times, thereby satisfying the second requirement (average coverage). The value of c can be tuned based on the system requirements.

CONVERSION TO MATRIX FORM

OBSERVATION

p_n can be represented by a matrix $\mathrm{F}_{p\times n}$ having binary values. Thus,

$$p_n = F[p][n]$$

Hence, $nodes(p)$ is the pth row of F and so,

$$C(p) = \sum_{n\in N} F[p][n]$$

Now, $C(p)$ can be aggregated into a single vector of P dimensions just like x_p:

$$C = [C(p)]_{p\times 1}$$

$$x = [x_p]_{p\times 1}$$

REFORMULATION

From the previous observations, we can reformulate our problem in the following form:

$$\min C^T x$$

$$s.t.,\ F^T x \geq k$$

$$where\ k\ is\ c \cdot \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1}$$

$$\&\ each\ dimension\ of\ x\ can\ take\ values \in \{0,\ 1\}$$

ALGORITHMS

Some of the other Methods we studied to solve our problem are:

- Branch and Bound
- Balas Additive Algorithm (Specialized Branch and Bound)

- Interior Point Method
- Lagrange and Subgradient

Since the other methods are out the scope of the current course, we have focused mainly on the Balas Additive Algorithm.

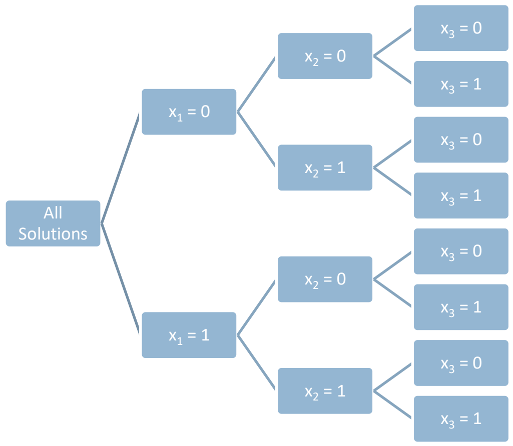
NAÏVE WAY

The first thing one might think of for solving binary-integer problems is to simply enumerate all of the possible solutions and then to choose the best one. This will actually work for very small problems, but it very rapidly becomes unworkable for even small- to medium-size problems, let alone industrial-scale problems.

For example, consider a binary 0/1 problem that has 20 variables. This will have 2^{20}=1,048,576 solutions to enumerate, which might be possible to do via computer. Now consider a slightly larger binary problem having 100 variables. Now there are $2^{100}=1.268 \times 10^{30}$ solutions to enumerate, which is likely impossible, even for a very fast computer. The combinatorial explosion is even worse for general integer variables that can take on even more values than just the two possibilities for binary variables. It is easy to construct problems in which the number of solutions is greater than the total number of atoms in the universe! Enumeration just won't work here – we need a better way of tackling the combinatorial explosion, which is when Branch and Bound comes in.

BALAS : SPECIALISED BRANCH AND BOUND

Consider the complete enumeration of a model having 3 binary variables x_1, x_2 and x_3, whose ranges are $0 \le x_1 \le 1$, $0 \le x_2 \le 1$ and $0 \le x_3 \le 1$.



Now the main idea in branch and bound is to avoid growing the whole tree as much as possible, because the entire tree is just too big in any real problem.

Instead, branch and bound grows the tree in stages, and grows only the most promising nodes at any stage.

There are mainly three processes we follow:

- Branching – a node is selected for further growth and the next generation of children of that node is created.
- Bounding – bound on the best value attained by growing a node is estimated.
- Pruning – cut off and permanently discard nodes when one can show that a node, or any of its descendants, will never be either feasible or optimal. This is precisely what prevents the search tree from growing too much.

THE PROCESS

Let at any intermediate point in the algorithm, we have the current version of the branch and bound tree, which consists of bud nodes (partial solutions, either feasible or infeasible) labeled with their bounding function values and other nodes.

1. The node selection policy now governs how to choose the next bud node for expansion. We used the best-first selection policy: choose the bud node that has the best value of the bounding function anywhere on the branch and bound tree. This means choosing the bud node with the smallest value of the bounding function.
2. Once a bud node has been chosen for expansion we choose the next numbered variable as the next variable to use in creating the child nodes of the bud node.
3. The method for showing that no descendent will be optimal is standard: if the bud node bounding function value is worse than the objective function value for the best complete feasible solution found so far, then the bud node can be pruned.
4. We finally stop when there are no more bud nodes left to consider for further branching.

One thing to note here is if we cannot set all of the variables to 0 without violating one or more constraints, then we prefer to set the variable that has the smallest index to 1. This is because the variables are ordered so that those earlier in the list increase the objective function by the smallest amount.

REFERENCES

- Adaptive Monitoring: Application of Probing to Adapt Passive Monitoring - Deepak Jeswani, Maitreya Natu, R. K. Ghosh (Journal of Network and Systems Management – October 2015)
- Adaptive path selection for link loss inference in network tomography applications - Y Qiao, J Jiao, Y Rao, H Ma (PloS one - 2016)