Bachelor Thesis
The Server Location Problem with Restricted Loads on Servers and Links

Alexander Svozil

October 3, 2013

1 Abstract

The server location problem with restricted loads on servers and links (SLRL) is an NP-Complete problem, introduced by Hiroyoshi Miwa et al. in the Paper "Method of Locating Mirror Servers to Alleviate Load on Servers and Links" [1]. The problem came up, because the massive volume of data distributed by content delivery networks (CDNs) require well located mirror servers in order not to badly influence the quality of their service. Two examples for CDNs would be "Amazon CloudFront" a traditional commercial CDN, or the "AT&T Inc." a Telco CDN which has advantages over traditional CDNs because they own the so called "last mile", the final leg of the telecommunications networks. CDN nodes are usually deployed in multiple locations, often over multiple backbones, reaching thousands of nodes with tens of thousands of servers. [2] [3] The two constraints induced by the choice of mirror servers is the number of maximum nodes accessing a mirror server and maximum number of nodes accessing a mirror server through a specific link. The constraint mentioned first, corresponds to the network load on the (mirror-)servers. The second constraint corresponds to the restricted load on the links. First I want to prove that SLRL is NP-complete. Next, I want to propose my own algorithm as a sequel to the existing greedy algorithm proposed by Miwa et al. [1].

Contents

2 Introduction

2.1 CDNs

"A content delivery network is a collaborative collection of network elements spanning the Internet, where content is replicated over several mirrored Web servers in order to perform transparent and effective delivery of content to the end user."[4, p. 3] To perform efficient delivery of data, CDNs need to have several mirrored Web servers and thus it is important to place them well, because placing the mirror server could be an expensive choice considering no prior calculation where it should go. The expensive choice manifests in high link usages and or a maximal neighbourhood being too big for the server to handle. The mirror servers naturally serve the same data, as they are there to alleviate the load of the other servers providing the data.

2.2 Similar Problems

A lot of thought has already been put into the server location problem. Two problems that describe this problem would be the p-center and the p-median problem. The purpose of these problems is mainly to shorten delay time from the client to the server.

2.2.1 P-Center Problem

"The p-Center problem consists of locating p facilities and assigning clients to them in order to minimize the maximum distance between a client and the facility to which he or she is allocated. It is used for example in locating fire stations or ambulances, where the distance from the facilities to their farthest assigned potential client should be minimum."[5] The p-center problem is one of the best known NP-hard discrete location problems.[6] For large instances, it is impractical to solve the problem exactly. There have been very good results with the tabu-search and variable neighborhood search. [5]

2.2.2 P-Median Problem

The p-Median Problem is very similar to the p-center problem. It can be defined as follows: given a graph G=(V,E) we are asked to find a set of nodes, S, of size p, where $S \subset V$, such that the weighted sum of the distances from the remaining nodes {V-S} to the Set S is minimized. [7] This problem belongs to the class of problems known as NP-hard.[8] There have been approaches to solve this problem with the tabu search [7] and solving the linear program by relaxing the binary constraints.[9]

2.2.3 NA (Node to Area)-connectivity

2.3 Open Shortest Path First

3 Server Location Problem with Restricted Loads on Servers and Links

3.1 Formal Definition

3.1.1 Neighbour Set

The neighbour set is the set, that belongs to a server. In our problem, it is defined as follows: If the node has a shortest path to a server node, he belongs to this server nodes neighbour set. If the node has several shortest paths to multiple server nodes, it will belong to all of those server nodes neighbour set. The server itself will belong to its own neighbour sets.

3.1.2 Load on a Link, m(e) or multiplicity of an edge

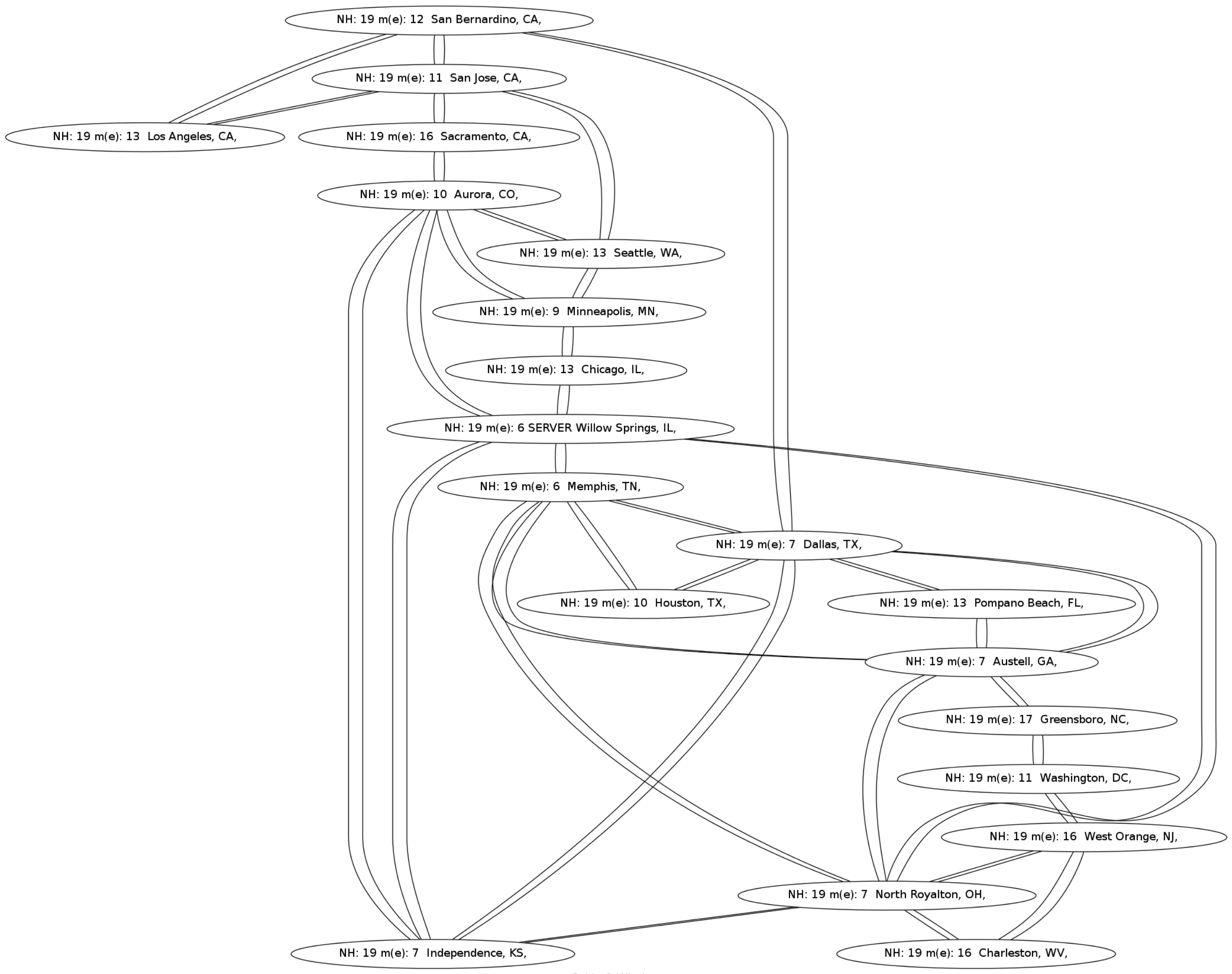
The load on a link is defined as follows: It is defined as the number of shortest paths, that cross a link. If there are several shortest paths with the same length to one or more servers every used edge is taken into account. This is done, because we must always consider the worst case when locating a server.

3.1.3 The Problem

Given: *Undirected graph G(V,E) and positive integers k,r and c.*
$k = \|S\|$, *where S is the set of servers* $S = \{s_1, s_2, \ldots s_k\} \subseteq V$
$r \geq \|V_i\|$ $(i = 1, 2, \ldots, k)$ *where* V_i *is the neighbour set of* s_i $(i = 1, 2, \ldots, k)$
$c \geq m(e)$, *where m(e) is the maximum load* $\forall e \in E$.

Wanted: *A set of Servers* $S = \{s_1, s_2, \ldots, s_k\} \subseteq V$ *such that* $|V_i| \leq r$ $(i = 1, 2, \ldots, k)$ *where* V_i *is the neighbour set of* s_i $(i = 1, 2, \ldots, k)$ *and that* $m(e) \leq c$ $(\forall e \in E)$.

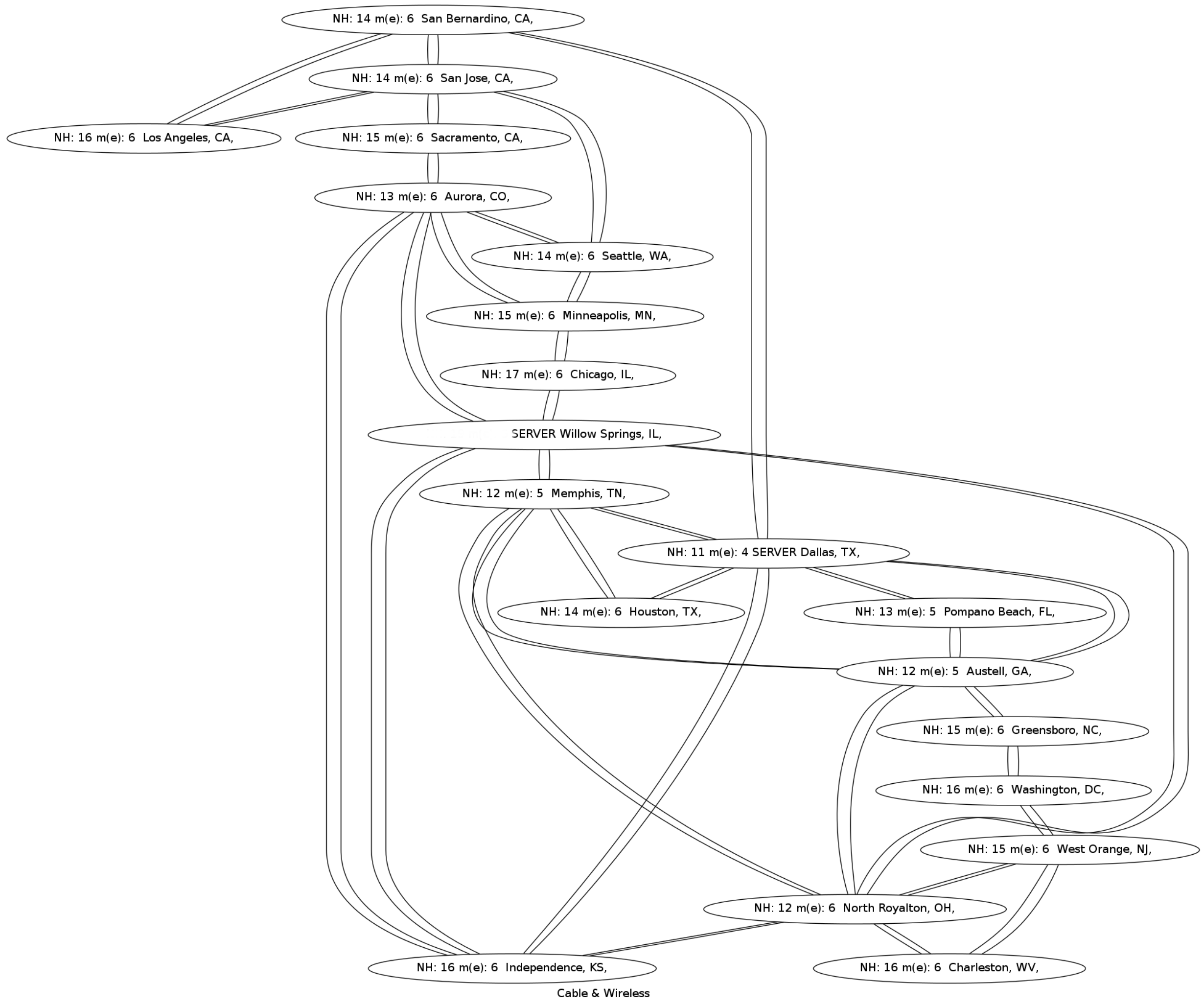
3.2 An Instance Of The Problem


NH: 19 m(e): 12 San Bernardino, CA,
NH: 19 m(e): 11 San Jose, CA,
NH: 19 m(e): 13 Los Angeles, CA,
NH: 19 m(e): 16 Sacramento, CA,
NH: 19 m(e): 10 Aurora, CO,
NH: 19 m(e): 13 Seattle, WA,
NH: 19 m(e): 9 Minneapolis, MN,
NH: 19 m(e): 13 Chicago, IL,
NH: 19 m(e): 6 SERVER Willow Springs, IL,
NH: 19 m(e): 6 Memphis, TN,
NH: 19 m(e): 7 Dallas, TX,
NH: 19 m(e): 10 Houston, TX,
NH: 19 m(e): 13 Pompano Beach, FL,
NH: 19 m(e): 7 Austell, GA,
NH: 19 m(e): 17 Greensboro, NC,
NH: 19 m(e): 11 Washington, DC,
NH: 19 m(e): 16 West Orange, NJ,
NH: 19 m(e): 7 North Royalton, OH,
NH: 19 m(e): 7 Independence, KS,
NH: 19 m(e): 16 Charleston, WV,

Cable & Wireless

The above shown graph is from the barebone network "Cable and Wireless". It is parsed out of a dataset of barebone networks provided by caida.[10] The nodes have two values in their bubbles: "NH and m(e)". NH stands for the maximum neighbourhood if that node was chosen as a server. Choosing the first server obviously implicates the maximum neighbourhood, because there are no mirror servers yet on the graph.
The multiplicity of an edge e, $m(e)$ is explained above. The number after $m(e)$ in the bubble describes $max(m(e))\ \forall e \in E$, if one decides to take

this node as a server. The greedy algorithm takes the node with the lowest $max(m(e))$ first.


NH: 14 m(e): 6 San Bernardino, CA,
NH: 14 m(e): 6 San Jose, CA,
NH: 16 m(e): 6 Los Angeles, CA,
NH: 15 m(e): 6 Sacramento, CA,
NH: 13 m(e): 6 Aurora, CO,
NH: 14 m(e): 6 Seattle, WA,
NH: 15 m(e): 6 Minneapolis, MN,
NH: 17 m(e): 6 Chicago, IL,
SERVER Willow Springs, IL,
NH: 12 m(e): 5 Memphis, TN,
NH: 11 m(e): 4 SERVER Dallas, TX,
NH: 14 m(e): 6 Houston, TX,
NH: 13 m(e): 5 Pompano Beach, FL,
NH: 12 m(e): 5 Austell, GA,
NH: 15 m(e): 6 Greensboro, NC,
NH: 16 m(e): 6 Washington, DC,
NH: 15 m(e): 6 West Orange, NJ,
NH: 12 m(e): 6 North Royalton, OH,
NH: 16 m(e): 6 Independence, KS,
NH: 16 m(e): 6 Charleston, WV,
Cable & Wireless

This graph shows the second server pick. This time our $m(e)$ is already low enough to suffice the constraint $m(e) \leq c$ thus the greedy algorithm proposed in the paper by Miwa et al. [1] chooses the node with the lowest max neighbourhood which is coincidentally also the one with the lowest $m(e)$.

3.3 Proof of NP-Completeness

4 Algorithm

4.1 Rules and Assumptions

4.2 Other Algorithms

The only prior algorithm for this problem is the greedy algorithm proposed by Miwa et al. [1]. It greedily chooses the node with the lowest maximal neighbourhood if the constraint $c \geq m(e)(\forall e \in E)$ is fulfilled. Otherwise it will choose a node where $min(m(e)(\forall e \in E))$ is true.

4.3 Data (Testinstances)

The data taken for the tests is from caida [10]. Parsing the data was one of the most difficult parts, because the data is not formatted well. It does not really follow the syntax proposed in the beginning of the file. I also modified the data in some concerns:

- If the graph G given for the barebone network was not connected, the connected component with less nodes of that graph is cut out.
- Normally the connected cities are listed in this manner: "city,state,country", in order to match with the numbers in the paper [1] only city and state are used for identification and the country is left out. Sometimes London is used on one hand as City in Canada and on the other hand in GB, denoted like this London, *, Canada and London, *, England. They are differentiated.
- Sometimes the State has a number. One example would be "Kansas City, KS2 " This is interpreted as "Kansas City, KS".

All those modifications were done to match the graphs proposed by [1]. As I did not have the data of the other paper used, I tried to match them with the number of nodes and number of vertices. Unfortunately three instances of my instances do not match the number of nodes and / or vertices proposed from the paper above. The varying instances are: "Sprint", "Cable and Wireless" and "UUNet".

5 Implementation

I decided to implement a tabu-search.

5.1 Determination of the neighbour sets of a server and loads of the links

When a server is added to the graph, r and $max(m(e))$ $\forall e \in E$ need to be recalculated, as the shortest paths from the clients to the servers change. This has to be done for every client in order to ensure that every client has its path(s) and server(s). The outer loop goes through every n that is not a server and tests out every server if it is the closest. There can be more than one closest server. It will be saved in the nearestServersList. The path to the server is then marked. If there is a new nearest Server, these stats are reset and the newly found nearest servers edges are marked. It takes $\mathcal{O}(|V| - |S| * |S| * |V + E|^2)$ to calculate the new r and the new $max(m(e))$ $\forall e \in E$, as BFS costs $\mathcal{O}(|V| + |E|)$ and backtracking every path can also result in visiting every node and every edge and thus costs $\mathcal{O}(|V| + |E|)$.

Algorithm 1 constraintsCalculation

```
Input: Graph (V,E) G, Servers S
for Each node n $\in G$ do
    if $!(n \in S)$ then
        $nearestServersList \leftarrow \{\};$
        $minServer \leftarrow 0;$        ▷ The stepcounter to the nearest server(s)
        for Each Server $s \in S$ do
            $BFS2(n, s)$
            $p \leftarrow s;$
            $i \leftarrow s.getDistance()$        ▷ distance to server acquired in BFS
            if $i \leq minServer || nearestServersList.isEmpty()$ then
                if $i < minServer$ then
                    for Each Server prevNearest $\in nearestServersList$ do
                        $prevnearest.tmpNeighbourhood = 0;$
                        $nearestServersList.clear();$
                        $usedEdges.clear();$
                    end for
                end if
                $minServer = i;$        ▷ new nearest server found, set length
                $nearestServersList.add(s);$  ▷ add it to the list of nearest servers
                $s.tmpNeighbourhood+ = 1;$     ▷ adds one to the temporal neighbourhood (it may get reset later on)
                $markPath2(s)$ ▷ This method recusively walks back every node's parents list to determine every path and add it to the usedEdges list.
            end if
        end for
        for Each Server s $\in nearestServerList$ do
            $s.neighborhood+ = s.tmpNeighbourhood;$
            $s.tmpNeighbourhood = 0;$
        end for
        $markEdges();$                ▷ increments usage of each usedEdges
    end if
end for
```

5.2 Modified BFS

The Breadth first search needed to be modified in order for it to be able to find all shortest paths from one node to another, as we need to consider every shortest path in order to simulate the worst case scenario regarding load on links. It's runtime $\mathcal{O}(|V| + |E|)$ is unmodifed.

Algorithm 2 BFS2

```
Input: Graph(V,E) G, Node dest, Node start
    $clearParents()$;
    $resetDistance()$;
    Queue<Node> nodeQueue ← {};
    Set<Node> visited ← {};
    $visited.add(start)$;
    $nodeQueue.offer(start)$;    ▷ Nodepair saves the node and the parentnode, so all the paths can be reconstructed
    while $!nodeQueue.isEmpty()$ do
        $Node\ curNode = nodeQueue.poll()$;    ▷ get first node out of queue
        $visited.add(curNode)$;    ▷ mark node as visited
        if $!curNode.equals(dest)$ then
            $return\ dest$;
        end if
        for $Edge\ e : curNode.getEdges()$ do
            if $!visited.contains(e.getNode2())$ then
                if $e.getNode2().getDistance() == curNode.getDistance() + 1 \| e.getNode2().getDistance() == -1$ then
                    $alreadyVis \leftarrow false$;
                    for $Node\ node : nodeQueue$ do
                        $node.addParent(curNode)$;
                        $alreadyVis \leftarrow true$;
                        $break$;
                    end for
                    if $!alreadyVis$ then
                        $nodeQueue.offer(e.getNode2())$;
                        $e.getNode2().addParent(curNode)$;
                        $e.getNode2().setdistance(curNode.getDistance() + 1)$;
                    end if
                end if
            end if
        end for
    end while
```

5.3 The local search

The Local search simply exchanges a server with a client and recalculates the constraints to create a neighborhood N of feasible solutions.

```
Algorithm 3 localsearch
Input: Graph(V,E), Set
```

5.4 The tabu-search

5.5 Networks and the approximation ratio

6 Conclusion

References

[1] R. Nakamura and H. Miwa, "Method of locating mirror servers to alleviate load on servers and links," in *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*, pp. 513–518, 2011.

[2] Wikipedia, "Content delivery network," 2013. [Online; accessed 4-September-2013].

[3] Wikipedia, "Last mile," 2013. [Online; accessed 4-September-2013].

[4] R. Buyya, M. Pathan, and A. Vakali, *Content Delivery Networks.* Springer Publishing Company, Incorporated, 1st ed., 2008.

[5] N. Mladenovic, M. Labbé, P. Hansen, E. Commerciales, and E. Commerciales, "Solving the p-center problem with tabu search and variable neighborhood search," 2000.

[6] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. I: The \$p\$-centers," *SIAM Journal on Applied Mathematics*, vol. 37, pp. 513–538, 1979.

[7] E. Rolland, D. A. Schilling, and J. R. Current, "An efficient tabu search procedure for the p-median problem," *European Journal of Operational Research*, vol. 96, no. 2, pp. 329 – 342, 1997.

[8] O. Kariv and S. L. Hakimi, "An algorithmic approach to network location problems. I: The \$p\$-medians," *SIAM Journal on Applied Mathematics*, vol. 37, pp. 539–560, 1979.

[9] K. E. Rosing, C. ReVelle, and H. Rosing-Vogelaar, "The p-median and its linear programming relaxation: An approach to large problems," *Journal of the Operational Research Society*, pp. 815–823, 1979.

[10] Caida, "Barebones - http://www.caida.org/tools/visualization/mapnet/data/inc.txt," 2013. [Online; accessed 9-September-2013].