Multidimensional facility location problem in Content Distribution Networks

Vivek Relan and Bhushan Sonawane
Computer Science and Electrical Engineering Department,
University of Maryland, Baltimore County
{relan1|bhushan1}@umbc.edu

4th May 2009

Abstract

Assignment problem in content distribution networks (CDN) is unsplittable hard-capacitated facility location problem (UHCFLP) because of underlying anycast routing protocol. We extend assignment problem in CDN to multidimensional facility location problem (MFLP). With our devised algorithm, assignment cost considers CPU utilization parameter along with request size and distance between client and server. The importance of incorporating CPU utilization parameter into assignment cost enables in real-time modeling of CDN. Therefore, additional requests are not assigned to overloaded server, even though network bandwidth is available. We observe that approximating MFLP is NP-hard without violating capacity constraints. Therefore, we relax the capacity constraints by \((1 + \epsilon)\) factor, for some \(\epsilon > 0\). For approximating MFLP in CDN, we use tree metric and devise \((1 + \epsilon)\) - approximation algorithm which violates capacity constraints by \((1 + \epsilon)\) factor, for some \(\epsilon > 0\).

Keywords:
Content Distribution Networks, Assignment problem, Unsplittable hard-capacitated facility location problem, Facility-location problem, Multidimensional assignment problem

1 Introduction

Content Distribution Network (CDN) is an effective approach to improve Internet service quality. CDN replicates the content from the place of origin to the replica servers scattered over the Internet and serves a request from a replica server close to where the request originates. CDN can be modeled as follows, assume an autonomous system (AS) with \(n\) servers, where each server can serve certain number of requests per unit time. A request enters the system through ingress Provider Edge (PE) routers. Cost for serving one particular request will depends upon distance between server and PE within the AS. CDN providers always aim at optimizing the connection cost over all input requests. Bateni [1] proved that approximating assignment problem in CDN is NP-hard without violating capacity constraints. According to Alzoubi et al [10], CDN provider’s optimization problem is nothing but special case of Generalized Assignment Problem (GAP), wherein it tries to minimize overall connection cost for all of the input requests. Bateni [1] mapped this CDN problem to unsplittable hard-capacitate facility location problem which provides solution with total connection cost at most \((1 + \epsilon)\) times the optimum with violation of capacities by at most \((1 + \epsilon)\) factor, for...
some $\epsilon > 0$.

Bateni’s solution considers network bandwidth constraint, but does not consider CPU computing capacity of servers. We extend Bateni’s solution to incorporate CPU computing capacity of servers. Real time modeling of CDN is possible with incorporation of CPU utilization parameter into assignment cost constraints. According to Betani’s solution, client request may get assigned to CPU overloaded server which has network bandwidth to satisfy the request. However, such assignment is incorrect. Therefore, there is a need to consider CPU capacity of server while assigning a request.

With incorporation of CPU computing capacity of each server as new parameter, old model of facility location problem in CDN gets mapped to multi-index quadratic assignment problem. In our new model, we consider CPU computing capacities while finding out optimal connection cost for serving input requests.

This enables in minimizing the total assignment cost by modeling content distribution networks with parameters: request size, CPU utilization, and distance between client and server. Our multi-index unsplittable hard-capacitated facility location problem which includes CPU computing capacity as added parameter can be modeled as follows:

$$\min_{\pi} \sum_{j \in J^*} f_j + \sum_{i \in B} n_i c_i d_{i, \pi(i)}$$

We observe that multi-dimensional facility location problem in CDN is NP-hard to approximate without violating capacity constraints. Therefore, we relax capacities to $(1 + \epsilon)$ for some $\epsilon > 0$. In order to solve this problem, we use tree metrics [13] which gives $(1 + \epsilon)$ approximation ratio by relaxing capacity constraints to $(1 + \epsilon)$ factor, for some $\epsilon > 0$.

2 Background

This section provides the background knowledge to understand a paper.

2.1 Overview of various assignment problems

The assignment problem [8] is one of the fundamental combinatorial optimization problems in the branch of optimization or operations research in mathematics. It consists of finding a maximum weight matching in a weighted bipartite graph. In its most general form, the problem is as follows: there are a number of agents and a number of tasks. Any agent can be assigned to perform any task, incurring some cost that may vary depending on the agent-task assignment. It is required to perform all tasks by assigning exactly one agent to each task in such a way that the total cost of the assignment is minimized. Here, we discuss various forms of assignment problems.

- Linear Assignment problem (LSAP)

  The Linear Assignment Problem is one of the most famous problems in linear programming and in combinatorial optimization. Informally speaking, we are given an n x n cost matrix $C = (c_{ij})$, and we want to select n elements of C, so that there is exactly one element in each row and one in each column, and the sum of the corresponding cost is minimum.

  The linear sum assignment problem (LSAP) can then be stated as

  $$\min_{\phi \in S_n} \sum_{i=1}^{n} C_{i, \phi(i)}$$
Mathematical model: By introducing a binary matrix $X = (x_{ij})$ such that:

- $X_{ij} = 1$ if row $i$ is assigned to column $j$,
- $X_{ij} = 0$ Otherwise

LSAP can be modeled as:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \cdot x_{ij}$$

such that

$$\sum_{j=1}^{n} x_{ij} = 1(i = 1, 2, 3, \ldots, n)$$

$$\sum_{i=1}^{n} x_{ij} = 1(j = 1, 2, 3, \ldots, n)$$

$$x_{ij} \in \{0, 1\} (i, j = 1, 2, 3, \ldots, n)$$

• **Quadratic Assignment Problem (QAP)**

The Quadratic Assignment Problem is one of the most challenging combinatorial optimization problems. We want to assign $n$ facilities to $n$ locations with the cost being proportional to the flow between the facilities multiplied by the distances between the locations, plus eventually costs for placing the facilities at their respective locations. The objective is to allocate each facility to a location such that the total cost is minimized. Objective function of Quadratic Assignment Problem:

$$\min_{\phi \in S_n} \sum_{i=1}^{n} \sum_{k=1}^{n} a_{ik} \cdot b_{\phi(i) \phi(k)} + \sum_{i=1}^{n} C_{i \phi(i)}$$

Mathematical model:

We are given three $n \times n$ matrices: the flow matrix $A=(a_{ij})$, the distance matrix $B=(b_{kl})$ and matrix $C=(c_{ik})$, where $c_{ik}$ is the cost of placing facility $i$ at location $k$.

Let,

- $X_{ij} = 1$ if facility $i$ is assigned to location $j$,
- $X_{ij} = 0$ Otherwise

QAP can be modeled as,

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} a_{ik} \cdot b_{jk} \cdot x_{ij} \cdot x_{kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} C_{ij} \cdot x_{ij}$$

such that

$$\sum_{j=1}^{n} x_{ij} = 1(i = 1, 2, 3, \ldots, n)$$

$$\sum_{i=1}^{n} x_{ij} = 1(j = 1, 2, 3, \ldots, n)$$

$$x_{ij} \in \{0, 1\} (i, j = 1, 2, 3, \ldots, n)$$
Multi-index Assignment Problems

In the case of 3-index assignment problems, there are two main models: the axial 3-index assignment problem and the planar 3-index assignment problem.

\[
\min_{\phi, \psi \in S_n} \sum_{i=1}^{n} C_{i\phi(i)\psi(i)}
\]

Axial 3-index Assignment Problem

We are given an \( n \times n \times n \) matrix \( C = (c_{ijk}) \), and we want to select \( n \) elements of \( C \), so that there is exactly one element in each two-dimensional face (in the three orientations), and the sum of the corresponding costs is minimum.

Mathematical model:

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk}
\]

such that

\[
\min \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ijk} (i = 1, 2, ..., n)
\]

\[
\min \sum_{i=1}^{n} \sum_{k=1}^{n} x_{ijk} (j = 1, 2, ..., n)
\]

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijk} (k = 1, 2, ..., n)
\]

\( x_{ijk} \in \{0, 1\} (i, j, k = 1, 2, 3, ..., n) \)

Planar 3-index Assignment Problem

We are given an \( n \times n \times n \) matrix \( C = (c_{ijk}) \), and we want to select \( n^2 \) elements of \( C \), so that there is exactly one element in each one-dimensional line (in the three orientations), and the sum of the corresponding costs is minimum.

Mathematical model:

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} c_{ijk} x_{ijk}
\]

such that

\[
\min \sum_{j=1}^{n} \sum_{k=1}^{n} x_{ijk} (i = 1, 2, ..., n)
\]

\[
\min \sum_{i=1}^{n} \sum_{k=1}^{n} x_{ijk} (j = 1, 2, ..., n)
\]

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ijk} (k = 1, 2, ..., n)
\]

\( x_{ijk} \in \{0, 1\} (i, j, k = 1, 2, 3, ..., n) \)
3 Related work

Assignment problem is prime interest of this project. Topic is very well researched in literature [1, 2, 3, 5]. The generalized assignment problem (GAP) can be viewed as a scheduling problem on parallel machines, where each machine has a capacity and each job has a size and a cost, each possibly dependent on the machine to which it is assigned, and the objective is to minimize the total cost incurred. The problem is NP-hard [9] and has been a central problem in operation research since 1975 [10]. Shmoys and Tardos [11] showed that in polynomial time, given a value C, we can either decide that no feasible solution of cost C exists, or else find a schedule of cost at most C where the load on each machine is at most twice that of the optimum.

The facility location problem is another central problem in operations research. It is also well studied in the field of approximation algorithms, and a number of different approximation algorithms have been proposed for this problem using a variety of techniques [12], [13].

In recent years, capacitated facility location has received a great deal of attention. Two main variants of the problem are soft-capacitated facility location and hard capacitated facility location. In hard-capacitated facility location problem, each facility is either opened at some location or not, whereas in soft-capacitated facility location problem, one may specify any integer number of facilities to be opened at that location. Bateni [1] proposes method to approximate NP-hard unsplittable hard-capacitated facility location problem in CDN. It provides solution with total connection cost at most \((1 + \epsilon)\) times the optimum with violation of capacities by at most \((1 + \epsilon)\) factor, for some \((1 + \epsilon) > 0\).

4 Model of Multi-index facility location problem in CDN

Our model for multi-index facility location problem in CDN encompasses following parameters, while finding optimal placement of input request amongst CDN servers:

- Distance between client and server
- Request size
- CPU utilization

This enables in minimizing the total assignment cost by modeling content distribution networks with various parameters.

Let \(A\) denotes set of servers. \(B\) denotes set of clients. \(N_j\) be the total network bandwidth of server \(j\). And, \(C_j\) denotes the total CPU capacity of server \(j\). Our multi-index unsplittable hard-capacitated facility location problem can be modeled as follows:

\[
\min_{\pi} \sum_{j \in A} f_j + \sum_{i \in B} n_i c_i d_{i, \pi(i)}
\]

- \(n_i\) Network bandwidth consumed by requester \(i\) (edge router).
- \(c_i\) denotes CPU utilized by requester \(i\).
- \(d_{i, \pi(i)}\) denotes distance between client \(i\) and server \(\pi(i)\).
- \(f_j\) denotes onetime cost of initializing facility \(j\).

Following are the constraints for above objective function

\[
\sum_{\pi(i) = j} n_i \leq N_j
\]

\[
\sum_{\pi(i) = j} c_i \leq C_j
\]
Theorem 4.1 Multi-index facility location problem in CDN is NP-Complete.

Step I) Show that Multi-index facility location problem $\in$ NP Multi-index facility location problem is decision problem as follows.

Input: Given weighted graph G. Facilities A[1,n] and clients B[1,m] are located on vertices of G. Each client i has demand with network bandwidth requirement $n_i$ and CPU requirement $C_i$. Each client needs to be assigned to facility j. Client needs to pay connection cost $n_i * c_i *$ distance between client i and server j. Each facility also has CPU capacity $C_j$ and Network bandwidth $N_j$. And, number $M$ which is cost of assignment for set of client request to servers.

Question: Does the given assignment have total cost equal to M?

Let T = "Non deterministic turing machine. On input A[1,n], B[1,m] M guess assignment function $\pi$ and verify whether given assignment cost M. If verification succeed then halt with Yes otherwise halt with no."

Given an assignment of client requests to servers, one can verify whether total cost of assignment equal to M in polynomial time. Therefore, Multi-index facility location problem $\in$ NP

Step II) Show that Multi-index facility location problem $\in$ NP-Hard

Let’s assume that there exists polynomial time algorithm for Multi-index facility location problem. For solving UHCMFL [1], we can use above algorithm as subroutine with following reduction:

CPU capacities of all servers are unlimited. By relaxing CPU constraints, decision of assignment from client demands to servers is only constrained by network bandwidth. Hence UHMCFL can be solved in polynomial time using Multi-index facility location problem.

As UMHCFL is NP-Hard, which proves that Multi-index facility location problem is also NP-Hard.

From result of step I and II, Multi-index facility location problem is NP-Complete.

Theorem 4.2 Multi-index facility location problem in CDN does not have any approximation algorithm, unless P=NP.

Let’s assume that all servers have connection cost and CPU cost 0, but facility costs are set to one. Uniform capacity case of Multi-index facility location problem is nothing but bin-packing problem. Bin-packing problem is NP-hard and it must require $(1 + \epsilon)$ relaxation of constraints. As Bin-packing problem does not have any approximation algorithm with violating capacity constraints. This means $(1 + \epsilon)$ relaxation of capacities is necessary for Multi-index facility location problem. Therefore, Multi-index facility location problem in CDN does not have any approximation algorithm, unless P=NP. Hence, approximation algorithm for multi-index facility location problem violates capacity constraints by $(1 + \epsilon)$ factor.

5 Our Contribution

In order to solve multi-index facility location problem, we use tree metrics. First step in our algorithm is to divide the clients into different groups according to their CPU and network bandwidth demands. In order to simplify the problem, we treat different demand ranges differently. We made assumption that all servers will have same network and CPU capacities. The grouping by size gets more complicated in the case of non-uniform capacities. We used a same dynamic programming approach of UMHCFL [1]. Each node of
the tree stores the clients not yet served in the subtree and the servers having some unused capacity.

Three types of costs are involved in this problem: the setup costs of facilities which is paid by them, the network connection costs, and CPU utilization cost. The later costs are shared between the facilities and clients in our algorithm. In resulting tree, path from root node to a client and facility has two portions. Each pays for its own portion of the path. For picking client and facilities for assignment, dynamic programming approach will account these costs and makes decision based on it. Using dynamic programming approach, optimal decisions are selected which are used to build a tree. By constructing tree in this fashion, we will know how much to pay for the costs in each path of tree. We only consider the case of uniform capacities.

Let us assume that none of the demands is too small. In particular, if the capacities of facilities are N and C then no client has demand smaller than $\epsilon \ast N$, and $\epsilon \ast C$. By making this assumption it is possible to group the demands into different sets. Our approach works on rooted trees.

To build a tree from given set of demands and server, we used Betani’s [1] method. From a subtree T with root v, let e be the immediate edge joining v to an outside vertex. Path that connects client in T to outside facility has to pass through edge e. Therefore, dynamic programming approach makes tradeoff while making decision about assignment from demand to facility. Based on tree structure, connection path from a client to a facility has two portions: one that moves up the tree towards the root and the other in which it moves downwards to reach the facility. Our approach charges the client for the first portion of the path and let the facility pay for the second portion. That means, servers and clients move to their least common ancestor and each pays the cost of getting there.

Betani’s [1] implementation does not consider CPU cost while making assignment decisions. Therefore, we have modified state structure of tree node as follows. Each node of tree is specified by (u, F-N, F-C, N, C, $F_{exp}$, $F_{imp}$), where u is the root of the subtree associated with this partial solution; F-N is a vector, that keeps the count of the available facilities with network bandwidth data; F-C stores available CPU bandwidth data of servers; N is a vector, that keeps the count of the outsourced large clients of different network bandwidth demands; C is a vector, that keeps the count of the outsourced large clients of different CPU bandwidth demands; $F_{exp}$ is the index of a facility being exported from this subtree. $F_{imp}$ is the index of a facility which is imported.

There are certain conditions which are necessary for making decision about merging of two states. And, merging two states involves assignment of demands to facility. A state $X = (u, F-N, F-C, N, C, x, y)$ is consistent with $X_1 = (u_1, F-N_1, F-C_1, N_1, C_1, x_1, y_1)$ and $X_2 = (u_2, F-N_2, F-C_2, N_2, C_2, x_2, y_2)$ if and only if all condition are satisfied.

1. $F-N_1 + F-N_2 - F-N = N_1 + N_2 - N \geq 0$;
2. $F-C_1 + F-C_2 - F-C = C_1 + C_2 - C \geq 0$;
3. $y_1 = x_2$ or $y_1 = y$ or $y_1 = NONE$;
4. $y_2 = x_1$ or $y_2 = y$ or $y_2 = NONE$; and
5. $x = x_1$ or $x = x_2$ or $x = NONE$.

The first two conditions ensure that the demand ignored at node u is actually matched to facilities of the same size. The next two make sure that what a subtree asks for, is either provided or is put on the request list for the next node. The last condition prevents us from exporting a facility we do not have. If all the above condition holds, then we can merge two states X1 and X2 to from X.

**Algorithm**

1. Change the instance into a rooted binary tree, with facilities and clients at leaves.
2. Initialize DP for leaves:
   - If there is a client with network bandwidth demand $N$ and CPU demand $C$ at a leaf $v$,
     - for a demand, let for any facility $x$, Dynamic programming value associated with $(v, 0, 0, 0, 0, \text{NONE}, x)$ be $N \times C \times \text{distance between } x \text{ and } v$
     - all other values corresponding to this leaf are set to infinity
   - If facility $i$ is located at this leaf,
     - let $(v,0,0, 0, 0,\text{NONE},\text{NONE})$ gets value 0;
     - put value infinity for any other DP cell corresponding to this subtree

3. Update the Dynamic programming memoization table values bottom-up. If the node $u$ has two children $u_1$ and $u_2$, a cell $X = (u, F-N, F-C, N, C, x, y)$ is updated (if this gives a better value) from $X_1 = (u_1, F-N_1, F-C_1, N_1, C_1, x_1, y_1)$ and $X_2 = (u_2, F-N_2, F-C_2, N_2, C_2, x_2, y_2)$ to $\text{cost}(X_1) + \text{cost}(X_2)$, if $C$ is consistent with $X_1$ and $X_2$.

4. Build the answer recursively from $\text{DP}(r,0,0, 0, 0,\text{NONE},\text{NONE})$, with $r$ being the root of the tree

**Time Complexity of our approach:**

Dynamic programming approximation approach to solve UMHCFL is $O(n^3/\epsilon)$. Our dynamic programming approximation approach also takes care of CPU constraints. Therefore, $n$ additional constrains need to be considered at each level of tree metrics. Hence, time complexity of our algorithm is $O(n^4/\epsilon)$.

6 **Conclusion**

We presented $(1+\epsilon)$ approximation algorithm for multi-index facility location problem in CDN by violating capacity constraints by $(1+\epsilon)$ factor, $\epsilon > 0$. With our proposed algorithm, CPU computing capacity of server is taken into consideration while making assignment. This enables real time modeling of CDN.

**Acknowledgement**

We would like to thank Dr. Alan Sherman for his useful suggestions. We would also like to thank our referees Mr. Tejas Lagvankar and Mr. Tushar Mahule for reading manuscript and pointing our several mistakes.

**References**


