Int. J. Communications, Network and System Sciences, 2010, 3, 850-854
doi:10.4236/ijcns.2010.311115 Published Online November 2010 (http://www.SciRP.org/journal/ijcns)

Service Networks Topological Design

Boris S. Verkhovsky
Computer Science Department, New Jersey Institute of Technology, Newark, USA
E-mail: *verb@njit.edu*
Received June 18, 2010; *revised August* 12, 2010; *accepted September* 23, 2010

Abstract

Topological design of service networks is studied in the paper. A quantitative model and algorithm minimizing cost of processing and delivery are described. An algorithm solving combinatorial problem of optimal design based on binary partitioning, a parametric search and dynamic programming optimization of a binary tree are described and demonstrated in numeric examples.

Keywords: Delivery/Processing Cost, Binary Partitioning, Dynamic Programming, First Responders, Average Complexity, Service Provider, Water Desalination

1. Introduction and Problem Definition

Modern satellite communication networks with their terrestrial users interconnected via terrestrial links with earth stations are an example of a service providing network (SPN).

Modern telecommunications is a highly competitive business. To reduce service fees and to make it economically attractive to potential customers, a communication company must decide how many earth stations of each type it needs, where to allocate them, and how the customers must be interconnected with the earth stations. A correct decision can save hundreds of millions of dollars annually and hence can attract more users.

In more general terms we consider a set of users that require a service [1]. The type of service can be either water desalination and/or purification and delivery to customers. Or it can be a regional gas supply. Or emergency services (ERs, fire departments, or police station or a set of first responders). Or a set of special postal service providers (mail deliverers, UPS offices, etc.).

From a computational point of view, the problem is *NP*-complete, *i.e.*, it requires brute-force algorithms or heuristics with exponential time-complexity. These approaches must find an optimal way of clustering all users, [2-5]. This paper describes a set of algorithms that solve the problem of clustering-and-location of all service providers with a polynomial time complexity. Preliminary results are provided in [6]. For more insights on problems and algorithms related with network design see [7-12].

2. Problem Statement

1) Let us consider locations of n users with coordinates $P_i = (a_i, b_i)$, $i = 1, \cdots, n$. Each user is characterized by a "volume" of required service w_i ("weight" of i-th user);

2) Let $C_k = (u_k, v_k)$ be a location of k-th service center (server, for short), $k = 1, 2, \cdots, m$;

3) Let S_k be a set of all users connected with the k-th server C_k;

4) Let $f(w_i, P_i, C_k)$ be a cost function of the link connecting the i-th user P_i and k-th server C_k;
here for all $i = 1, 2, \cdots, n$ P_i are the inputs and for all $k = 1, 2, \cdots, m$ C_k are decision variables.

Then a minimal total cost of all links and all servers equals

$$\min_{S_1,\cdots,S_m} \min_{C_1,\cdots,C_m} \left[\sum_{k=1}^{m} \sum_{i\in S_k} f_i(P_i, C_k) + q_k\left(\sum_{i\in S_k} w_i\right) \right], \tag{1}$$

where $q_k(\sum_{i\in S_k} w_i)$ is the cost of k-th server as a nonlinear function of all service flows. Thus the problem (1) requires a comprehensive analysis in order to find optimal clusters (subsets) $S_1, \cdots, S_m$.

Models and methods of clustering in general has been studied and described in [5]. Surveys on models and algorithms related to clustering are provided in [5,9].

3. Special Cases

Case 1: The number m of servers is known and the cost function of every server is independent of required processing volume. If locations of all servers are specified, then it is easy to find the clusters S_k.

Indeed, in

$$S_k := \left\{ i : f\left(w_i, P_i, C_k\right) := \min_{1 \le j \le m} f\left(w_i, P_i, C_j\right) \right\} \quad (2)$$

i.e., every user is connected with the closest or least expensive delivery link.

Case 2: If for $k = 1, \cdots, m$ S_k are known, then the optimal allocation of every server can be determined independently:

$$\min_{C_k} \sum_{i \in S_k} f\left(w_i, P_i, C_k\right) \quad (3)$$

for $k = 1, \cdots, m$.

Case 3: If $f(w_i, P_i, C_k) = w_i \times dist(P_i, C_k)$, then the problem (3) is known as a Weber problem. This problem has been investigated by many authors over the last thirty years. For references see [10,13-16].

Case4: $q_k(\sum_{i \in S_k} w_i)$ is a linear or convex function, *i.e.*,

$$q_k\left(w_1 + w_2\right) \ge q_k\left(w_1\right) + q_k\left(w_2\right)$$

Then

$$q_k\left(\sum_{i \in S_k} w_i\right) \ge q_k\left(\sum_{i \in S_{2k}} w_i\right) + q_k\left(\sum_{i \in S_{2k+1}} w_i\right)$$

which means that the more clusters the better.

The difficulties arise if the clusters S_k are not known, the cost of every server is neither small nor flow-independent, and the number m of servers and their optimal locations (u_k, v_k) for all k are not known.

4. Division onto Two Clusters

It is important to stress that there is a substantial difference between the two cases: $m = 1$ and $m = 2$. In the latter case the problem can be solved by repetitive application of an algorithm for the Weber problem. This must be done for all possible pairs of clusters S_1 and S_2. There are $2^{n-1} - 1$ different ways to partite n points onto two subsets S_1 and S_2 and, for each clustering, two Weber problems must be solved. Thus, the total time complexity of this brute-force approach {even for $m = 2$} is $O(2^n)$, [3].

5. Binary Parametric Partitioning

In this section we provide a procedure that divides a network N_1 with one server S_1 onto two sub-networks N_2 and N_3 with two servers.

Step A1: {find optimal location of "center of gravity" C_0 for all n users, [1]}; consider $m = 1$ and solve the problem

$$\min_C \sum_{i=1}^{n} f\left(w_i, P_i, C\right). \quad (4)$$

Step A2: consider a straight line L and rotate it around the center of gravity C_0; {for every angle x of rotation the line L divides all points P_i onto two clusters $S_1(x)$ and $S_2(x)$};

Step A3: for every point (user) consider polar coordinates (ρ_i, d_i) using C_0 as the origin of the coordinate system; {here ρ_i is an angular coordinate of P_i};

Step A4: **for** $i = 1$ **to** n **do**

$$x_i := \rho_i \bmod \pi; \quad (5)$$

sort all x_i in ascending order;

Step A5: **if** $x_i = \rho_i$ **then** $ch_i = 1$ **else** $ch_i = 0$;

Step A6: **if** ($x_i \le x$ **and** $ch_i = 1$) **or** ($x_i > x$ **and** $ch_i = 0$) **then** $P_i \in S_1(x)$ **else** $P_i \in S_2(x)$;

Step A7: **for** $k = 1,2$ **and** $i \in S_k(x)$ compute

$$g_k\left(S_k(x)\right) := \min_{C_k} \sum_{i \in S_k} f\left(w_i, P_i, C_k\right); \quad (6)$$

Step A8: {compute the cost of two servers and all links}:

$$h(x) := q_1\left(\sum_{i \in S_1} w_i\right) + q_2\left(\sum_{i \in S_2} w_i\right) + g_1\left(S_1(x)\right) + g_2\left(S_2(x)\right) \quad (7)$$

Remark 1**:** The line L divides all n points onto two clusters, $S_1(x)$ and $S_2(x)$, by at most n different ways as the angle x changes from 0 to π;

Step A9: Rotate the separating line L and find an angle that minimizes the function

$$h(x)\ \{see(7)\}:\ h(r) := \min_{0 \le x \le \pi} h(x) \quad (8)$$

Robustness of partitioning: In Step A9, the angle of rotation x of line L is the *parameter*. Several thousands computer experiments demonstrate that L divides all points/users onto two subsets S_1 and S_2 in such a way that for $k = 1, 2$ the following property almost always holds: if $i \in S_k(r)$,

then $$f(w_i, P_i, C_k) \le f(w_i, P_i, C_{3-k}) \quad (9)$$

However, if n is large, then with small probabilities there are one or two points (called "fugitives"), for which (9) does not hold;

Step A10: **if for** $i \in S_1(r)$

$$f(w_i, P_i, C_1) > f(w_i, P_i, C_2),$$

then reassign $i \in S_2(r)$;

if for $i \in S_2(r)$

$$f(w_i, P_i, C_2) > f(w_i, P_i, C_1),$$

then reassign $i \in S_1(r)$;

Remark 3: $i \in S_1(r)$, then

$$\text{Prob}\,[f(w_i, P_i, C_1) > f(w_i, P_i, C_2)] = \beta,$$

where

$$\beta = 0, \text{if } n \le 400 \text{ and } \beta \le 1/200 \quad \text{if } n \ge 500$$

Therefore the separating like with very high probability cluster the points onto two sets, for which (9) holds.

Step A11: using (7) and (8) update optimal locations of C_1 and C_2 for new values of $S_1(r)$ and $S_2(r)$; {we define $S_1(r)$ and $S_2(r)$ as the optimal binary partitioning}.

Remark 4: The computer experiments for $25 \le n \le 600$ indicate remarkable robustness of the Binary Partitioning Algorithm (BPA): the Step 10 and Step 11 do not create instability of the BPA, [6].

6. Search for "Center of Gravity"

Step B1: assign *flag*: = 0;

$$u := \sum_{i \in N_1} w_i a_i / \sum_{i \in N_1} w_i;$$
$$v := \sum_{i \in N_1} w_i b_i / \sum_{i \in N_1} w_i$$

Step B2: for all $i \in N_1$

$$R_i := \sqrt{(u - a_i)^2 + (v - b_i)^2};$$

Step B3: *old(u,v)*: = (*u*,*v*);

$$u := \left(\sum_i w_i x_i / R_i\right) / \left(\sum_i w_i / R_i\right);$$
$$v := \left(\sum_i w_i y_i / R_i\right) / \left(\sum_i w_i / R_i\right)$$

Step B4: **while**

$$dist\left[old(u,v),(u,v)\right] > \varepsilon,$$

repeat Steps B2 and B3;

Step B5 {search of a stationary point *SP*; ε is a specified accuracy of location of the center of gravity}: Assign *SP*: = (*u*,*v*);

Step B6: **if** for all $j \in N_1$ { $dist(SP, P_j) > \varepsilon$ **and** *flag* = 0}, **then** *SP* is the "center of gravity";

if for all $j \in N_1$ { $dist(SP, P_j) > \varepsilon$ **and** $flag = -1$ }, **then** $N_1 := N_1 + \{pnt\}$; *flag*: = 0; **goto** Step B2;

Step B7: **if** $dist(SP, P_j) \le \varepsilon$ **then** $flag = -1$; *pnt*: = *k*; $N_1 := N_1 - \{pnt\}$.

7. Minimax Search for Minimum of *h*(*x*)

Let *h* be a function computable on a set *S* of *N* discrete points $x_1, \cdots, x_N$. It is obvious that *N* evaluations of *h* at points $x_1, \cdots, x_N$ are enough to solve any problem by total enumeration. However, since *h* is a periodic function with known period *P, i.e.*, $h(x_i + kP) = h(x_i)$ holds for every integer *k*, and for every *i* = 1, $\cdots$, *N*, an optimal algorithm with time complexity of order $\Theta(\log N)$.is developed and published in [16,17].

8. Binary Partitioning & Associated Binary Tree

Let $H_k := t_k + s_k$ where H_k is a combined cost of the network N_k. Let's consider an algorithm that divides the network N_1 into two sub-networks N_2 and N_3 with corresponding service delivery costs t_2 and t_3 and corresponding costs of servers s_2 and s_3. We assume that the algorithm divides N_1 into two subnets in such a way that $t_2 + t_3 + s_2 + s_3$ is minimal.

For further consideration we represent the binary partitioning as a binary tree where the root of the tree represents a cluster (set of all users) S_1 and associated with it the network N_1. In general, a *k*-th node of the binary tree represents a cluster S_k and associated with it a network N_k. Two children of the *k*-th node represent two sub-networks N_{2k} and N_{2k+1} of the network N_k (as result of the binary partitioning).

From the above definitions and from the essence of the problem it is clear that for all *k* the following inequalities hold

$$s_k \ge s_{2k}, s_k \ge s_{2k+1} \text{ and } t_k \ge t_{2k} + t_{2k+1}. \quad (10)$$

The latter inequality holds because each sub-network N_{2k} and N_{2k+1} has a smaller number of users than N_k.

9. Non-Monotone Nature of Total Cost

If $H_k > H_{2k} + H_{2k+1}$, then it is obvious that a partitioning into two clusters (sub-networks) is cost-wise beneficial.

However, $H_k < H_{2k} + H_{2k+1}$ does ***not*** imply that any further partitioning is not cost-wise beneficial. To illustrate that let's consider a network N_k and its *six* subnetworks N_{2k}, N_{2k+1}, N_{4k}, N_{4k+1}, N_{4k+2}, N_{4k+3}.

Remark 6: To demonstrate various cases we consider two scenarios of inputs in the following table.

Case H_1 = 91 (see **Table 1**): since $H_1 > H_2 + H_3$, then the binary partitioning of N_1 into two sub-networks is gainful;

Case H_1 = 86 (see **Table 2**) illustrates that a local analysis of the total costs does not provide a correct insight.

Table 1. t_1 = 67 and for t_5 = 11.

Sub-networks N_i	N_1	N_2	N_3	N_4	N_5	N_6	N_7
Delivery-link cost t_i	**67**	25	29	12	**11**	10	7
Server cost s_k	24	19	17	10	14	12	9
Total cost H_k	91	44	46	22	25	22	16

Table 2. t_1 = 62 and for t_5 = 7.

Sub-networks N_i	N_1	N_2	N_3	N_4	N_5	N_6	N_7
Delivery-link cost t_i	**62**	25	29	12	**7**	10	7
Server cost s_k	24	19	17	10	14	12	9
Total cost H_k	86	44	46	22	21	22	16

In this case $H_1 < H_2 + H_3$, which only implies that there is no reason to divide the network N_1 into two sub-networks N_2 and N_3. However, further analysis demonstrates that

$$H_1 > H_4 + H_5 + H_6 + H_7 \quad \text{if } H_5 = 21;$$

$$\{\text{indeed, } 86 > 22 + 21 + 22 + 16 = 81\};$$

and $$H_1 > H_2 + H_6 + H_7 \quad \text{if } H_5 = 25;$$

$$\{\text{indeed, } 91 > 44 + 22 + 16 = 82\};$$

These examples illustrate that for a proper partitioning a global rather than a local analysis is required.

Definition 1: We say that a network N_k is *indivisible* if there is no cost-wise advantage to divide it onto any number of sub-networks.

In addition, some sub-networks may not be further divisible if they do not satisfy at least one of the following threshold conditions:

a) Their request for service is lower than a specified threshold;

b) The number of users in the cluster is smaller than a specified threshold.

Definition 2: We say that an optimal configuration of a service network is determined if all indivisible sub-networks of the initial network N_1 are known.

10. Dynamic Programming Algorithm

{The algorithm assigns final labels to all nodes of the associated binary tree; then determines all balanced nodes and then all prime nodes};

Bottom-up mode: a) {assign to k-node a label};

$$L_k := H_k, \quad k = 1, 2, \cdots, m; \qquad (11)$$

b) **if** i-th node is a leaf **then** its final label

$$F_k := L_k; \qquad (12)$$

c) **if** both children of k-th node have final labels **then**

$$F_k := \min\left(L_k, F_{2k} + F_{2k+1}\right); \qquad (13)$$

d) **if** the final labels F_k are computed for all nodes **then goto** the next mode;

Top-down mode: Starting from k=1 we find a node j, for which holds

$$F_j = L_j. \qquad (14)$$

It can be shown that it is not cost-wise advantageous to consider the descendants of this node, *i.e.*, this node is indivisible.

Definition 3: We say that j-th node is *balanced* if $F_j = L_j$ holds.

Definition 4: A node that does not have a balanced ancestor is a *prime* node.

Proposition: The set P^{opt} of all prime nodes represents the *optimal* partitioning.

Proof: Let's prove by reduction that (13) is always computable, *i.e.*, when we need to compute a final label for the k-th node, the final nodes of its both children are already computed.

Indeed, consider a sub-tree T1 with five nodes {A, B, C, D, E} and sub-tree T2 with seven nodes {A, B, C, D, E, G, H}, where in both sub-trees node A is a root.

Sub-tree T1: Let B and E be the children of A; and C and D be the children of B. In addition, let the nodes C, D and E be leaves.

Then by definition (12),

$$F_C := L_C;\ F_D := L_D \text{ and } F_E := L_E. \qquad (15)$$

Therefore, $F_B := \min(L_B,\ F_C + F_D)$

and $$F_A := \min(L_A,\ F_B + F_E). \qquad (16)$$

Thus, if T1 is a sub-tree of a larger tree, we can compute five final labels, *i.e.*, we reduce the number of unknown final labels by five.

Sub-tree T2: Let B and E be the children of A; C and D be the children of B; G and H be children of E. In T2 nodes C, D, G and H are the leaves.

It is easy to see that this case can be reduced to the previous case by computing the final label of E. Indeed, by definition (12),

$$F_G := L_G \text{ and } F_H := L_H \qquad (17)$$

Therefore, $$F_E := \min(L_E,\ F_G + F_H) \qquad (18)$$

Thus, after the final label of node E is known, we can consider it as a virtual leave. In other words, we reduced the sub-tree T2 to the sub-tree T1.

Hence, if T2 is a sub-tree of a larger tree, we can compute the all final labels of its nodes, *i.e.*, we reduce the number of unknown final labels by seven.

It is clear that in order to compute a final label for every node we need one addition and one comparison. Hence the overall complexity to compute the final labels has order $O(M)$, where M is the number of nodes in the initial tree. In the worst case every leave has at least 2 users. Thus the total number of nodes M on the tree does not exceed $n - 1$, where n is the number of the users. Therefore the complexity equals $O(n)$.

11. Optimal Algorithm for Large *n*

It is easy to see that the parametric partitioning requires in general case exactly n rotations of the separating line L. As a result, the time-complexity $f(n)$ to divide n users onto *two* clusters is equal $f(n) = an^2 + O_1(n)$ for large n. However, from computer experiments for large number of users n $h(x)$ has one maximum and one minimum when L is rotated on 180 degrees. In this case the search algorithm requires O(log$n$) rotations of the separating line L. As a result, $f(n) = bn\log n + O_2(n)$ for large n and overall worst-case complexity is of order $O\left(n^2 \log n\right)$.

As it shown in [18] an average complexity of the problem can be further improved. The developed approach demonstrates that the average complexity of the overall binary partitioning is of order $O\left(n \log^2 n\right)$.

12. Conclusions

The ideas and algorithms described in this paper have numerous applications. In the set of algorithms provided above the Binary Partitioning algorithm (BPA) is a core sub-algorithm for solutions of many problems dealing with configuration/topological design of networks and other clustering problems. As the core sub-algorithm, the BPA has to be repeatedly executed many times; therefore it is essential to make the BPA computationally as efficient as possible. There are several computational blocks within the BPA:

- To determine an optimal location of the "centre of gravity";
- To detect of a minimum of computable function using the smallest number of its probes;
- To estimate an average complexity of a divide-and-conquer algorithm.

We developed an appropriate quantitative analysis to handle efficiency of the BPA.

Finally, it is shown how to apply a dynamic programming to tackle combinatorial complexity of reconstructing the best network after numerous binary partitions.

13. Acknowledgement

I express my appreciation to D. Nowak for his assistance.

14. References

[1] M. Fiddler and V. Sander, "A Parameter Based Admission Control for Differentiated Services Networks," *Computer Networks*, Vol. 44, No. 4, March 2004, pp. 463-479.

[2] A. Chaves and L. Lorena, "Clustering Search Algorithm for the Capacitated Centered Clustering Problem," *Computers and Operations Research*, Vol. 37, No. 3, March 2010, pp. 552-558.

[3] G. Diehr, "Evaluation of a Branch-and-Bound Algorithm for Clustering," *SIAM Journal on Scientific and Statistical Computing*, Vol. 6, No. 2, April 1985, pp. 268-284.

[4] J. Heath, M. Fu and W. Jank, "New Global Optimization Algorithms for Model-Based Clustering," *Computational Statistics and Data Analysis*, Vol. 53, No. 12, October 2009, pp. 3999-4017.

[5] A. Kusiak, A. Vannelli and K. R. Kumar, "Clustering Analysis: Models and Algorithms," *Control and Cybernetics*, Vol. 15, No. 2, 1986, pp. 139-154.

[6] B. Verkhovsky, "Satellite Communication Networks: Configuration Design of Terrestrial Subnetworks", *Journal of Telecommunications Management*, 2010.

[7] M. Gen and R. Cheng, "Evolutionary Network Design: Hybrid Genetic Algorithms Approach," *International Journal of Computational Intelligence and Applications*, Vol.3, No. 4, December 2003, pp. 357- 380.

[8] H. L. Chen and R. Tim, "Network Design with Weighted Players," *Theory of Computing Systems*, Vol. 45, No. 2, June 2009, pp. 302-324.

[9] D. S. Johnson, J. K. Lenstra and A. H. G. Rinooy Kan, "The Complexity of the Network Design Problem," *Networks*, Vol. 8, No. 4, 1978, pp. 279-285.

[10] P. McGregor and D. Shen, "Network Design: An Algorithm for the Access Facility Location Problem," *IEEE Transactions on Communications*, Vol. COM-25, No. 1, January 1977, pp. 61-73.

[11] J. Smith, F. Cruz and T. V. Woensel, "Topological Network Design of General, Finite, Multi-Server Queuing Networks," *European Journal of Operational Research*, Vol. 201, No. 2, March 2010, pp. 427-441.

[12] B. Verkhovsky, "Constrained Shortest Path Algorithm for Network Design," *International Journal of General Systems*, Vol. 23, No. 2, 1996, pp. 183-195.

[13] M. Bischoff and K. Dächert, "Allocation Search Methods for a Generalized Class of Location–Allocation Problems," *European Journal of Operational Research*, Vol. 192, No. 3, February 2009, pp. 793-807.

[14] R. Bellman, "An Application of Dynamic Programming to Location–Allocation Problems," *SIAM Review*, Vol. 7, No. 1, January 1965, pp. 126-128.

[15] J. Zhou and B. Liu, "Modeling Capacitated Location–Allocation Problem with Fuzzy Demands," *Computers and Industrial Engineering*, Vol. 53, No. 3, October 2007, pp. 454-468.

[16] B. Veroy (Verkhovsky), "An Optimal Algorithm for Search of Extreme of a Bimodal Function," *Journal of Complexity*, Vol. 2, 1986, pp. 323-332.

[17] B. Veroy, "Optimal Search Algorithms for Extrema of a Discrete Periodic Bimodal Function," *Journal of Complexity*, Vol. 5, No. 2, June 1989, pp. 238-250.

[18] B. Veroy, "Average Complexity of a Divide-and-Conquer Algorithms," *Information Processing Letters*, Vol. 29, No. 6, December 1988, pp. 319-326.