IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-5, Issue-11 E-ISSN: 2347-2693

Facility Location: A Theoretical Approach for Flood Relief

Vairaprakash Gurusamy[1*], K. Nandhini[2]

[1*]Department of Computer Applications, School of IT, Madurai Kamaraj University, Madurai, India
[2]Technical Support Engineer, Concentrix India Pvt Ltd, Chennai, India

[*]*Corresponding Author: vairaprakashmca@gmail.com*

Available online at: www.ijcseonline.org

Received: 20/Oct/2017, Revised: 29/Oct/2017, Accepted: 22/Nov/2017, Published: 30/Nov/2017

***Abstract* ---**We present a theoretical approach to come up with an effective mechanism for flood relief in terms of facility location problem with certain constraints. Facility location problem is a well studied economical decision problem to locate limited facility on demand points to cover maximum demand. Let consider a facility network under link failure. The problem is to locate emergency response facilities on a network with links that are subject to a failure model, called vulnerability − based dependency. We address the MAX-EXP-COVER-R problem in the facility network subjects to VB-dependency failure model. The MAX-EXP-COVER-R problem is known to be NP-hard. Let the distance factor R is relaxed from MAX-EXP-COVER-R problem, then it becomes MAX-EXP-COVER problem. The MAX-EXP-COVER problem can be solved in linear time using greedy algorithm. The MAX-EXP-COVER problem is further reduced into an instance of full binary tree, and then run MAX-WT-K-LEAF-SUBTREE problem on the tree will outputs the solution for MAX-EXP-COVER problem. Let the distance factor R = 1, then MAX-EXP-COVER-R problem is equivalent to budgeted dominating set problem with budget k and the induced subgraph of the solution set is need not to be connected.

***Keywords*---**Facility location, Disaster management, Approximation algorithm, NP-Hardness, Dominating sets, Link failure model, Graph theory, Hardness.

I. INTRODUCTION

The facility location problems capture the common features of the optimization problems. The goal of facility location is to serve a set of demand points, typically called as clients, by an opening a set of access points, typically called as facilities, in a cost effective and efficient manner. The distances between the clients and facilities are assumed to satisfy metric properties. There is a location dependent cost for opening a facility at each location. Typically, a solution to a facility location problem is specified by a set of facilities to be opened and an assignment of the clients to the open facilities [1].

The sum of costs of opening the facilities is called the facility cost, and the sum of distances of each client to the facility it is assigned to is called as the service cost of the solution. Different variants of the facility location problem are obtained by combining these costs in different ways. A richer set of problems emerges by considering formulations in which each facility can serve at most a specified number of clients or by making the cost of a facility depend on the number of clients it serves after the assignment. The distance metric is assumed to be a real number represents the maximum distance that can cover by a open facility and a demand point, which obtain service from the corresponding open facility.

Given an instance of (G = (V, E), k, R), where V is the vertex set or demand point and E is connectivity between demand points. Each edge $e \in E$ is associated with a distance metric $l_e \in R$. Each vertex $v_i \in V$ is demand point with a non-negative demand $w_i \in R$. The distance factor R is given as maximum facility coverage limit for any open facility node. We need to setup k open facilities on V such that the demand satisfied is maximized. The objective function is given as,

$$\underset{F \subseteq V, |F| \leq k}{\text{Max}} \sum_{v_i \in v} Pri$$

where Pr_i is the profit function associated with each vertex. The profit function Pr_i for each vertex $v_i \in V$ is given as,

$$Pr_i = \begin{cases} w_i, & \text{if } v_i \in F, \\ w_i, & \text{if } \exists u \in F \mid d(u, v_i) \le R, \\ 0, & \text{otherwise.} \end{cases}$$

$d(u, v)$ be the shortest path distance between the vertices.

Let we introduce a variant of facility location problem, called facility networks with unreliable links. A network is given with finite set of demand points, unreliable links (reliability of the link is given as probability value of the edge) and an integer k denoting the maximum number of potential facilities that can be opened at demand points. Our goal is to locate k facilities such that expected demand satisfied is maximized. The facility location problem is an economical decision problem to open limited number of facilities on a demand node to satisfy the maximum number of demand nodes in the network. Suppose the amount facility that can serve by a potential demand node is limited, then it can cover or serve only limited number of demand nodes. Let infinite capacity be given to the potential facility nodes, then each potential facility node could serve to the demand nodes which are reachable [4]. In this work we studies a link failure model, called Vulnerability-Based dependency (VB-dependency) [4]. In case of VB-dependency, failure of any link l implies the failure of all links weaker than l. A potential disaster may damage the network and some of the unreliable links could fail. Facility location problem can be solved by variants of dominating sets in different scenario. The MAX-EXP-COVER-R problem is a maximization problem. Let the factor R is relaxed from MAX-EXP-COVER-R problem then it became MAX-EXP-COVER problem. If a network is subjected to VB- dependency, then MAX-EXP-COVER problem is solvable in linear time using either dynamic programming or greedy method [4].

II. FACILITY LOCATION ON A NETWORK WITH UNRELIABLE LINKS

In case of unreliable links, each link $e_j \in E$ is associated with a surviving probability $p_j \in [0, 1]$. Suppose a potential disaster event damages *G*, then the links are subject to random failure. After disaster every link is in binary state that either working or non-working. Links are subject to failure after disaster but not nodes. After disaster each node survives and an emergency response facility can be located at any node.

2.1 Vulnerability based dependency and realization

Given two links e_i and e_j with survival probability p_i and p_j, we say e_i and e_j have VB dependency, if $p_i < p_j$ then $\Pr[e_i \text{fails} \mid e_j \text{ fails}] = 1$ [5]. Failure of any link implies failure of all links weaker than that one. In real time scenario it is not the case always, but it is useful to reduce the 2^m unique realization of survival network into $m + 1$ realizations. Factorize the vulnerability of the components in a link, such that it finds all strongly connected components even if any edge failure in disaster. Sort the edges with respect to probabilities in decreasing order (assume probabilities are unique) indexed from $1 \cdots m$. 1^{st} index represents strongest and m^{th} index represents weakest edge. Under the *VB-dependency*, only $(m + 1)$-distinct surviving network realization possible with positive probability.

The realizations are represented as G^i, *f or* $i = \{0, \ldots, m\}$. Each network realization $G^{(i)}$ is associated with a probability P_i. For each realization *i*, $G^{(i)}$ can be represented by a *m*-bit binary vector. $G^{(0)} = \{00 \ldots 0\}$ is the realization where every edge failed after disaster. The probability of such realization depends on the strongest link. $G^{(i)}$ is the survival network realization, in which i^{th} indexed edge is the minimum probability edge survived. $G^{(i)}$ can be represented as a binary vector in form of $i\ 1^s$ followed by $m + 1 - i\ 0^s$, that is $P_0 = 1 - p_1$. $G^{(m)} = \{11 \cdots 1\}$ is the realization where all links survive after disaster. Probability of the realization is depends on weakest link, that is $P_m = p_m$. For any realization $G^{(k)}$ for $k = \{1, \ldots, m - 1\}$, probability of the realization is the gap between the probabilities of strongest failure link and weakest survival link, that is $P_k = p_k - p_{k+1} \quad \forall k \in \{1, \ldots, m-1\}$.

2.2 MAX-EXP-COVER-R problem

Let $G = (V, E)$ be the facility network, we need to open at most k potential facilities on V . Once R is fixed and a potential facility opened in vertex v, then v can cover demands of all the vertices connected to it with distance less than or equal to R. For a fixed open facility, we need to compute the maximum demand cover by total facility in each realization of network, where link failure subject to *VB*-dependency. The MAX-EXP-COVER-R problem is defined as to place at most k facilities on a network such that the expected total demand covered within distance R is maximized.

Let R be the maximum covering distance, and k be the budget for open facility. Choose $F \subseteq V$, such that $|F| \leq k$. Let I^q l_v^q be a binary indicator variable associated to each vertex on each realization such that it takes the value 1, if demand of the vertex v is covered by any facility node in F in the realization $G^{(q)}$, and 0, otherwise. Let Pq be the probability of the realization $G^{(q)}$, which is described in Section 2.1. MAX-EXP-COVER-R problem is given as follows.

$$\max_{F \subseteq V, |F| \leq k} \sum_{q=0}^{m} \sum_{v_i \in V} P_q w_i I^q$$

MAX-EXP-COVER-R problem is proved to be NP-Hard [2]. Maximum k-facility location is a well known NP-Hard problem is used to show that MAX-EXP-COVER-R is NP-Hard.

MAX-EXP-COVER problem be the modified version of MAX-EXP-COVER-R problem such that, distance limit R is relaxed. A potential open facility can cover all the vertices connected to it, irrespective of distance factor. In this case, I^q is 1, if v has a potential facility node connected to it in the same component. MAX-EXP-COVER problem is solved using MAX-WT-K-LEAF-SUBTREE problem. Given an instance of facility network with unreliable links can be reformulated into a rooted tree, called *component tree*. Then component tree is solved using MAX-WT-K-LEAF-SUBTREE by either dynamic programming or a greedy algorithm.

III. COMPONENT TREE CONSTRUCTION AND MAX-WT-K-LEAF-SUBTREE

Consider G is connected. Under *VB-Dependancy* of link Max-WT-K-Subtree, there are $G^{(k)}$, $k = 0,1,...,$ m unique realizations are possible. $G^{(m)}$ corresponds to G, and $G^{(0)}$ corresponds to completely disconnected graph (without edges). Sort the edges in descending order of the probability. When we remove an edge one by one from weakest link, we get the realization from $G^{(m-1)}$ to $G^{(0)}$. When an edge disconnected from G, the number of connected components in G is remain same or increment by 1. Suppose removal of an edge $e \in E$ makes G into disconnected, then e is a *disconnecter*. Suppose e_i be the first link disconnects G into two component then mark e_i as $e_{[1]}$. Continue the procedure till G is completely disconnected, that is $G^{(0)}$ obtained. Given m edges and n vertices, removal of $n-1$ edges make the graph to be disconnected. From the weakest probability edge to strongest one, the corresponding disconnecting edge is marked from $0,...,m-1$.

Component-Tree is a full binary tree, has two type of nodes (edge type and vertex type). Root node and intermediate nodes are edge type (disconnecter ($e_{[j]} = e_i$)) and leaf nodes are vertex type. Every disconnecter has two children represents two connected components. A component tree is a binary tree, because removal of an edge splits the graph into at most 2 components. Recursively these connected components are disconnected by consequent dis-connectors until the component become a single vertex.

Let say $e_{[1]}$ is the first disconnecter, then $Node_{e_{[1]}}$ is root node of Component-Tree. $Node_{e_{[1]}}$ has two children. Since the tree is full binary tree with n leaf nodes then, there are n-1 non leaf nodes exist. Associate two quantities with each node of the Component-Tree: a weight W and a probability P. At any leaf node corresponds to an original node v_i of G, the weight $W(i) = w_i$, and the probability $P(i) = P_0$, the probability of the realization where all vertices are in separate component (it can cover it self by placing a open facility). For non-leaf nodes, weight of the node $e_{[i]}$ (e_j) is the sum of weight of its left subtree and right subtree (sum of weights of all leaf nodes under it).

$$W(i) = \begin{cases} w_i & \text{if } i \text{ is a leaf node represents } v_i \\ W(L) + W(R) & \text{if } i \text{ is a non leaf node represents a disconnector } e_{[i]} = e_j \end{cases}$$

where L and R are the corresponding left and right subtree.

Probability $P(e_{[i]} = e_j)$ is set to sum of the probability of all the realization in which leaves of subtree L and R exactly the nodes of a single connected component. Let say the failure of e_j splits the component into two parts (L and R). But component is formed when $e_{[i-1]} = e_l$ fails, that is $G^{(l+1)}$ is the first realization that e_l fails for the first time. We know that $l > j$, because sorted in descending order ($p_l < p_j$). Hence $P(e_{[i]} = e_j)$ is the sum of probability of realizations from $G^{(j+1)}$ to $G^{(l+1)}$.

$$P(e_{[i]}) = \sum_{t=j+1}^{l+1} Pr(G^{(t)})$$

$$P(e_{[i]}) = \sum_{t=j+1}^{l+1} P_t - P_{t-1}$$

$$P(e_{[i]}) = P_j - P_{l+1}$$

Once weight and probability is calculated for each node, the we can compute expected weight (EW (i)), which is expressed as $EW(e_{[i]}) = W(e_{[i]})P(e_{[i]})$ for non-leaf nodes and $EW(i) = w_i(1 - P_1)$ for leaf nodes. Given a single rooted binary tree, where each node is associated with a nonnegative expected weight (EW), find the maximum cost induced rooted tree with k-Leaves.

Dynamic programming algorithm for MAX-WT-K-LEAF-SUBTREE problem is formulated as follows. Let define the function $MaxW$ (v, t) for each node $v \in T$, and for every positive integer $t \leq k$, such that $MaxW$ (v, t) denotes the maximum expected weight obtained by choosing exactly t leaves of the subtree rooted at v. For some $v \in V$, $t = 0$ means that there is no leaf chosen in the the subtree rooted at v.

Let $v \in T$ be a node. Right child and left child of v is represented as v_r and v_l respectively. The subtree rooted at v_l and v_r are represented as L and R respectively. If $t = 0$ for any $v \in T$, means that $MaxW$ (v, 0) = 0. If a subtree T_V rooted at v has lesser or equal number of leaves than t, then $MaxW$ (v, t) = $\sum_{u \in T_V} EW(u)$.

$$MaxW(v,t) = \begin{cases} \max_{0 \leq t^r \leq t} \{MaxW(v_l, t^r) + MaxW(v_r, t - t^r)\} + EW(v), & v \text{ is a non-leaf node and } t > 0 \\ EW(v), & v \text{ is a leaf ndoe } and t > 0 \\ 0, & \text{Otherwise} \end{cases}$$

Given a limit t for a vertex $v \in V$, at most t leaf vertex can be chosen from the subtree rooted at v. We know that T is a complete binary tree. Given a vertex v and an integer t, DP will find a $t^r \leq t$, such that t^r leaf nodes are chosen from L and $t - t^r$ leaf nodes are chosen from R. To find a better t^r, We need to iterate the value from 0 to t, such that the maximum weight is obtained. Recursive procedure is solved from leaf to root. The running time of the algorithm is $O(k.n^2)$.

MAX-WT-K-LEAF-SUBTREE can be solvable in linear time using *The greedy algorithm*. Given a problem instance of *(T, k)*, greedy algorithm compute a new tree with additional quantity (an integer) $C(v)$ associated with each node $v \in T$. The sum of weights of nodes from root to the node v is stored in $C(v)$. The cost $C(v)$ is given as,

$$C(v) = \begin{cases} WE(v), & v \text{ is root node} \\ C(v_p) + WE(v), & \text{Otherwise} \end{cases}$$

where v_p is the parent node to v. The cost for each node is computed in top-down approach by traversing the tree. Once $C(v)$ is computed for all the nodes then, choose the leaf that contains the maximum $C(v)$ as greedy choice. Let l_{OTP} be the leaf node

with maximum $C(l_{OTP})$. For every nodes v in the path from l_{OTP} to root, set $EW(v) = 0$. This updated tree is represented as T^r. The optimal substructure for this problem is given as $(T^r, k-1)$.

Correctness of the greedy algorithm is to show that greedy choice and optimal substructure is correct. Let $lOTP$ be the leaf node having maximum value $C(l_{OTP})$. We need to show that for any $k \geq 1$ in the problem instance of *(T, k)*, the l_{OTP} is contained in optimal solution (OPT). For the contradiction, consider l_{OTP} is not in OPT. Choose a leaf node l in OPT, such that the least common ancestor of l^* and $lOTP$ has minimum height. We know that $C(l^*) \leq C(l_{OTP})$ (l_{OTP} is the leaf having maximum $C(l_{OTP})$). Let v^r be the least common ancestor for l^* and l_{OTP}. Choose a leaf node l_1 in OPT, such that the least common ancestor of l^* and l_1 has minimum height, represent the least common ancestor as l^r. Suppose l^r is an ancestor of v^r, then consider the new solution by excluding l^* from OPT and add l_{OPT}. The weight of the new solution S^* containing l_{OPT} is

$$EW(T(S^*)) = EW(T(OPT)) - C(l^*) + C(v^r) + C(l_{OPT}) - C(v^r)$$

$$= EW(T(OPT)) - C(l^*) + C(l_{OPT})$$

$$> EW(T(OPT))$$

the above equation contradict the case that l_{OPT} is excluded from OPT. Suppose v^r is an ancestor of $v_1{}^r$, then consider the solution by excluding l^* from OPT and add l_{OPT}. The weight of the new solution S^* containing l_{OPT} is given below.

$$EW(T(S^*)) = EW(T(OPT)) - C(l^*) + C(v_1{}^r) + C(l_{OPT}) - C(v^r)$$

$$> EW(T(OPT)) - C(l^*) + C(l_{OPT})$$

$$> EW(T(OPT))$$

expected weight of S^* is more than OPT is a contradiction.

Greedy choice is proved to be correct. The optimal substructure of tree T is to reduce the weight of the nodes in the path from $lOPT$ to root node, to zero. This procedure is repeated for k times. The running time of the greedy algorithm is $O(kn)$.

IV. PRELIMINARY INVESTIGATION AND FUTURE DIRECTION

In this section, we present our preliminary investigation and direction for future work. The facility location problem is generally posted on a single facility type. There are some variants which are provided with multiple facility types. In this case, each facility has separate underlying networks. We have argued some preliminary works related to multiple facility types and other ways to compute facility location.

4.1 MAX-WT-K-LEAF-SUBTREE PROBLEM ON 2-ROOTED TREE

Suppose the network has different kinds of facilities that want to serve to a demand point. The MAX-EXP-COVER- R problem can be formulated as Max-Exp-Cover-R problem with l-type facility. For each facility type, a separate underlying network with same vertex set will be given. The demand on vertices and the probability of link failure is different for each underlying networks. The MAX-EXP-COVER problem is linear time solvable for $l = 1$ is given in this survey, Section 3.The complexity status for $l \geq 2$ is unknown.

A 2-rooted tree is tree in which two special nodes are singled out, called roots. The tree has disjoint node set except at leaves. Once we removed leaves from 2-rooted tree, the tree become two disjoint binary tree with single root each. Tree is rooted by two nodes, v_1 and u_1, and except leaf level, both the induces trees are disjoint.

We presented a dynamic programming and greedy algorithm for MAX-WT-K-LEAF-SUBTREE problem on rooted tree. We have constructed a counter example which shows that greedy algorithm fails to solve the MAX-WT-K-

LEAF- SUBTREE problem on 2-trees.

4.2 LP for MAX-WT-K-LEAF-SUBTREE problem

Another natural way to solve the optimization problems is linear programming. We present an LP formulation for the MAX-WT-K-LEAF-SUBTREE problem. The input is a rooted complete binary tree T, consists of node set $V = \{v_1, v_2, ..., v_n\}$. Each node v_i is associated with a weight $w_i \geq 0$. Let $L \subset V$ be the set of leaf nodes. Given a tree T with node set V and leaves L, and an integer k ($k \leq |L|$), find subset of leaves $L^r \subseteq L$ with $|L^r| \leq k$, such that the total weight of the rooted induced subtree of L^r is maximized.

Let x_i be the decision variable associated with each vertex $v_i \in V$. The x_i decides that whether v_i is included in the solution or not.

$$x_i = \begin{cases} 1, & \text{if } v_i \text{ presented in the induced subtree} \\ 0, & \text{otherwise} \end{cases}$$

The objective function is to maximize the total weight of nodes included in the induced subtree such that at most *k* leaves are picked in the subtree. So objective function is given as,

$$\text{Maximize} \sum_{v_i \in V} x_i . w_i$$

$$\text{s.t,} \sum_{v_i \in L} x_i \leq k \qquad (4.1)$$

$$x_{l(i)} + x_{r(i)} \geq x_i \qquad \forall i \in V \mid L \qquad (4.2)$$

$$x_{p(i)} \geq x_i \qquad \forall i \in V \mid \{r\} \qquad (4.3)$$

$$x_i = \{0, 1\} \qquad \forall i \in V \mid \{r\} \qquad (4.4)$$

$$x_r = 1 \qquad r \text{ is the root} \qquad (4.5)$$

$p(i)$, $l(i)$ and $r(i)$ represent the parent node, left child and right child of node *i* respectively. The equation (4.1) enforce the constrain that for any feasible solution consists of at most *k* leaves. The equation (4.3) stands for any node *i* is presented in solution then it enforce that all its predecessors must be presented in the solution. The equation 4.2 stands for the constraint that, if none of the children has not been selected to the solution, then parent node will not be selected.

4.3 Role of PCDS and BCDS in MAX-EXP-COVER-R problem

Recall that the distance factor *R* is used in facility location under link failure problem. A demand node can be served by a potential facility node if the distance between them is less than or equal to *R*. In case of networks with un weighted edges, the distance is measured as edge count along the path between the nodes. The MAX-EXP-COVER-R problem is known to be NP-hard [1]. Let the distance factor *R* is relaxed from it, then the problem becomes MAX-EXP-COVER problem. Suppose the distance factor $R = 1$, then each potential facility node can cover at most the distance of one hop. The MAX-EXP-COVER-R problem can be modeled as a BCDS problem such that each potential node can have profit of its immediate neighborhood demand and the budget is same as *k*. The solution of the BCDS problem [5] is a connected minimum dominating set that dominates maximum number of vertices, but the solution of MAX-EXP-COVER-R problem is not necessarily connected. For arbitrary *R* value, the instance of Max-Exp-Cover-R can be modeled as a *R*-dominating set (a dominating set $D \subseteq V$ such that the each dominants can dominate up to the R^{th} neighbor) [3].

V. CONCLUSION

The dominating sets and its variants are used in facility location problem with link failure model. The VB-dependency is a well studied link failure model used in facility location problem. The MAX-EXP-COVER-R problem is addressed in facility location under VB-dependency. The dominating sets and its variants can be used in the MAX-EXP-COVER- R problem. The facility location problem can be formulated to multiple kind of facilities where each can have a different network structure. This problem is addressed in MAX-EXP-COVER-R problem with *l*-type facilities.

REFERENCES

[1] Gerard Cornuejols, Marshall L. Fisher, and George L. Nemhauser. *"location of bank accounts to optimize float: An analytic study of exact and approximate algorithms.*" Management Science, 23(8):789–810, 1977.

[2] Michael R. Garey and David S. Johnson. "*Computers and Intractability: A Guide to the Theory of NP-Completeness.*" W.H. Freeman and Co., San Francisco, CA, 1979.

[3] S. Guha and S. Khuller. "*Approximation algorithms for connected dominating sets*", Algorithmica, 20(4):374–387, 1998.

[4] Refael Hassin, R. Ravi, and F. Sibel Salman. "*Tractable Cases of Facility Location on a Network with a Linear Reliability Order of Links*", pages 275–276. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[5] Hervé Kerivin and A. Ridha Mahjoub. "*Design of survivable networks: A survey",* Netw., 46(1):1–21, August 2005.

[6] Samir Khuller, Manish Purohit, and Kanthi K. Sarpatwar. "*Analyzing the optimal neighborhood: Algorithms for budgeted and partial connected dominating set problems",* In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '14, pages 1702–1713, Philadelphia, PA, USA, 2014. Society for Industrial and Applied Mathematics.

[7] F Sibel Salman and Eda Yücel. "*Emergency facility location under random network damage: Insights from the istanbul case",* Computers & Operations Research, 62:266–281, 2015.

Authors Profile

Mr. Vairaprakash Gurusamy pursed MCA from Bharathidasan University, Trichy in 2010. He is currently pursuing Ph.D from Madurai Kamaraj University, Madurai, India. He has published more than 10 research papers in reputed international journals like Scopus Indexed, UGC approved, SCI Indexed, Web of Science, Thomson Reuters etc. His main research work focuses on Bid Data Analytics, Distributed System, Artificial Intelligence, NLP, Cloud Computing, Data Mining and IOT. He has 3 years of Industry Experience and 4 years of Research Experience.

Ms. K. Nandhini pursed B.Tech (ECE) from Kalasalingam University, Krishnan Kovil, India in 2014. She has published more than 10 research papers in reputed international journals like Scopus Indexed, UGC approved, SCI Indexed, Web of Science, Thomson Reuters etc. His main research work focuses on Bid Data Analytics, Distributed System, Artificial Intelligence, NLP, Cloud Computing, Data Mining and IOT. She has 3 years of Industry Experience.

