韓國經營科學會誌
第29卷 第1號
2004年 3月

트리 네트워크 상에서의 p-미디안 문제에 대한 효율적인 알고리즘 개발에 관한 연구*

조 건**

A Study on Developing an Efficient Algorithm for the p-median Problem on a Tree Network*

Geon Cho**

Abstract

Given a tree network on which each node has its own demand and also stands for a candidate location of a potential facility, such as plant or warehouse, the p-median problem on the network (PMPOT) is to select less than or equal to p number of facility locations so that the whole demand on a node is satisfied from only one facility and the total demand occurred on the network can be satisfied from those facilities with the minimum total cost, where the total cost is the sum of transportation costs and the fixed costs of establishing facilities.

Tamir(1996) developed an $O(pn^2)$ algorithm for PMPOT which is known to be the best algorithm in terms of the time complexity, where n is the number of nodes in the network, but he didn't make any comments or explanation about implementation details for finding the optimal solution. In contrast to Tamir's work, Kariv and Hakimi(1979) developed $O(p^2n^2)$ algorithm for PMPOT and presented $O(n^2)$ algorithm for finding the optimal solution in detail.

In this paper, we not only develop another $O(pn^2)$ dynamic programming algorithm for PMPOT that is competitive to Tamir's algorithm in terms of the time complexity, but also present $O(n)$ algorithm that is more efficient than Kariv and Hakimi's algorithm in finding the optimal solution. Finally, we implement our algorithm on a set of randomly generated problems and report the computational results.

Keyword : p-median, facility location, dynamic programming, tree network

논문접수일 : 2003년 3월 11일 논문게재확정일 : 2004년 1월 27일
* 이 논문은 2002년도 한국학술진흥재단의 지원에 의하여 연구되었음(KRF-2002-013-C00078).
** 전남대학교 경영대학 경영학부 부교수, 경영연구소 상임연구원

1. 서 론

기업의 생산시설 및 물류센터 입지선정은 기업의 중요한 장기 의사결정문제 중의 하나로써 일정 지역에서 발생되는 수요를 충족시키면서 동시에 토지 구입비, 시설물 건축비 등과 관련된 초기투자 비용과 인건비, 창고 및 보관비 등과 관련된 운영비용, 그리고 수송비로 대표되는 물류비용 등을 고려하여 총비용을 최소화할 수 있도록 전략적으로 결정되어진다. 그런데 이와 같은 여러 가지 비용요인들 중 특히 물류비용은 해마다 큰 폭으로 증가하는 추세에 있는 것으로 나타났는데, 특히 물류비용 중 수송비가 가장 큰 비중을 차지하고 있는 것으로 나타났다. 이와 같은 수송비의 계속적인 증가 요인으로는 고객의 다양한 욕구에 따른 소량다빈도 운송니즈의 증대와 도로, 철도, 항만 등과 같은 기반시설의 열악성 등을 들 수 있겠지만, 무엇보다도 기업들의 생산시설 및 물류센터 입지선정 및 운영에 관한 효율적인 방안의 부재가 주요 원인으로 간주되어진다. 따라서 본 연구에서는 이에 대한 효율적인 방안 제시의 일환으로 전형적인 입지선정문제(facility location problem) 중의 하나인 p-미디안 문제(p-median problem)를 트리 네트워크(tree network) 상으로 한정하여 이론적·실험적 측면에서 동시에 효율적인 다항식 시간 알고리듬(polynomial time algorithm)을 개발해 보고자 한다.

입지선정문제는 일반적으로 목적함수와 제약식의 다양한 형태에 따라 제한용량이 없는 입지선정문제(uncapacitated facility location problem : UFLP), 제한용량이 있는 입지선정문제(capacitated facility location problem : CFLP), p-센터 문제(p-center problem), p-미디안 문제 등으로 구분되어진다. 특히 p-미디안 문제는 공장, 창고, 물류센터 또는 공공시설 등을 설치할 수 있는 후보입지가 주어져 있다고 가정하고 각 후보입지는 소비자 수요 발생지역을 나타내며 각 시설로부터 각 소비자에게 제품을 수송할 때 소요되는 단위 당 수송비와 수송거리가 주어져 있다고 가정할 때, 최소의 수송비용으로 모든 소비자의 수요를 충족시킬 수 있는 p개 이하의 시설 설치 입지를 결정하는 문제로써, 경찰서, 소방서, 전화국, 공공의료시설, 환경처리시설 등과 같은 공공시설이나 백화점, 대형할인매장, 자동차영업소 등과 같이 경쟁사들과의 경쟁이 치열한 민간시설의 입지선정문제나 통신 및 전력수송 집선장치 위치선정문제(concentrator location problem), 파이프라인 시스템 설계문제(pipeline system design problem) 등과 같은 많은 응용분야에서 자주 발생되는 문제이다.

p-미디안 문제는 이를 구성하고 있는 기반 네트워크(underlying network)의 형태에 따라 보통 일반 네트워크 상에서의 p-미디안 문제(p-median problem on a general network : PMPOG)와 트리 네트워크 상에서의 p-미디안 문제(p-median problem on a tree network : PMPOT)로 구분된다. PMPOG는 Kariv and Hakimi(1979), Garey and Johnson(1979) 등에 의해 NP-hard임이 증명되었으며, 수많은 휴리스틱(heuristic) 알고리듬과 모조다항식 시간(pseudo-polynomial time) 알고리듬들이 문헌상에 소개되어 있다. Ernst and Krishnamoorthy(1998)는 수송 및 통신망 설계에서 자주 발생하는 제한용량이 없는 p-허브 미디안 문제(uncapacitated p-hub median problem)를 최단거리 문제(shortest path problem)와 분단탐색법(branch-and-bound method)을 활용하여 최적해를 구하는 해법을 제시하였으며, Baldacci *et al.*(2001)은 제한용량이 있는 p-미디안 문제(capacitated p-median problem)를 집합분할문제(set partitioning problem)로 공식화하여 이를 푸는 최적 알고리듬을 제시하였다. Drezner(1995)는 몇몇 시설들의 위치가 이미 정해져 있다는 가정하에 p개의 새로운 시설에 대한 위치를 결정하는 조건부 p-미디안 문제(conditional p-median problem)에 대한 휴리스틱 알고리듬을 개발하였다. 그밖에도 Christofides and Beasley(1982), Boffey *et al.*(1984), Ahn *et al.*(1988), Hribar and Daskin(1997) 등이 p-미디안 문제를 푸는 휴리스틱 알고리듬들을 개발하였다.

PMPOT는 0-1 선형정수계획 모형(0-1 linear integer programming model)으로 공식화할 수 있으며 또한 트리 구조의 특성을 잘 활용함으로써 다항식 시간(polynomial time)으로 최적해를 구하는 알고리듬들이 문헌상에 많이 소개되어 있다. Cavalier and Sherali(1986)는 수요밀도함수(demand density function)가 구분적 일양(piecewise uniform)임을 가정하고 트리 그래프(tree graph) 상에서의 2-미디안 문제와 체인 그래프(chain graph) 상에서의 확률적 p-미디안 문제(stochastic p-median problem)를 푸는 최적 알고리듬(exact algorithm)을 설계하였으며, Burkard *et al.*(2000)은 핵발전소(nuclear plant), 신병 훈련소(military depot), 쓰레기 매립장(garbage dump) 등과 같은 기피시설(obnoxious facility)의 입지선정문제와 관련된 트리 상에서의 양수/음수의 가중치를 갖는 2-미디안 문제를 푸는 $O(n^2)$ 시간 알고리듬을 개발하였다. 그러나 전형적인 PMPOT를 푸는 대표적인 알고리듬으로는 Kariv and Hakimi(1979)의 $O(p^2n^2)$시간 알고리듬과 Hsu(1982)의 $O(pn^3)$ 시간 알고리듬, 그리고 Tamir(1996)의 $O(pn^2)$시간 알고리듬 등이 있다(여기서 n은 트리 네트워크 상의 노드의 수를 나타내며 각 노드는 생산시설의 후보입지를 의미함.)

Tamir(1996)의 $O(pn^2)$시간 "leaves to root" 동적계획 알고리듬은 이론적으로 PMPOT의 최적값(optimal value)을 계산하는 알고리듬 중 가장 좋은 시간 복잡도(time complexity)를 갖는 알고리듬으로 알려져 있는데, 여기서 그는 각 노드간 거리를 $O(n^2)$시간의 정렬 알고리듬(sorting algorithm)을 이용하여 선행처리과정(preprocessing)에서 정렬한 후 이를 순환식(recursive formula)에 활용하고 있는 것이 특징이다. 그러나 Tamir(1996)는 트리의 노드 집합이 특별히 정의된 두 개의 노드 집합으로 분할(partition)될 수 있음을 이용하여 단지 그가 제시한 알고리듬의 시간 복잡도가 Kariv and Hakimi(1979)의 알고리듬보다 이론적으로 우월함을 증명하였을 뿐 그의 알고리듬에 대한 구체적인 실행 방법에 대해서는 전혀 언급하지 않고 있다. 따라서 그의 알고리듬에 대한 계산 실험 측면에서의 효율성을 검증하기가 어려울 뿐 만 아니라 많은 응용분야에서 발생하는 현실적인 문제들을 풀기 위해 이를 직접 활용하는 데에도 많은 어려움이 있는 것으로 판단된다. 반면에 Kariv and Hakimi (1979)는 비록 시간 복잡도에서는 Tamir(1996)에 뒤지지만 그들이 개발한 알고리듬의 실행(implementation) 방법 및 과정을 상세히 소개함으로써 계산 실험(computational test) 방법을 제시하고 있으며 또한 $O(n^2)$시간으로 최적해(optimal solution)를 구하는 알고리듬을 제시하고 있는 것이 특징이다.

따라서 본 연구에서는 이론적으로 Tamir(1996)의 알고리듬과 동일한 시간 복잡도를 갖는 효율적인 동적계획 알고리듬을 개발함은 물론 알고리듬의 실행 방법을 구체적으로 설명하며 Kariv and Hakimi(1979)가 제시한 최적해를 구하는 알고리듬보다 우월한 $O(n)$시간 알고리듬을 개발함으로써 PMPOT는 물론 이와 관련된 다양한 응용문제를 푸는데 좀 더 용이하게 활용될 수 있도록 하고자 한다. 이를 위해 본 연구에서는 매 연산(each iteration)마다 완전부분트리(complete subtree)의 루트(root)를 역깊이우선탐색순서(reversed depth first search order)를 따라 이동하면서 최적값(optimal value)을 완전부분트리 상으로 한정하여 계산하는 순환식을 제시한다. 또한 Tamir(1996) 및 Halldorsson *et al.*(1999)이 제시한 트리상에서의 노드 집합 분할에 대한 분석 기법을 활용함으로써 본 연구에서 제시한 PMPOT의 최적값을 계산하는 동적계획 알고리듬의 시간 복잡도가 $O(pn^2)$임을 보이고자 한다.

본 논문은 다음과 같이 구성된다. 2장에서는 PMPOT의 0-1 선형정수계획 모형을 소개하고 3장에서는 먼저 일반 트리 네트워크가 많아야(at most) n-3개의 가상 노드(dummy node)를 이용함으로써 이분 트리(binary tree)로 전환될 수 있음을 보

이고 이를 이용하여 PMPOT의 최적값을 계산하는 동적계획 알고리듬과 최적해를 구하는 알고리듬을 제시하며 또한 이들의 시간 복잡도가 각각 $O(pn^2)$, $O(n)$임을 보인다. 4장에서는 본 연구에서 제시한 알고리듬을 C언어로 코딩하여 여러 가지 유형의 문제들에 대한 테스트 결과를 제시하며 마지막 5장은 결론으로 한다.

2. PMPOT의 0-1 선형정수계획 모형

무방향 트리 네트워크(undirected tree network) $\hat{T}=(V,E)$가 주어져 있다고 가정하자. 이때 $V=\{0,1,2,\cdots,n\}$는 입지후보지 및 소비자 수요 발생지역을 나타내는 노드들(nodes)의 집합이고, 이 노드들에는 깊이우선탐색순서(depth first search order)에 의해 0부터 n까지 번호가 부여되어 있다고 가정한다. $E=\{(p_i,i)\mid i\in V\}$는 임의의 두 노드를 연결하는 아크들(arcs)의 집합으로써 p_i는 노드 i의 직상위 선행노드(predecessor node)를 나타낸다. 이 때 노드 0은 루트 노드(root node)를 나타내며 명백히 p_0는 정의되지 않지만, 편의상 $p_0=-1$이라고 가정한다. 트리 네트워크 상에서는 임의의 두 노드 $i,j(j<i)$를 연결하는 경로(path) $\gamma[j,i]$(혹은 $\gamma[i,j]$)가 유일하게 존재하며 또한 E와 $V\backslash\{0\}$은 일대일 대응(one-to-one correspondence)관계에 있기 때문에 모든 $i\in V\backslash\{0\}$에 대하여 아크 (p_i,i)를 간략히 아크 i로 표시할 수 있다. 또한 본 연구에서는 다음과 같은 두 가지 가정을 설정한다.

(1) (수요 비분리 가정 : indivisible demand assumption) 모든 노드에 대하여, 각 노드에서 요구되는 총수요량은 한 생산시설로부터 전부 공급되어진다.

(2) (인접성 가정 : contiguity assumption) 만약 노드 j에서 요구되는 총수요량이 노드 i에 위치한 생산시설로부터 공급되어진다면 경로 $\gamma[j,i]$상에 있는 모든 노드에서 요구되는 총수요량 또한 노드 j로부터 공급되어진다.

수요 비분리 가정은 각 지역에서 발생되는 수요량이 전량 단일 생산시설로부터 충족되어짐을 의미하며, 인접성 가정은 트리구조를 갖는 네트워크의 특성(즉, 임의의 두 지역을 연결하는 경로는 유일하다)을 잘 반영하는 매우 현실적인 가정으로써 한 지역에서 발생되는 수요량이 한 생산시설로부터 충족되어지는 경우 이 두 지역을 연결하는 경로상에 있는 지역에서 발생되는 모든 수요량도 동일한 생산시설로부터 충족되어짐을 의미한다. 이 두 가지 가정 하에서는 한 생산시설로부터 총수요량을 공급받는 노드들의 집합과 그들을 연결하는 아크들의 집합은 주어진 네트워크 구조인 트리의 부분트리(subtree)를 이루고 있음을 알 수 있으며, 이는 PMPOT가 p개 이하의 생산시설을 중심으로 하여 트리를 부분트리로 분할하는 일종의 트리분할문제(tree partitioning problem)임을 알 수 있다.

임의의 두 노드 i, j에 대하여 노드 i에 대한 노드 j의 직상위 선행노드 p_j^i는 $\gamma[j,i]$상에 있는 노드들 중에서 j에 가장 가까운 노드로 정의한다. 따라서 $p_j=p_j^0$임을 알 수 있으며 p_i^i는 정의되지 않으나, 편의상 $p_i^i=-1$이라고 가정한다.

● 기호 정의

w_i : 노드 i의 총수요량

f_i : 노드 i에 위치한 생산시설 설치비용

d_i : 아크 i의 거리

$d(i,j)=\sum_{v\in\gamma[i,j]} d_v$: 노드 i, j 간의 거리

c_{ij} : 노드 i에 위치한 생산시설로부터 노드 j의 총수요량을 충족시키는데 소요되는 총 수송비용

$$x_{ij}=\begin{cases}1, & \text{만약 노드 } i \text{ 에 있는 생산시설이 노드 } j \text{ 의 총수요를 충족시키면,}\\ 0, & \text{그렇지 않으면.}\end{cases}$$

$$y_i=\begin{cases}1, & \text{만약 노드 } i \text{ 에 생산시설이 설치되면,}\\ 0, & \text{그렇지 않으면.}\end{cases}$$

이 때 w_i, f_i는 음이 아닌 정수들(non negative integers)로 가정하며 x_{ij}, y_i는 결정변수들(decision variables)을 나타낸다. 또한 $g_j(\cdot)$를 비선형 증가함수(nonlinear increasing function)라고 할 때 일반적으로 $c_{ij}=g_j(d(i,j))$로 나타낼 수 있는데, 특별히 $g_j(\cdot)$가 선형함수인 경우에는 단위수요/단위거리당 수송비용이 일정하다고 가정할 때 $c_{ij}=w_j\times d(i,j)$로 표시될 수 있다. 본 연구에서는 편의상 $g_j(\cdot)$는 선형함수인 것으로 가정한다. 이제 (*PMPOT*)를 0-1 선형정수계획 모형으로 공식화하면 다음과 같다.

(*PMPOT*)

$$\min \sum_{i=0}^{n} f_i y_i + \sum_{i=0}^{n}\sum_{j=0}^{n} c_{ij}x_{ij}$$

$$\text{s.t.} \sum_{i=0}^{n} x_{ij}=1, \qquad j=0,1,2,\cdots,n \qquad (1)$$

$$x_{ij} \le x_{ip_j'}, \qquad i,j=0,1,2,\cdots,n \qquad (2)$$

$$x_{ij} \le y_i, \qquad i,j=0,1,2,\cdots,n \qquad (3)$$

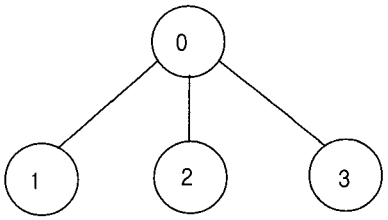
$$\sum_{i=0}^{n} y_i \le p \qquad (4)$$

$$x_{ij}, y_i \in \{0,1\}, \quad i,j=0,1,2,\cdots,n$$

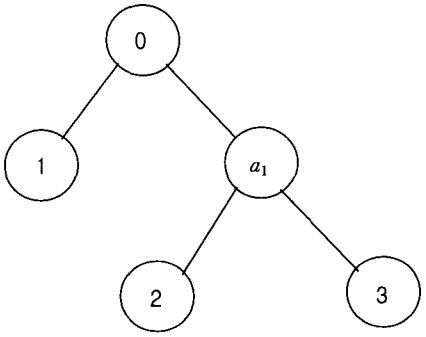
(*PMPOT*)에서 목적함수는 생산시설 설치비용과 설치된 생산시설로부터 각 노드의 총수요량을 충족시키는데 소요되는 총비용의 합을 나타낸다. 제약식 (1)은 각 노드에서 발생되는 총수요량이 한 생산시설로부터 전량 충족되어짐을 나타내고(indivisible demand constraints), 제약식 (2)는 만약 한 노드의 총수요량이 한 생산시설로부터 충족된다면 그 노드의 직상위 노드 또한 동일한 생산시설로부터 충족됨을 의미하며 결과적으로 그 노드와 그 생산시설을 연결하는 경로 상에 있는 모든 노드의 총수요량이 동일한 생산시설로부터 충족됨을 나타낸다(contiguity constraints). 또한 제약식 (3)은 노드 i에 생산시설이 설치되지 않으면 어떤 노드에서 발생되는 수요도 노드 i로부터 충족될 수 없음을 나타내며(즉, $y_i=0$이면 $x_{ij}=0$임을 나타냄), 제약식 (4)는 총 생산시설의 개수가 미리 주어진 수 p보다 크지 않아야 함을 나타낸다.

3. 동적계획 알고리듬

일반적으로 n개의 노드를 갖는 트리 네트워크는 많아야(at most) n-3개의 가상 노드(dummy node)를 이용함으로써 이분 트리(binary tree)로 전환될 수 있음을 쉽게 알 수 있다. 예컨대, [그림 1]과 같은 일반 트리가 주어져 있다고 가정할 때 가상노드 a_1을 이용하면 [그림 2]와 같이 이분 트리로 전환할 수 있으며, 이 때 이분 트리의 가상 노드 a_1과 관련하여 $d(0,a_1)=0$, $d(a_1,2)=d(0,2)$, $d(a_1,3)=d(0,3)$, $w_{a_1}=0$, $f_{a_1}=\infty$로 각각 정의하면 원래 주어진 일반 트리 상에서의 (*PMPOT*)를 이분 트리 상에서의 (*PMPOT*)로 전환할 수 있음을 알 수 있다. 따라서 본 연구에서는 편의상 $\hat{T}$를 이분 트리로 가정한다.



[그림 1] 일반 트리 구조



[그림 2] 이분 트리 구조

또한 임의의 트리 T에 대해 $|T|$는 T 내에 있는 노드의 개수를 나타내며 $r(T)$는 T의 루트노드(즉, $r(T)=\min\{j \mid j\in T\}$)를 나타내고 두 집합 V_k, E_k를 각각 $V_k=\{j\in V \mid k\in\gamma[0,j]\}$, $E_k=\{(p_i,i)\mid i\in V_k\backslash\{k\}\}$라고 정의할 때 $T(k)=(V_k,E_k)$는 $r(T(k))=k$인 $\widehat{T}$의 완전부분트리(complete subtree)가 됨을 알 수 있다. 이제 PMPOT를 푸는 동적계획 알고리듬을 구체적으로 제시하기 전에 이에 관련된 순환식(recursion formula)을 먼저 제시하고자 한다. 이를 위해(*PMPOT*)를 $T(k)$상으로 제한하여 얻은 부분문제 $(PMPOT)_{T(k)}$에 대한 최적값(optimal value)을 다음과 같이 두 가지로 정의해야 하는데 이는 $T(k)$의 루트 노드 k에서 발생하는 수요를 충족시키는 최적의 생산시설에 대한 위치 i를 찾기 위해서는 명백히 $i\in T(k)$인 경우와 $i\in\widehat{T}\backslash T(k)$인 경우를 동시에 고려해야 하기 때문이다.

(1) $G(i,k,q)$: = 노드 $i\in T(k)$에 생산시설이 설치되어 있고 노드 i가 노드 k에서 발생되는 수요를 충족시키면서 또한 1개 이상 및 q개 이하의 생산시설이 $T(k)$내에 설치될 수 있을 때의 $(PMPOT)_{T(k)}$에 대한 최적값

(2) $F(i,k,q)$: = 노드 $i\in\widehat{T}\backslash T(k)$에 생산시설이 설치되어 있고 노드 i가 노드 k에서 발생되는 수요를 충족시키면서 또한 q개 이하의 생산시설이 $T(k)$내에 설치될 수 있을 때의 $(PMPOT)_{T(k)}$에 대한 최적값

이 때 만약 노드 k가 잎 노드라면 $G(k,k,1)=f_k$임을 알 수 있고 $i\neq k$인 모든 노드 i에 대하여 $F(i,k,0)=w_k\times d(i,k)$라고 정의할 수 있으며 편의상 본 연구에서는 모든 노드 k와 $i\in T(k)$에 대하여 $G(i,k,0)=\infty$로 가정한다. 또한 잎 노드가 아닌 노드(non-leaf node) k의 직하위 후행노드(immediate successor node)를 u, v라고 하면(즉, $p_u=p_v=k$), $G(i,k,q)$와 $F(i,k,q)$값은 적절한 q'에 대응되는 $(PMPOT)_{T(u)}$와 $(PMPOT)_{T(v)}$의 최적값 $G(i,u,q')$, $G(i,v,q')$, $F(i,u,q')$, $F(i,v,q')$ 등을 이용하여 다음과 같이 구할 수 있다.

Case 1) $i=k$; 이 경우에는 이미 한 생산시설이 루트노드 k에 설치된 경우이므로 $G(k,k,q)$값은 집합 $\Omega_1=\{(q_1,q_2)\mid q_1+q_2=q-1,\ q_1\le|T(u)|,$ and $q_2\le|T(v)|\}$의 모든 원소 (q_1,q_2)에 대해 다음과 같은 네 가지 경우에 대한 최소값을 계산함으로써 구할 수 있다([그림 3] 참조).

(1) k가 u,v에서 발생되는 수요를 모두 충족시키는 경우(즉, $F(k,u,q_1)+F(k,v,q_2)$를 계산)

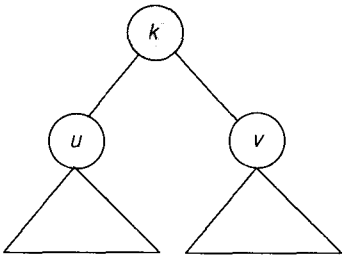
(2) k가 u에서 발생되는 수요만을 충족시키고 v에서 발생되는 수요는 충족시키지 않는 경우 (즉, $F(k,u,q_1)+\min_{j\in T(v)} G(j,v,q_2)$를 계산)

(3) k가 v에서 발생되는 수요만을 충족시키고 u에서 발생되는 수요는 충족시키지 않는 경우 (즉, $\min_{j\in T(u)} G(j,u,q_1)+F(k,v,q_2)$를 계산)

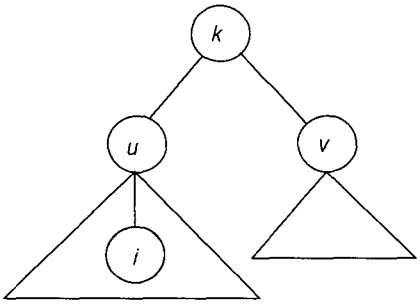
(4) k가 u,v에서 발생되는 수요를 모두 충족시키지 않는 경우 (즉, $\min_{j\in T(u)} G(j,u,q_1)+\min_{j\in T(v)} G(j,v,q_2)$를 계산)

Case 2) $i\in T(u)$; 이 경우에는 이미 한 생산시설이 $T(u)$상의 노드 i에 설치된 경우이므로 인접성 가정에 의해 i는 u에서 발생되는 수요를 충족시킨다. 따라서 $G(i,k,q)$는 집합 $\Omega_2=\{(q_1,q_2)\mid q_1+q_2=q, 1\le q_1\le|T(u)|,$ and $q_2\le|T(v)|\}$의 모든 원소 (q_1,q_2)에 대해 다음과 같은 두 가지 경우에 대한 최소값을 계산함으로써 구할 수 있다([그림 4] 참조).

(1) i가 v에서 발생되는 수요를 충족시키는 경우 (즉, $G(i,u,q_1)+F(i,v,q_2)$를 계산)



[그림 3] 생산시설이 루트 노드 k에 설치된 경우



[그림 4] 생산시설이 $T(v)$상의 노드 i에 설치된 경우

(2) i가 v에서 발생되는 수요를 충족시키지 않는 경우(즉, $G(i,u,q_1)+\min_{j\in T(v)} G(j,v,q_2)$를 계산)

Case 3) $i\in T(v)$; 이 경우는 Case 2)에서와 마찬가지로 집합 $\Omega_3=\{(q_1,q_2)\mid q_1+q_2=q,\ q_1\le|T(u)|,\ \text{and}\ 1\le q_2\le|T(v)|\}$의 모든 원소 (q_1,q_2)에 대해 다음과 같은 두 가지 경우에 대한 최소값을 계산함으로써 $G(i,k,q)$값을 구할 수 있다([그림 5] 참조).

(1) i가 v에서 발생되는 수요를 충족시키는 경우 (즉, $F(i,u,q_1)+G(i,v,q_2)$를 계산)

(2) i가 v에서 발생되는 수요를 충족시키지 않는 경우(즉, $\min_{j\in T(u)} G(j,u,q_1)+G(i,v,q_2)$를 계산)

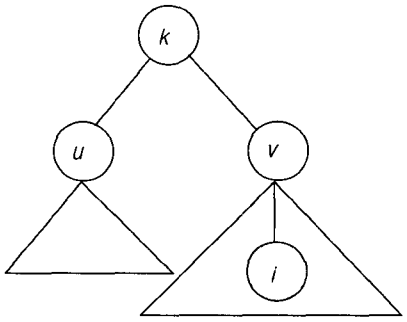
Case 4) $i\notin T(k)$; 이 경우에도 Case 1)에서와 유사하게 집합 $\Omega_4=\{(q_1,q_2)\mid q_1+q_2=q,\ q_1\le|T(u)|,\ \text{and}\ q_2\le|T(v)|\}$의 모든 원소 (q_1,q_2)에 대해 다음과 같은 네 가지 경우에 대한 최소값을 계산함으로써 $F(i,k,q)$값을 구할 수 있다([그림 6] 참조).

(1) i가 u, v에서 발생되는 수요를 모두 충족시키는 경우(즉, $F(i,u,q_1)+F(i,v,q_2)$를 계산)

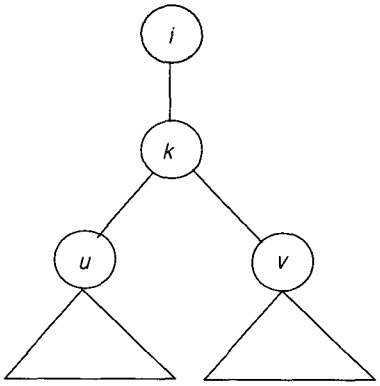
(2) i가 u에서 발생되는 수요만을 충족시키고 v에서 발생되는 수요는 충족시키지 않는 경우(즉, $F(i,u,q_1)+\min_{j\in T(v)} G(j,v,q_2)$를 계산)

(3) i가 v에서 발생되는 수요만을 충족시키고 u에서 발생되는 수요는 충족시키지 않는 경우(즉, $\min_{j\in T(u)} G(j,u,q_1)+F(i,v,q_2)$를 계산),

(4) i가 u,v에서 발생되는 수요를 모두 충족시키지 않는 경우
(즉, $\min_{j\in T(u)} G(j,u,q_1)+\min_{j\in T(v)} G(j,v,q_2)$를 계산)



[그림 5] 생산시설이 $T(v)$상의 노드 i에 설치된 경우



[그림 6] 생산시설이 $T(k)$밖에 있는 노드 i에 설치된 경우

이상을 요약하여 정리하면 다음과 같이 $(PMPOT)_{T(u)}$와 $(PMPOT)_{T(v)}$의 최적값들을 이용하여 $(PMPOT)_{T(k)}$에 대한 최적값 $G(i,k,q)$와 $F(i,k,q)$를 구하는 순환식을 얻을 수 있다.

● 순환식

(1) (초기화)

모든 잎 노드(leaf node) j와 $i \neq j$인 모든 노드 i에 대하여

$$G(j,j,1) := f_j ;$$
$$F(i,j,0) := g_j(d(j,i)) = w_j \times d(j,i) ;$$

모든 노드 k와 $i \in T(k)$에 대하여

$$G(i,k,0) := \infty ;$$

(2) 잎 노드가 아닌 노드(non-leaf node) k의 직하위 후행노드를 u, v라 하면 생산시설이 위치한 노드 i에 대해 다음과 같은 네 가지 경우가 발생한다.

Case 1) $i = k$;

$$G(k,k,q) := f_k + \min_{(q_1,q_2)\in\Omega_1} \left\{ \min\left\{ F(k,u,q_1), \min_{j\in T(u)} G(j,u,q_1) \right\} + \min\left\{ F(k,v,q_2), \min_{j\in T(v)} G(j,v,q_2) \right\} \right\};$$

Case 2) $i \in T(u)$;

$$G(i,k,q) := w_k \times d(k,i) + \min_{(q_1,q_2)\in\Omega_2} \left\{ G(i,u,q_1) + \min\left\{ F(i,v,q_2), \min_{j\in T(v)} G(j,v,q_2) \right\} \right\} ;$$

Case 3) $i \in T(v)$;

$$G(i,k,q) := w_k \times d(k,i) + \min_{(q_1,q_2)\in\Omega_3} \left\{ G(i,v,q_2) + \min\left\{ F(i,u,q_1), \min_{j\in T(u)} G(j,u,q_1) \right\} \right\};$$

Case 4) $i \notin T(k)$;

$$F(i,k,q) := w_k \times d(k,i) + \min_{(q_1,q_2)\in\Omega_4} \left\{ \min\left\{ F(i,u,q_1), \min_{j\in T(u)} G(j,u,q_1) \right\} + \min\left\{ F(i,v,q_2), \min_{j\in T(v)} G(j,v,q_2) \right\} \right\};$$

위에서 소개된 집합 Ω_1, Ω_2, Ω_3, Ω_4는 Tamir (1996)가 제시한 순환식에서 사용된 형태와 매우 유사한 형태로 정의되었는데 이는 후에 본 연구에서 개발한 알고리듬의 시간 복잡도가 $O(pn^2)$임을 증명하는데 매우 중요하게 활용된다. 그럼에도 불구하고 실제로 본 연구에서 제시한 순환식과 Tamir (1996)가 제시한 순환식 자체에는 전체적인 형태로나 개념적인 측면에서 명백히 큰 차이가 있다. 사실 본 연구에서는 최적값을 완전부분트리 상으로 한정하여 계산해 가는 순환식을 제시한 반면 Tamir (1996)는 전체 트리를 대상으로 각 노드간 거리를 $O(n^2)$시간의 정렬 알고리듬을 이용하여 선행처리 과정에서 정렬한 후 이를 순환식에 활용하고 있다는 차이점이 있다.

지금까지 소개한 기호 및 순환식을 바탕으로 본 연구에서 개발한 $(PMPOT)$를 푸는 동적계획 알고리듬의 핵심 아이디어를 설명하고자 한다. $T(k)$ 상에서의 최적값 $G(i,k,q)$와 $F(i,k,q)$를 구하기 위해서는 k의 직하위 후행노드 u, v에 대해 $T(u)$, $T(v)$ 상에서의 최적값들이 먼저 계산되어져야 하기 때문에 알고리듬 ***PMPOT***에서는 $(PMPOT)$에 대한 최적값을 찾기 위해 $T(k)$의 루트 노드 k를 역 DFS 순서(reversed DFS order)를 따라 이동하면서 위의 순환식을 적용하며, 결과적으로 $(PMPOT)$에 대한 최적값은 $\min_{0\le i\le n} G(i,0,p)$으로 구할 수 있다. 그런데 $G(i,k,q)$의 정의에 의해 $i \in T(k)$이고 i에 설치된 생산시설로부터 k의 총수요량이 충족되어짐으로 식 (1)과 식 (2)에 의해 $\gamma[i,k]$상에 있는 모든 노드의 총수요량도 i로부터 충족되어진다. 따라서 $T(k)$상에 설치될 수

있는 최대의 생산시설 개수는 $|T(k)|-|\gamma[i,k]|+1$임을 알 수 있으며 또한 식 (4)에 의해 $T(k)$상에는 p개 이하의 생산시설이 설치되어져야 하므로 $G(i,k,q)$에서 사용된 q값의 상한(upper bound)은 $\min\{p,|T(k)|-|\gamma[i,k]|+1\}$임을 알 수 있다. 그리고 $F(i,k,q)$에 대해서는 $i\in\widehat{T}\setminus T(k)$이며 k의 총수요량이 i에 설치된 생산시설로부터 충족되어짐으로 $T(k)$상에 설치될 수 있는 최대의 생산시설 개수는 $|T(k)|-1$임을 알 수 있으며 결과적으로 식 (4)에 의해 $F(i,k,q)$에서 사용된 q값의 상한은 $\min\{p,|T(k)|-1\}$임을 알 수 있다. 따라서 (*PMPOT*)에 대한 최적값은 각 노드 i에 대해 $q=\min\{p,|T(0)|-|\gamma[0,i]|+1\}$ 값을 이용함으로써 $\min_{0\le i\le n} G(i,0,q)$로 구할 수 있다.

이제 위에서 설명한 내용을 바탕으로 본 연구에서 개발한 알고리듬 ***PMPOT*** 를 다음과 같이 제시하고자 한다.

- **Algorithm *PMPOT***

단계 1) $k:=n$;

단계 2) $s:=k+|T(k)|-1$로 두고 k의 직하위 후행노드를 u, v라고 한다 ;

단계 3) $q:=\min\{p,|T(k)|-|\gamma[s,k]|+1\}$로 두고 다음을 수행한다 ;

부분단계 1) 만약 $s=k$이면 단계 4)로 간다 ;

부분단계 2) $temp_q:=\min\{p,|T(s)|-1\}$로 두고 모든 $\bar{q}=0,1,2,\cdots,temp_q$에 대하여 $F(k,s,\bar{q})$를 계산한다;

부분단계 3) 만약 $s\in T(u)$이면 모든 노드 $i\in T(v)$에 대하여 $temp_q:=\min\{p,|T(i)|-1\}$로 두고 모든 $\bar{q}=0,1,2,\cdots,temp_q$에 대하여 $F(s,i,\bar{q})$를 계산한다 ;

부분단계 4) 만약 $s\in T(v)$이면 모든 노드 $i\in T(u)$에 대하여 $temp_q:=\min\{p,|T(i)|-1\}$로 두고 모든 $\bar{q}=0,1,2,\cdots,temp_q$에 대하여 $F(s,i,\bar{q})$를 계산한다;

단계 4) 모든 $\bar{q}=0,1,2,\cdots,q$에 대하여 $G(s,k,\bar{q})$를 계산한 후 $s:=s-1$로 두고 만약 $s\in T(k)$이면 단계 3)으로 간다;

단계 5) $q:=\min\{p,|T(k)|\}$로 두고 모든 $\bar{q}=0,1,2,\cdots,q$에 대하여 $\min_{j\in T(k)} G(j,k,\bar{q})$를 계산한다 ;

단계 6) $k:=k-1$로 두고 만약 $k\ge 0$이면 단계 2)로 간다 ;

이제 본 연구에서 제시된 (*PMPOT*)에 대한 최적값을 찾는 알고리듬의 시간 복잡도(time complexity)를 계산해 보고자 한다. 알고리듬 ***PMPOT*** 에서는 역 DFS 순서를 따라 부분 트리 $T(k)$의 루트 노드 k를 옮겨가면서 순환식을 적용하여 0 혹은 1부터 q까지의 값에 대한 $G(i,k,q)$와 $F(i,k,q)$ 값을 계산한다. 따라서 먼저 주어진 k와 관련된 모든 i, q값에 대해 $G(i,k,q)$와 $F(i,k,q)$ 값들을 계산하는데 필요한 시간 복잡도를 계산해 보도록 한다. 이미 앞에서도 살펴보았듯이 q에 대한 상한은 $\max\{\min\{p,|T(k)|-|\gamma[i,k]|+1\},\ \min\{p,|T(k)|-1\}\}\le\min\{p,|T(k)|\}$임을 쉽게 알 수 있으며 또한 $G(i,k,q)$와 $F(i,k,q)$값은 Ω_1, Ω_2, Ω_3, 혹은 Ω_4의 원소 (q_1,q_2)에 대해 $G(i,u,q_1)$, $G(i,v,q_2)$, $F(i,u,q_1)$, $F(i,v,q_2)$ 값들을 이용하여 계산되어짐은 물론 q_1,q_2에 대한 상한도 $q_1\le\min\{p,|T(u)|\}$, $q_2\le\min\{p,|T(v)|\}$임을 알 수 있다. 따라서 주어진 k와 관련된 모든 i, q값에 대해 $G(i,k,q)$와 $F(i,k,q)$값을 계산하는데 필요한 시간 복잡도는 $O(|T(k)|\times\min\{p,|T(u)|\}\times\min\{p,|T(v)|\})\approx O(np^2)$임을 알 수 있으며, 결과적으로 이와 같은 단순한 분석에 의하면 본 연구에서 제시된 알고리듬의 시간 복잡도는 $O(n^2p^2)$임을 알 수 있다.

그러나 앞에서도 잠시 언급하였듯이 본 연구에

서 제시한 순환식에 사용된 집합 Ω_1, Ω_2, Ω_3, Ω_4의 형태가 Tamir(1996)가 제시한 순환식에서 사용된 형태와 매우 유사하게 정의되었는데 이를 이용하면 Tamir(1996) 및 Halldorsson *et al.*(1999)이 제시한 트리상에서의 노드 집합 분할에 대한 분석 결과를 적절히 활용함으로써 본 연구에서 개발한 알고리듬의 시간 복잡도가 $O(pn^2)$으로 향상될 수 있음을 증명할 수 있다. 이를 좀 더 구체적으로 살펴보기 위해 먼저 다음의 두 가지 용어를 정의한다. 잎 노드가 아닌 노드 k의 직하위 후행노드를 u, v라 할 때 만약 $|T(u)| \geq \frac{p}{2}$와 $|T(v)| \geq \frac{p}{2}$가 동시에 만족되면 노드 k를 *중량급* 노드(heavy node)라고 하고 그렇지 않으면 *경량급* 노드(light node)라고 정의한다. 이 때 다음과 같은 두 가지 사실이 성립됨을 알 수 있다(증명은 Tamir(1996) 및 Halldorsson *et al.*(1999) 참조).

1) $\hat{T}$상의 모든 중량급 노드의 개수는 많아야 $\frac{2n}{p}$개이다.
2) 임의의 노드 k와 $T(k)$상에 있는 모든 경량급 노드 i와 모든 q값에 대해 $G(i,k,q)$와 $F(i,k,q)$값들을 계산하는데 필요한 시간 복잡도는 $O(p|T(k)|^2)$이다.

따라서 1)에 의해 $\hat{T}$상의 모든 중량급 노드들과 그에 관련된 모든 i, q값에 대해 $G(i,k,q)$와 $F(i,k,q)$값을 계산하는데 필요한 시간 복잡도는 $O(np^2 \times \frac{2n}{p}) \approx O(pn^2)$임을 알 수 있다. 또한 2)에 의해 $\hat{T} = T(0)$상의 모든 경량급 노드들과 그에 관련된 모든 i, q값에 대해 $G(i,k,q)$와 $F(i,k,q)$값을 계산하는데 필요한 시간 복잡도는 $O(p|T(0)|^2) = O(pn^2)$임을 알 수 있으므로 결과적으로 본 연구에서 제시된 알고리듬의 시간 복잡도는 $O(pn^2)$임을 증명할 수 있다.

아래에 제시된 알고리듬 ***Optimal_Solution***에서는 (*PMPOT*)에 대한 최적해를 구한다. 이를 위해 먼저 루트 노드 0에서 발생하는 수요를 충족시키는 생산시설의 위치 i를 찾은 후 노드 1에서부터 시작하여 DFS 순서를 따라 이동하면서 i로부터 충족되어지는 노드들을 모두 찾는다. 이 과정에서 만약 i로부터 충족되지 않는 노드 k가 발견되면 모든 $T(k)$상의 노드들은 i로부터 충족될 수 없으므로 k를 먼저 *STACK*에 저장한 후 노드 $k+|T(k)|$로 이동한 후 i로부터 충족되어지는 노드 찾기를 계속한다. 이제 i로부터 충족되어지는 노드들을 모두 찾고 나면 다시 *STACK*에 저장된 노드들 중 한 노드 k를 선택하여 k에서 발생하는 수요를 충족시키는 생산시설의 위치 i'를 찾은 후 노드 $k+1$에서부터 시작하여 DFS 순서를 따라 이동하면서 i'으로부터 충족되어지는 $T(k)$상의 노드들을 모두 찾는 과정을 반복한다. 따라서 위와 같은 방법을 *STACK*이 공집합(empty set)이 될 때까지 반복하면 각 노드를 한번씩만 체크함으로써 각 노드 j에 대한 *Solution(j)*를 구할 수 있기 때문에 결과적으로 ***Optimal_Solution***의 시간 복잡도는 $O(n)$임을 쉽게 알 수 있다.

이제 *Solution(j)*를 노드 j에서 발생되는 수요를 충족시키는 최적의 생산시설의 위치(최적해)라고 정의할 때 본 연구에서 개발한 알고리듬 ***Optimal_Solution***은 다음과 같다.

- **Algorithm *Optimal_Solution***

단계 1) $k := 0$; $STACK := \{0\}$; $j := 1$;
단계 2) k를 충족시키는 i를 찾아 *Solution(k)* : $= i$로 둔다 ;
단계 3) 만약 i가 j를 충족시키면 *Solution(j)* : $= i$; $j := j+1$로 둔다 ;
단계 4) 만약 i가 j를 충족시키지 않으면 $STACK$: $= STACK \bigcup \{j\}$; $j := j+|T(j)|$로 둔다 ;
단계 5) 만약 $j \in T(k)$이면 단계 3)으로 간다 ;

단계 6) $STACK := STACK \setminus \{k\}$로 두고 만약 $STACK \neq \emptyset$이면 다음을 수행한다;

부분단계 1) $STACK$의 한 원소 k를 선택한다;

부분단계 2) $j := k+1$로 두고 단계 2)로 간다;

4. 계산 실험

본 연구에서 제시된 알고리듬에 대한 효율성 및 계산 실험의 용이성을 검증해 보기 위해 이를 C언어로 코딩하여 다음과 같은 방법으로 생성된 여러 가지 유형의 문제들에 대한 계산 실험(computational test)을 수행하고자 한다.

먼저 트리 $\hat{T}$의 구조를 임의로 결정한다. 이를 위해 트리 $\hat{T}$의 총 노드 수 n을 고정한 후 루트노드로부터 시작하여 직하위노드 수를 구간 $[0, \log_2 n]$에서 임의로 결정하고 이를 넓이우선탐색(breadth first search order : BFS) 순서로 나열한 후, 다시 앞서 생성된 각 노드의 직하위노드 수를 구간 $[0, \log_2 n]$에서 임의로 결정하여 이를 BFS 순서로 나열하는 방법을 반복적으로 수행하는데 이는 생성된 총 노드 수가 n이 될 때까지 반복된다. 다음은 일단 BFS 순서를 갖는 일반 트리가 생성되면 이를 가상 노드를 이용함으로써 DFS 순서를 갖는 이분 트리로 변환한 후 이에 본 알고리듬을 적용하여 최적해를 구한 다음 이를 원래 생성된 BFS 순서를 갖는 일반 트리에 관한 최적해로 변환한다.

본 계산 실험에서는 편의상 w_i를 구간 $[1,50]$에서 임의로 결정하며 f_i를 세 개의 다른 구간 $[0, 1000]$, $[5000, 10000]$, $[10000, 20000]$에서 각각 임의로 결정한 후 이들 각각의 경우에 대하여 d_i를 두 개의 다른 구간 $[1,20]$, $[1,100]$에서 각각 임의로 결정한다. 또한 f_i, d_i 값들이 생성되는 구간에 따라 구분되는 여섯 가지 테스트 문제 유형에 대해 총 노드 수 및 최대 생산시설 설치 개수를 각각 $n = 20, 50, 100$, $p = 1, 2, 5, 10$으로 구분하고 각각의 경우에 대한 테스트 문제를 10문제씩 임의로 생성하여 Intel Petium III 600 MHz 컴퓨터 환경에서 테스트를 실시함으로써 총 노드 수, 최대 생산 시설 설치 개수, 생산시설 설치비용과 수송비용의 크기에 대한 비율의 변화 등이 각각 CPU 시간 및 최적 생산시설 설치 개수에 어떠한 영향을 주는지 살펴보고자 한다.

<표 1>은 테스트 결과로부터 도출된 각 테스트 문제 유형별 CPU 시간과 최적 생산시설 설치 개수들의 최소값(minimum value), 평균값(average value), 최대값(maximum value) 등을 나타내 주고 있다. <표 1>에서 볼 수 있듯이 CPU 시간은 f_i, d_i 값들의 변화에 대해 민감하지 않음을 알 수 있으며, n에 대해서는 근사적으로 제곱의 비율(quadratic rate)로, p에 대해서는 근사적으로 선형의 비율(linear rate)로 변화함을 알 수 있다. 반면에 최적 생산시설 설치 개수는 f_i, d_i 값들의 변화에 대해 매우 민감함을 알 수 있다. 예컨대, 고정된 d_i 값 생성 구간에 대해 생산시설 설치비용 f_i가 증가할수록 생산시설 설치 개수가 상대적으로 감소함을 알 수 있는데 이는 생산시설 설치비용이 증가할수록 가능하면 적은 곳에 생산시설을 설치하여 이로부터 수송을 통해 각 노드에서 발생되는 수요를 충족시키는 것이 최적해가 될 수 있는 가능성이 높다는 것을 알 수 있다. 또한 고정된 f_i 값 생성 구간에 대해 아크의 거리 d_i가 증가할수록 수송비용이 증가하므로 이때의 생산시설 설치 개수는 상대적으로 증가함을 알 수 있는데 이는 가능하면 많은 곳에 생산시설을 설치하여 이들로부터 주변에서 발생되는 수요들을 충족시킴으로써 가급적 수송비용을 줄일 수 있도록 하는 것이 최적해가 될 수 있는 가능성이 높다는 것을 알 수 있다.

5. 결 론

본 연구에서는 먼저 트리 네트워크 상에서의 p-미디안 문제(PMPOT)에 대한 0-1 선형정수계획

〈표 1〉 PMPOT의 각 유형별 계산 실험 결과

n			20				50				100			
p			1	2	5	10	1	2	5	10	1	2	5	10
$f_i \in [0, 1000]$, $d_i \in [1,20]$	CPU 시간 (단위 : 초)	min	0.001	0.010	0.010	0.030	0.060	0.080	0.130	0.210	0.210	0.301	0.591	1.082
		ave	0.010	0.012	0.023	0.036	0.078	0.091	0.165	0.266	0.245	0.337	0.643	1.295
		max	0.020	0.020	0.030	0.051	0.130	0.120	0.201	0.321	0.350	0.400	0.731	1.442
	생산시설 설치개수	min	1	1	1	1	1	1	1	1	1	1	1	1
		ave	1	1	1	3	1	1	2	3	1	1	3	4
		max	1	2	2	6	1	2	4	6	1	2	5	9
$f_i \in [0, 1000]$, $d_i \in [1, 100]$	CPU 시간 (단위 : 초)	min	0.001	0.010	0.020	0.030	0.050	0.080	0.140	0.260	0.200	0.311	0.601	1.242
		ave	0.018	0.012	0.023	0.041	0.066	0.088	0.163	0.308	0.238	0.346	0.684	1.308
		max	0.040	0.030	0.030	0.060	0.120	0.110	0.180	0.361	0.360	0.390	0.792	1.402
	생산시설 설치개수	min	1	1	1	3	1	1	1	1	1	1	2	1
		ave	1	1	2	4	1	1	2	5	1	1	3	5
		max	1	2	5	7	1	2	4	9	1	2	5	9
$f_i \in [5000, 10000]$, $d_i \in [1, 20]$	CPU 시간 (단위 : 초)	min	0.010	0.010	0.020	0.020	0.050	0.070	0.120	0.240	0.220	0.281	0.641	1.112
		ave	0.015	0.017	0.021	0.035	0.079	0.091	0.151	0.302	0.271	0.357	0.741	1.255
		max	0.040	0.040	0.030	0.050	0.170	0.120	0.180	0.411	0.430	0.481	0.861	1.472
	생산시설 설치개수	min	1	1	1	1	1	1	1	1	1	1	1	1
		ave	1	1	1	1	1	1	1	1	1	1	1	1
		max	1	2	2	2	1	1	2	4	1	2	2	4
$f_i \in [5000, 10000]$, $d_i \in [1, 100]$	CPU 시간 (단위 : 초)	min	0.001	0.010	0.010	0.030	0.050	0.070	0.130	0.270	0.211	0.290	0.581	1.151
		ave	0.014	0.012	0.023	0.045	0.064	0.083	0.154	0.328	0.249	0.359	0.668	1.302
		max	0.030	0.020	0.030	0.080	0.090	0.110	0.190	0.430	0.280	0.521	0.761	1.442
	생산시설 설치개수	min	1	1	1	1	1	1	1	1	1	1	1	1
		ave	1	1	1	1	1	1	1	2	1	1	2	2
		max	1	2	4	4	1	2	3	3	1	2	4	4
$f_i \in [10000, 20000]$, $d_i \in [1, 20]$	CPU시간 (단위 : 초)	min	0.001	0.010	0.020	0.030	0.050	0.070	0.140	0.260	0.201	0.301	0.601	1.042
		ave	0.008	0.013	0.024	0.045	0.077	0.090	0.167	0.338	0.240	0.336	0.672	1.247
		max	0.010	0.020	0.030	0.080	0.180	0.121	0.191	0.471	0.300	0.361	0.782	1.552
	생산시설 설치개수	min	1	1	1	1	1	1	1	1	1	1	1	1
		ave	1	1	1	1	1	1	1	1	1	1	1	1
		max	1	1	1	1	1	1	2	2	1	2	1	2
$f_i \in [10000, 20000]$, $d_i \in [1, 100]$	CPU 시간 (단위 : 초)	min	0.001	0.010	0.010	0.030	0.050	0.070	0.140	0.230	0.190	0.310	0.601	1.161
		ave	0.009	0.011	0.024	0.050	0.059	0.084	0.162	0.316	0.264	0.337	0.656	1.302
		max	0.020	0.020	0.040	0.101	0.080	0.110	0.181	0.511	0.480	0.360	0.801	1.583
	생산시설 설치개수	min	1	1	1	1	1	1	1	1	1	1	1	1
		ave	1	1	1	1	1	1	1	1	1	1	1	1
		max	1	1	3	3	1	2	2	2	1	2	3	4

모형을 소개하였고 다음으로 일반 트리 네트워크가 많아야 n-3개의 가상 노드를 이용함으로써 이분 트리로 전환될 수 있음을 보였다. 또한 Kariv and Hakimi(1979)가 $O(p^2n^2)$ 시간으로 최적값을 구하는 알고리듬을 제시하였고 Tamir(1996)는 전체 트리를 대상으로 각 노드간 거리를 $O(n^2)$시간의 정렬 알고리듬을 이용하여 선행처리과정에서 정렬한 후 이를 순환식에 활용함으로써 $O(pn^2)$시간 동적계획 알고리듬을 개발한 반면 본 연구에서는 일단 순환식을 주어진 트리 네트워크의 완전부분트리 상으로 한정하여 계산하도록 정의한 후 매 연산마다 완전부분트리의 루트를 역깊이우선탐색 순서를 따라 이동하면서 순환식을 적용해 가는 $O(pn^2)$시간 동적계획 알고리듬을 개발하였다. 또한 Kariv and Hakimi(1979)는 $O(n^2)$시간으로 최적해를 구하는 알고리듬을 제시하였고 Tamir(1996)는 최적해를 구하는 효율적인 방법에 대해서는 전혀 언급하지 않은 반면 본 연구에서는 $O(n)$시간으로 최적해를 구하는 알고리듬을 제시하였으며 알고리듬에 대한 효율성 및 계산 실험의 용이성을 검증해 보기 위해 이를 C언어로 코딩하여 임의로 생성된 여러 가지 유형의 문제들에 대한 계산 실험결과를 제시하였다.

결과적으로 본 연구에서 개발된 알고리듬은 이론적인 측면에서 향후 NP-hard 문제로 알려져 있는 일반 네트워크 상에서의 p-미디안 문제(PMPOG)나 공급사슬관리(supply chain management : SCM)에서 발생되는 다양한 형태의 최적화 문제, 예컨대 안전재고 및 생산시설 네트워크 설계 모델과 관련된 최적화문제 등을 푸는 효율적인 알고리듬의 개발에 체계적이고 효율적으로 확장·적용될 수 있을 것으로 기대된다. 또한 실험적인 측면에서도 본 연구에서 개발된 알고리듬의 실행이 매우 용이한 것으로 판단되므로 본 연구결과를 PMPOT와 관련된 다양한 응용문제에 직접적으로 활용함으로써 의사결정자들의 비효율적이고 비합리적인 의사결정으로 인한 막대한 비용과 시간의 낭비를 사전에 방지하는데 일조할 수 있을 것으로 기대된다.

참 고 문 헌

[1] Ahn, S.H., C. Cooper, G. Cornuejols and A. Frieze, "Probabilistic Analysis of a Relaxation for the K-median Problem," *Mathematics of Operations Research*, 13(1988), pp.1-31.

[2] Baldacci, R., E. Hadjiconstantinou, V. Maniezzo and A. Mingozzi, "A New Method for Solving Capacitated Location Problems Based on a Set Partitioning Approach," *Computers and Operations Research*, 29, (2002), pp.365-386.

[3] Boffey, T.B. and J. KarKazis, "P-median and Multi-medians," *Journal of Operations Research Society*, 35(1984), pp.57-64.

[4] Burkard, R.E., E. Cela and H. Dollani, "2-Medians in Trees with Pos/Neg Weights," *Discrete Applied Mathematics*, 105(2000), pp.51-71.

[5] Cavalier, T.M. and H.D. Sherali, "Network Location Problems with Continuous Link Demands : p-medians on a chain and 2-medians on a tree," *European Journal of Operational Research*, 23(1986), pp.246-255.

[6] Christofides, N. and J.E. Beasley, "A Tree Search Algorithm for the p-median Problem," *European Journal of Operational Research*, 10(1982), pp.196-204.

[7] Drezner, Z., "On the Conditional p-median Problem," *Computers and Operations Research*, Vol.22, No.5(1995), pp.525-530.

[8] Ernst, A.T. and Krishnamoorthy, M., "An Exact Solution Approach Based on Shor-

test-Paths for p-Hub Median Problems," *INFORMS Journal on Computing*, Vol. 10, No.2(1988), pp.149-162.

[9] Garey, M.R. and D.S. Johnson, *Computers and Intractability : A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, California, 1979.

[10] Halldorsson, M.M., K. Iwano, N. Katoh, and T. Tokuyama, "Finding Subsets Maximizing Minimum Structures," *SIAM Journal on Discrete Mathematics*, 12(1999), pp. 342-359.

[11] Hribar, M. and M.S. Daskin, "A Dynamic Programming Heuristic for the p-median Problem," *European Journal of Operational Research*, 101(1997), pp.499-508.

[12] Hsu, W.L., "The Distance-Domination Numbers of Trees," *Operations Research Letters*, 1(1982), pp.96-100.

[13] Kariv, O. and S.L. Hakimi, "An Algorithmic Approach to Network Location Problems, Part II : p-medians," *SIAM Journal on Applied Mathematics*, 37(1979), pp.539-560.

[14] Tamir, A., "An $O(pn^2)$ Algorithm for the p-median and Related Problems on Tree Graphs," *Operations Research Letters*, 19 (1996), pp.59-64.