저장능력이 무한대인 장소입지문제에 벤더즈 분해기법과 GAMS의 적용*

이상진**

Solution Method of the Uncapapacitated Facility Location Problem Using GAMS and Benders' Decomposition Approach*

Lee, Sang Jin**

ABSTRACT

The uncapacitated facility location problem considered here is to determine facility location sites, minimizing the total cost of establishing facilities and serving customer demand points which require primary and back-up services. To solve this problem effectively, we propose two things in this study. First, we propose an idea of Benders' decomposition approach as a solution method of the problem. Second, we implement the problem on GAMS. Using GAMS(General Algebraic Modeling System) can utilize an mixed-integer programming solver such as ZOOM/XMP and provide a completely general automated implementation with a proposed solution method.

1. 서 론

본 연구에서 고려하고 있는 장소입지문제는 모든 수요지에서 최우선적인 1차 공급지와 보조적인 2차 공급지를 필수로 하고 있으며 공급지의 공급능력이나 수요지의 저장능력은 무한대로 가정하고 있다. 이와 같은 조건을 충족시키면서 공급지에서의 설비건설비와 공급지에서 부터 수요지까지의 공급비용을 최소화하는 공급지를 결정하려고 한다. 이 문제는 모든 수요지에서 필요한 공급을 중단없이 받아야 하는 경우에 발생할 수 있는데 즉, 평상시에는 1차 공급지로 부터 공급을 받지만 1차 공급지로 부터의 공급에 이상이 생기거나 위급상황이 발생했을 때 공급의 중단없이 2차 공급지로 부터 계속적인 공급을 필요로 하는 경우에 발생하는 문제이다. 예를 들면 구급 차량

* 본 연구는 국방대학원 연구조성비에 의하여 연구되었음
** 국방대학원 조교수

이나 소방서의 배치 계획, 가스 및 배전시설 배치 계획에 이와 같은 상황이 발생할 수 있고 또한 데이타베이스를 활용하고 있는 컴퓨터 네트워크 배치 계획에도 발생할 수 있다. 예를 들어 은행 각 지점의 단말기들은 고객의 정보에 대한 데이타베이스를 1차적으로 공급받는 데이타베이스 공급입지가 필요하며 만약 이 1차 입지로 부터의 커뮤니케이션 네트워크가 문제가 있어 자료를 공급받지 못한다면 곧 바로 보조 2차 데이타베이스 설비로 부터 필요한 자료를 공급받아야 은행의 업무는 진행될 수 있다. 이 문제에서 한 특정 공급지가 같은 수요지에 대하여 1차적이면서 2차적인 공급지로서 역할을 동시에 수행할 수 없다.

이 문제는 2차적인 공급관계를 필수적으로 요구하기 때문에 단순한 '저장 능력이 무한대인 장소 입지문제'보다 문제의 규모가 훨씬 증가하게 되며 수요지가 많을 경우 특별한 형태를 가진 대규모의 정수계획문제로 형성된다. 예를 들어 문제에서 수요지가 200군데, 예상공급입지가 20군데라면 제약조건식들이 4,400개, 의사결정변수가 8,020개인 정수계획문제로 형성된다. 정수계획문제는 문제의 규모가 커질수록 해법의 어려움이 지수적으로 증가하며 또한 일정 규모 이상은 문제해법이 불가능하다. [4, 8]

Pirkul[10]은 이러한 문제의 최적해를 구하기 위해 특별하게 고안된 휴리스틱 규칙을 제안하였다. 그의 휴리스틱 기법은 라그란지언 완화기법을 활용하는 것으로 기존의 알고리듬보다 효율적으로 최적해를 구할 수 있었다. 본 연구는 해법으로 휴리스틱과 같은 새로운 규칙을 제안하는 것이 아니라 문제의 활용성을 제고하려는 의도로 시작되었는데 즉, 기존의 상업화된 선형 혹은 정수계획패키지를 활용하여 문제를 용이하게 풀려는 목적을 가지고 있다. 이 목적을 수행하기 위해 먼저 벤더즈 분해기법[3, 6, 7]의 아이디어를 적용한 새로운 알고리듬을 도출해내고자 한다. 벤더즈 분해기법을 적용할 때 대규모의 문제는 주문제(master problem)와 수요지 만큼의 하위문제들(sub-problems)로 분해할 수 있다. 다음으로 분해된 문제들의 최적해를 구하는데 GAMS(General Algebraic Modeling System)라는 순서적 모형언어를 사용하면 문제의 활용성을 제고할 수 있다. 분해문제의 전체 최적해를 구하기 위해서는 주문제와 하위문제들의 부문 최적해를 반복적(iteratively)으로 구해야 하는데 반복과정에 필요한 모든 절차들을 GAMS가 수행한다. GAMS를 활용하지 않는다면 주문제와 하위문제들의 부문 최적해를 구하기 위해서 이용자 자신이 새로운 코드를 만들어 써야 하거나 혹은 기존의 선형 혹은 정수계획 패키지(예를 들면 MINOS, ZOOM, IMSL)를 이용하더라도 반복과정에 필요한 자료의 이동을 위해서 FORTRAN이나 C 등의 프로그램 언어로 코드를 만들어야 한다. GAMS를 활용할 경우, 주문제와 하위문제들 각각의 최적해를 구하기 위한 패키지와 자료의 이동 및 통제를 위한 모든 절차를 GAMS가 제공할 수 있다. 반복적인 알고리듬을 GAMS가 수행하므로 GAMS의 기본적인 문법규칙만 이해하게 되면 문제를 용이하게 풀 수 있다.

이러한 GAMS 실행 아이디어는 다른 분해해법이나 특수문제들에 적용할 수 있다 예를 들어, 확률적 선형계획문제(Stochastic LP Problems)[2]나 절대적 우선순위 목표계획법(preemptive priority GP problem)[1] 등의 최적해를 구하기 위해서 분해해법이나 LP문제를 반복적으로 해결하는 것이 필요한데 이러한 문제들의 해법에 GAMS 실행 아이디어를 직접적으로 활용할 수 있다.

2. 수학적 모형과 해법

2.1 수학적 모형

1차 및 2차 공급지를 필요로하는 저장능력이 무한대인 장소입지문제는 다음과 같이 수학적으로 형성할 수 있다.

$$\min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}+\sum_{i\in I}\sum_{j\in J} \bar{c}_{ij}z_{ij} + \sum_{j\in J} v_j y_j$$

s.t. (1)

$$\sum_{j\in J} x_{ij}=1 \quad \forall i\in I$$

$$\sum_{j\in J} z_{ij}=1 \quad \forall i\in I$$

$$x_{ij}+z_{ij}-y_j\le 0 \quad \forall i\in I,\ j\in J$$

$$x_{ij},\ z_{ij},\ y_j\in \{0,\ 1\} \quad \forall i\in I,\ j\in J$$

여기서 i(i=1,…,m)와 j(j=1,…,n)는 각각 수요지와 예상공급입지를 의미한다. c_{ij}와 $\bar{c}_{ij}$는 각각 최우선적 그리고 보조적인 공급입지 j로부터 수요지 i까지 공급할 때의 변동비 즉, 생산 및 공급비용의 합계를 의미한다. v_j의 값은 예상공급입지 j에 공장을 건설할 때의 비용이다. 예상공급입지 j가 수요지 i에 대하여 최우선적 공급예상입지로 선정된다면 할당변수 x_{ij}의 값은 1이 될 것이고 그렇지 못하다면 이 값은 0이 될 것이다. 이와 같은 논리는 2차적 할당변수 z_{ij}의 값에도 똑 같이 적용된다. 2차적 공급입지를 필수로하는 제약조건과 이에 따르는 할당변수들 때문에 문제가 특별한 모양을 가진 모형이 된다.

예상공급입지변수 y_j의 값은 공급지가 j에 위치하게 될 때 1이 될 것이고 그렇지 못하다면 값은 0이 될 것이다. 첫번째와 두번째 제약조건식들은 모든 수요지 i가 각각 최우선적 그리고 보조적 예상공급입지를 하나씩은 가져야 한다는 조건이다. 세번째 제약조건식은 같은 예상공급입지 j가 한 수요지에 대하여 최우선적 그리고 보조적 공급입지기능을 동시에 수행할 수 없다는 것이다. 네번째 제약 조건식은 정수조건이다.

2.2. 알고리듬

특수한 형태를 가진 문제를 좀 더 효율적으로 풀기 위해서 혹은 기존의 패키지로는 풀 수 없는 대규모의 문제를 풀기 위해서 벤더즈 분해기법이 많이 활용되고 있다. 우리들은 벤더즈 분해기법의 아이디어를 '저장능력이 무한대인 장소입지문제'의 해법에 적용할 것이다.

벤더즈 분해기법의 아이디어를 요약하면 다음과 같다. 먼저 어떤 특정한 변수를 특정한 값으로 고정하여 이 고정된 값으로 하위문제의 우변상수를 수정한다. 이 하위문제를 풀어 쌍대값을 구한다. 이 쌍대값을 이용하여 새로운 제약조건식을 만들어 주문제에 첨가한다. 이 첨가된 주문제를 풀어 이전의 고정된 값보다 더 나은 값을 얻는다. 새로이 고정된 값으로 하위문제의 우변상수를 수정한다. 이와 같은 절차를 알고리듬의 종료조건을 충족할 때까지 계속 반복한다.

위의 아이디어를 우리 문제에 적용해보자. 먼저, 어떤 변수를 고정하는 것이 보다 효과적인가를 고려해 보자. 공급입지변수 y_j를 고정하는 것이 할당변수 x_{ij}와 z_{ij}를 고정하는 것보다 문제 규모를 줄이는데 있어 훨씬 효과적이다. 왜냐하면 공급입지변수가 0으로 고정될 때마다 이에 상응하는 할당변수들은 자연적으로 0으로 고정되기 때문에 이 할당변수들은 고려하지 않아도 된다. 반대로 할당변수 중의 하나를 0으로 고정하게 되더라도 이에 상응하는 공급입지변수가 0이 되지 않을 수도 있기 때문에 문제를 줄일 수 있는 규모가 상대적으로 작아지게 된다.

공급입지변수 y_j^*를 고정하면 문제 (1)은 다음의 하위문제 (2)로 유도할 수 있다.

$$\min \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}+\sum_{i\in I}\sum_{j\in J} \bar{c}_{ij}z_{ij} + \sum_{j\in J} v_j y_j$$

s.t. (2)

$$\sum_{j\in J} x_{ij}=1 \quad \forall i\in I$$
$$\sum_{j\in J} z_{ij}=1 \quad \forall i\in I$$
$$x_{ij}+z_{ij}\le y_j^* \quad \forall i\in I,\ j\in J$$
$$x_{ij},\ z_{ij}\in\{0,\ 1\} \quad \forall i\in I,\ j\in J$$

여기서 정수할당변수 x_{ij}와 z_{ij}는 문제의 형성에 영향을 주지 않고 하한한계값이 0이고 상한한계값이 1인 양의 실수값으로 변환할 수 있다. 왜냐하면 정수조건을 완화하여도 문제의 최적해는 자동적으로 정수값이 될 것이기 때문이다.

$$0\le x_{ij}\le 1,\ 0\le z_{ij}\le 1,\ \forall i\in I,\ j\in J \qquad (3)$$

문제 (2)의 네번째 정수조건식 대신에 완화된 식 (3)을 사용하면 문제 (2)는 선형계획문제가 된다. 이 연구에서는 정수제약조건이 완화된 선형계획문제를 하위문제로 사용할 것이다. 하위문제는 수요지 m개 만큼의 또 다른 하위문제들로 분리할 수 있다. 왜냐하면 문제 (2)에서 각각의 수요지별 하위문제들은 서로 독립적으로 존재하고 있기 때문에 문제 (2)는 분해가 가능하다.

매 반복시 하위문제에서 최소한 2개 이상의 공급입지변수의 값이 1로 고정된다면 (이 말은 2개 이상의 입지가 공급지로 선정되어 건설된다는 의미이다) 하위문제는 항상 실행가능하다. 문제 (1)에서는 첫번째, 두번째, 그리고 세번째 제약조건식들이 최소한 두개 이상의 공급입지변수의 값들을 1로 고정하도록 하는 역할을 내재적으로 수행하였기에 문제 (1)은 언제나 실행가능이었다. 그런데 새로운 알고리듬을 유도해내는 과정에서 세번째 제약조건식이 변화하게 되고 이 결과 최소한 2개 이상의 공급입지변수의 값들을 1로 고정하는 역할을 더 이상 수행하지 못하게 되었다. 그래서 하위문제가 늘 실행가능하기 위해서는 앞으로 유도될 주문제에 공급입지변수 중 최소한 2개 이상의 값을 1로 고정시키는 명백한 제약조건식을 추가해야만 한다. 즉, 공급입지변수는 다음의 제약조건집합을 충족시켜야 한다.

$$R=\{y_i\,|\,\sum_{j\in J} y_j\ge 2,\ y_j\in\{0,\ 1\},\quad \forall j\in J\} \qquad (4)$$

주문제를 유도하기 위해 하위문제를 다음과 같은 쌍대문제로 변환하여보자.

$$\max \sum_{i\in I}(\lambda_i)+\sum_{i\in I}(\alpha_i)+\sum_{i\in I}\sum_{j\in J}(\mu_{ij}y_j^*)$$

s.t. (5)

$$\lambda_i+\mu_{ij}\le c_{ij} \quad \forall i,\ j$$
$$\alpha_i+\mu_{ij}\le \bar{c}_{ij} \quad \forall i,\ j$$
$$\mu_{ij}\le 0,\ \lambda_i\text{와 }\alpha_i\text{는 부호 제한 없음}$$

주문제에서의 제약조건집합 (4)와 하위문제에서의 제약조건식 (3)에 의해 하위문제는 언제나 한계를 가진 실행가능영역을 갖게 된다. 하위문제가 이와 같은 조건을 갖게 되면 쌍대정리에 의해 하위문제의 쌍대문제도 한계를 가진 실행가능영역을 갖게 된다. 그래서 하위문제가 최적값을 갖게 되면 LP의 강쌍대정리에 의하여 쌍대문제 (5)도 같은 최적값을 갖게 된다. 쌍대정리에 대한 논의를 통해서 문제 (1)은 다음 문제 (6)과 완전히 동등한 문제로 재형성할 수 있다.

$$\min_{y\in R}\{\sum_{j\in J} v_j y_j+\max\{\sum_{i\in I}(\lambda_i)+\sum_{i\in I}(\alpha_i)$$
$$+\sum_{i\in I}\sum_{j\in J}(\mu_{ij}y_{ij}^*)\,|\,\lambda_i+\mu_{ij}\le c_{ij},$$
$$\alpha_i+\mu_{ij}\le \bar{c}_{ij},\ \mu_{ij}\le 0,\quad \forall i,\ j\} \qquad (6)$$

문제 (6)의 내부 최대화 문제의 제약조건집합을 고려해보자.

$$P=\{(\lambda_i,\ \alpha_i,\ \mu_{ij})\,|\,\lambda_i+\mu_{ij}\le c_{ij},$$
$$\alpha_i+\mu_{ij}\le \bar{c}_{ij},\ \mu_{ij}\le 0,\quad \forall i,\ j\} \qquad (7)$$

앞서 지적한바와 같이 하위문제가 한계를 가진 실행가능영역을 가졌기 때문에 쌍대문제도 한계를 가진 실행가능영역을 가진다. 즉, 집합 P는 유한개의 꼭지점(extreme points: 정점)을 가진 집합이 되며 이 꼭지점들은 $(\lambda_i^k, \alpha_i^k, \mu_{ij}^k)$ k=1,…,K 으로 표현할 수 있다. 이 성질을 이용하여 문제 (6)은 다음 문제 (8)과 같이 표현할 수 있다.

$$\min_{y\in R}\{\sum_{j\in J}v_j y_j+\max_{1\le k\le K}\{\sum_{i\in I}(\lambda_i^k)+\sum_{i\in I}(\alpha_i^k) +\sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j^*)\} \quad (8)$$

문제 (8)은 다음 문제 (P^k)와 동등하다.

$$\begin{aligned}&\min\ y_0\\ &\text{s.t.} \quad (P^k)\\ &\quad y_0\ge\sum_{j\in J}v_j y_j+\sum_{i\in I}(\lambda_i^k)+\sum_{i\in I}(\alpha_i^k)\\ &\qquad +\sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j),\ k=1,\cdots,K,\\ &\quad \sum_{j\in J}y_j\ge 2,\\ &\quad y_j\in\{0,1\},\quad \forall i\in I,\ j\in J\end{aligned}$$

문제 (P^k)는 "완전 주문제(full master problem)"라 부른다. 문제 (P^k)를 풀 때의 최적해는 문제 (1)을 풀 때의 최적해와 같다. 그러나, 집합 P에서 유한개의 꼭지점을 한꺼번에 모두 알아내기가 어려우며 특히 집합 P의 꼭지점이 아주 많다면 한꺼번에 이 모두를 도출해 내는 것은 거의 불가능하다. 즉, 문제 (P^k)의 첫번째 제약식($y_0\ge\sum_{j\in J}v_j y_j+\sum_{i\in I}(\lambda_i^k)+\sum_{i\in I}(\alpha_i^k)+\sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j)$, k=1,…,K,)들이 처음부터 명확하게 모두 나타나지 않기 때문에 알고리듬을 반복할 때마다 제약조건식을 하나씩 유도해 내어야만 한다. 벤더즈 분해기법에 의해 유도된 알고리듬의 흐름은 대략 다음과 같다.

단계 1 : 먼저 장소입지변수의 값 중 2개 이상을 1로 고정한다.

단계 2 : 결정된 장소입지변수 값으로 하위문제의 우변상수를 수정하여 하위문제를 푼다. 해결된 하위문제들의 쌍대값들을 이용하여 제약조건식을 하나 유도한다.

단계 3 : 유도된 제약조건식을 주문제에 첨가한다. 첨가된 주문제를 풀어 이전보다 나은 장소입지변수를 결정한다.

단계 4 : 종료조건을 충족시켰는지 시험하고 충족되지 않았다면 다시 2단계로 되돌아 간다. 충족되었다면 최적해를 구한다.

이 알고리듬은 종료조건을 충족할 때까지 반복되는데 종료조건은 상한한계값(UB)이 하한한계값(LB)과 ε(Epsilon)을 합한 값보다 작거나 같으면 종료조건을 충족시키게 된다. 이 알고리듬의 매력 중의 하나는 매 반복시 주문제와 하위문제의 목적함수값으로 부터 상한한계값과 하한한계값들을 용이하게 구할 수 있고 또한 상한한계값과 하한한계값들이 최적 목적함수값에 수렴한다는 것이다.

이제 하한한계값과 상한한계값을 어떻게 구하는지 살펴보자. 먼저 하한한계값이 결정되는 과정을 살펴보자. $(\lambda_i^0, \alpha_i^0, \mu_{ij}^0)$을 집합 P의 실행가능영역안에 있는 한 점으로 정의하자. 이 점이 꼭 꼭지점이어야 하는 것은 아니며 실행가능 영역내의 어느 점이라도 가능하다. 그렇다면 쌍대문제 (5)로 부터 다음과 같은 결과를 유도할 수 있다.

$$\begin{aligned}&(\sum_{i\in I}(\lambda_i^0)+\sum_{i\in I}(\alpha_i^0)+\sum_{i\in I}\sum_{j\in J}(\mu_{ij}^0 y_j^*))\\ &\quad\le \max\{\sum_{i\in I}(\lambda_i^k)+\sum_{i\in I}(\alpha_i^k)\\ &\qquad +\sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j^*),\quad \forall k\}\end{aligned} \quad (9)$$

이 결과를 이용하면 제약조건식 $(y_0 \geq \sum_{j\in J} v_j y_j + \sum_{i\in I} (\lambda_i^k) + \sum_{i\in I}(\alpha_i^k) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j))$에서 y_0의 값이 문제 (P^1)의 제약조건식 $(y_0 \geq \sum_{j\in J} v_j y_j + \sum_{i\in I}(\lambda_i^1) + \sum_{i\in I}(\alpha_i^1) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^1 y_j)$에서의 y_0의 값보다 작거나 같다는 것을 알 수 있다. 이제 문제 (P^1)의 최적해를 $(y_0^1, y_j^1, \forall j)$라고 하자. 그러면 이에 상응한 하한한계값은 $LB=y_0^1$가 되는데 왜냐하면 문제 (P^1)의 제약조건식들은 문제 (P^k)의 제약조건식들의 부분집합이기 때문이다. 즉, 매 반복시 하한한계값은 첨가된 주문제를 풀 때의 최적값과 같다.

상한한계값을 구하기 위해 공급입지변수 y_j는 고정하고 하위문제를 풀어 최적 할당변수를 결정하자. 공급입지변수 y_j를 y_j^1로 고정하면 문제 (1)의 해공간(solution space)이 제한되는 것을 제외하고는 하위문제는 문제 (1)과 동일하기 때문에 하위문제에 대한 최적값과 입지를 건설하는 비용의 합계는 필요로 하는 상한한계값이 된다. 즉, 상한한계값 $UB=\sum_{j\in J} v_j y_j^1 + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}^1 + \sum_{i\in I}\sum_{j\in J}\bar{c}_{ij}z_{ij}^1$이 된다. 그런데, 하위문제가 최적값을 가지면 이의 쌍대문제 (5)도 똑같은 최적값을 갖게 되므로 $UB=\sum_{j\in J} v_j y_j^1 + (\sum_{i\in I}(\lambda_i^1) + \sum_{i\in I}(\alpha_{ij}^1) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^1 y_j^1))$가 된다. 즉, 매 반복시 하위문제의 최적값이 상한한계값을 구하는데 이용된다.

집합 P가 유한개의 꼭지점만을 가졌다는 사실 때문에 이 알고리듬의 수렴을 보장할 수 있지만 이 과정에서 꼭지점들을 반복하여 구하지 않아야 한다는 사실이 증명되어야 한다. 이것은 다음과 같이 증명된다.

(증명) 여기서 집합 P의 두개의 꼭지점 $(\lambda_i^k, \alpha_i^k, \mu_{ij}^k)$과 $(\lambda_i^{k-1}, \alpha_i^{k-1}, \mu_{ij}^{k-1})$을 고려해 보자. 이 점들은 k−1번째와 k번째 반복에서 연속적으로 구할 수 있는 꼭지점들이다. 앞에서 보여준 바와 같이 $LB=y_0^k=\sum_{j\in J} v_j y_j^k + \sum_{i\in I}(\lambda_i^{k-1}) + \sum_{i\in I}(\alpha_i^{k-1}) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^{k-1} y_j^k) \leq y_0^*$를 유도할 수 있고 하위문제와 쌍대문제 (5)로 부터 다음을 유도할 수 있다.

$$\begin{aligned} UB &= \sum_{j\in J} v_j y_j^k + \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij}^k + \sum_{i\in I}\sum_{j\in J}\bar{c}_{ij}z_{ij}^k \\ &= \sum_{j\in J} v_j y_j^k + \sum_{i\in I}(\lambda_i^k) + \sum_{i\in I}(\alpha_i^k) \\ &\quad + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j^k) \geq y_0^* \end{aligned}$$

이 k번째 꼭지점이 최적해가 아니라면 등식 $LB<y_0^*<UB$이 성립하며 이것을 이용하여 다음을 유도할 수 있다.

$$\begin{aligned} &\sum_{i\in I}(\lambda_i^k) + \sum_{i\in I}(\alpha_i^k) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^k y_j^k) > \\ &\quad \sum_{i\in I}(\lambda_i^{k-1}) + \sum_{i\in I}(\alpha_i^{k-1}) + \sum_{i\in I}\sum_{j\in J}(\mu_{ij}^{k-1} y_j^k) \end{aligned}$$

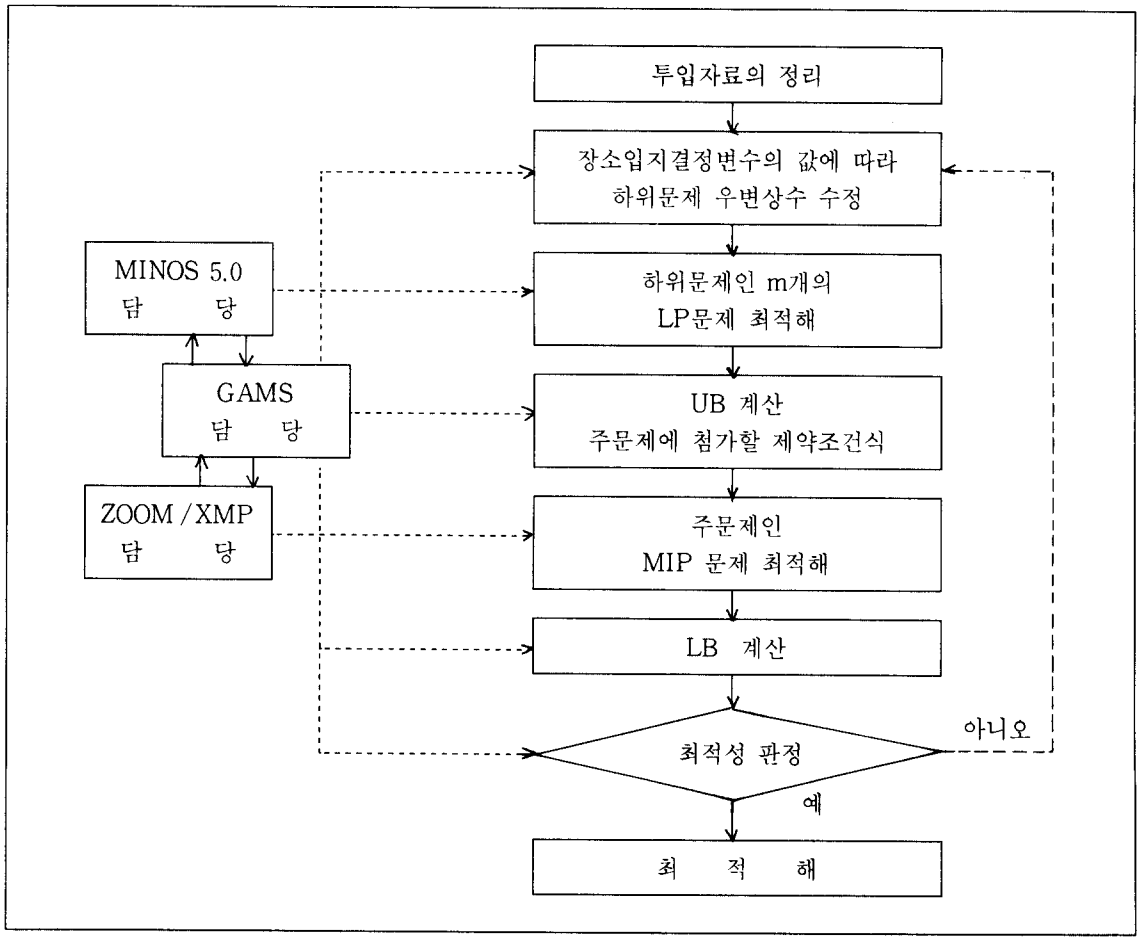
위의 식은 꼭지점 $(\lambda_i^k, \alpha_i^k, \mu_{ij}^k)$에서의 쌍대문제 (5)의 최적값이 꼭지점 $(\lambda_i^{k-1}, \alpha_i^{k-1}, \mu_{ij}^{k-1})$에서의 최적값보다 크다는 것을 보여준다. 그런 까닭에 꼭지점 $(\lambda_i^k, \alpha_i^k, \mu_{ij}^k)$는 $(\lambda_i^{k-1}, \alpha_i^{k-1}, \mu_{ij}^{k-1})$와 항상 달라야만 한다. □

이 문제의 알고리듬은 〈그림 1〉에 나타나 있다. 그림 우편에는 알고리듬의 흐름도가 나타나 있고 좌편은 알고리듬과 GAMS와의 관계를 표시하고 있다. 알고리듬의 처음 단계는 먼저 투입자료를 정리하는 것이다. 색인(index)값을 정리하고 예상공급입지변수 중에서 몇개의 변수를 선택하여 값을 1로 고정해야 한다. 여기서 몇개의 변수값을 선택하며 또 어떤 변수를 고정해야 할 것인지에 대한 기존의 연구가 없기 때문에 어떤 것을 몇개 선택할 것인지가 문제가 된다. 변수값을 많이 0으로 고정하면 할수록 이에 상응하는 할당변수들의 값이 0이 되기 때문에 하위문제의 크기가 많이 줄어들 것이다. 대략의 선택기준은 입지를 건설하는 비용(v_j)이 우선적이며 보조적인 공급비용(c_{ij}

와 $\bar{c}_{ij}$)보다 상대적으로 아주 높다면 예상공급입지를 많이 선택하는 것이 바람직하며 반대일 경우에는 예상공급입지를 작게 선택해도 되겠다는 것이다.

다음 2단계는 하위문제의 우변상수값을 수정하고 최적해를 구하는 것이며 또한 상한한계값과 주문제에 첨가될 제약조건식을 구하는 것이다. 3단계는 주문제의 최적해를 구하고 하한한계값을 계산한다. 4단계는 하한한계값과 상한한계값을 비교하여 종료조건이 충족되지 않으면(즉, UB≥LB+ε) 다시 2단계로 되돌아가지만 충족되면 최적해를 구한다. 이 알고리듬에서 1회 반복(iteration)은 2단계부터 4단계까지 진행하며 상응한 하위문제와 주문제의 부문최적해를 구하는 것이다.

종료조건이 충족되면 주문제에서 공급입지변수의 전체 최적해가 결정되지만 이 때 하위문제에서 결정되는 할당변수값들은 전체문제의 최적해가 아니다. 이 때의 할당변수값들은 이전 반복단계의 공급입지변수값에 따라 수정된 하위문제에서 결정된 값이기 때문에 최적할당변수는 현 반복단계에서 결정된 공급입지변수값에 따라 수정된 하위문제의 최적해에서 구해야 한다. 즉, 이 알고리듬에서 전체 최적해를 구하기 위해서는 하위문제를 주문제보다 한번 더 구해야 한다.



〈그림 1〉 입지문제의 흐름도

3. GAMS의 실행

저장능력이 무한대인 장소입지결정문제는 벤더즈 분해기법에 의하여 혼합정수계획문제인 주문제와 선형계획문제인 m개의 하위문제들로 분해된다. 이 문제는 주문제와 하위문제들의 최적해를 구하기 위해 패키지들을 반복적으로 사용하며 반복시마다 주문제의 규모와 크기가 변하기 때문에 동태적이다. 그래서 이 문제에서 GAMS의 문법규칙(syntax)은 다른 정태적인 문제의 GAMS 문법규칙과는 다른 특성들을 요구하고 있다. 이 연구에서는 일반적인 GAMS의 문법규칙을 모두 설명하지 않고 동태적인 문제의 해결을 위한 GAMS의 문법규칙 즉, 동태적인 특성들만을 설명하겠다. 또한 Paules와 Floudas[9]가 이러한 동태적 문제들의 알고리듬 개발 절차에 대해 언급하고 있기 때문에 여기서는 [9]에서 제시한 것과 다른 부분들만을 중점적으로 설명하겠다.

이 문제의 전체 최적해를 구하기 위한 알고리듬과 GAMS의 관계는 흐름도 〈그림 1〉에서 보여주고 있다. 〈그림 1〉에서 보여주는 바와 같이 GAMS는 LP나 MIP를 풀기 위한 패키지를 통제하며 분해된 문제들의 우변상수식을 수정하거나 제약조건식을 첨가하기 위해 필요한 자료의 이동을 담당하게 된다.

여기서 먼저 하위문제와 주문제들을 풀기 위한 패키지를 살펴보고 다음 이들 하위문제와 주문제들 사이에 자료의 이동을 위해 필요한 GAMS에서의 문법규칙에 대해서 살펴보자.

〈그림 1〉의 흐름도에서 보여주는 바와 같이 하위문제들의 최적해를 구하기 위해서 GAMS에서 현재 이용가능한 LP 패키지인 MINOS 5.0이나 BDMLP 두가지 중 하나를 이용할 수 있다. 주문제인 혼합정수계획문제의 최적해를 구하기 위해서 MIP문제 패키지인 ZOOM/XMP를 이용할 수 있다.

먼저 하위문제가 어떻게 GAMS에서 형성되는지를 살펴보자. 하위문제는 수요지 m개로 분해되며 분해된 하위문제 각각은 크게 3가지 종류의 제약조건식들이 필요하다. 각 수요지는 최우선적이며 보조적인 공급입지를 필요로 하며 GAMS 모형에서는 이들의 이름을 각각 PRIFAC, SECFAC 이라고 표시한다. 나머지 한 종류의 제약조건식은 한 공급지가 같은 수요지에 대하여 최우선적이며 보조적인 공급지 역할을 동시에 수행할 수 없다는 것으로 이름을 NOBOTH라 표시한다.

```
SUBOBJ.. SUM((J), CP(J)*X(J))+ SUM
   ((J), CS(J)*Z(J))=E=SUBOBJCOST
PRIFAC.. SUM(J, X(J))=E=1
SECFAC.. SUM(J, Z(J))=E=1
NOBOTH(J).. X(J)+Z(J)=L=YVAL(J)
```

하위문제를 GAMS 내에서 모델로 형성하기 위해서 다음과 같은 문법규칙이 필요하다.

```
MODEL SUBPROBLEM /SUBOBJ,
   PRIFAC, SECFAC, NOBOTH /
```

주문제를 형성하기 위해서 매 반복시마다 제약조건식이 하나씩 추가되어야 한다. 이 제약조건식은 하위문제에서 구한 쌍대값을 이용하여 구할 수 있는데 위의 하위문제 제약조건식 각각에 상응하는 쌍대값을 MARG1, MARG2, MARG3라고 할 때 첨가할 제약조건식은 아래의 CONSTS에 표현되고 주문제는 다음과 같이 도출해 낼 수 있다.

```
MASOBJ.. SUM(J, V(J)*Y(J))
   +YNOT =E= MASOBJCOST
CONSTS(K).. SUM(I, MARG1(I, K)
```

```
+MARG2(I,K))+SUM((I, J),
MARG3(I, J, K)*Y(J))=L=YNOT
MINREQ.. SUM(J, Y(J))=G=2
```

주문제를 GAMS 내에서 모델로 형성하기 위해 다음이 필요하다.

```
MODEL MASTER /MASOBJ, CONSTS,
MINREQ/
```

이제 주문제와 하위문제 사이에 필요한 자료의 이동과 통제에 대한 문법규칙들을 살펴보자. 알고리듬에서 하위문제들은 매번 반복할 때마다 문제의 규모나 구조가 변화되지 않지만 정수변수인 장소입지변수의 값에 따라 이에 상응하는 제약조건식(NOBOTH)들의 우변상수값이 변화한다. 주문제의 경우에는 매번 반복될 때마다 새로운 제약조건식(CONSTS)이 추가됨에 따라 모형이 변화하게 된다. 알고리듬이 실행되고 있는 동안 GAMS는 연이은 주문제들에 제약조건식을 하나씩 추가해야 한다. 이러한 동태적 모형을 구성하는데 중요한 것이 동태적 집합(dynamic sets)의 사용이다.

이를 위해서 GAMS 첫부분에서 4개의 추가적인 집합을 지정해야만 한다. 먼저, 하위문제들과 주문제의 최적해를 각각 구해야하는 단계를 반복해야하는데 이 반복의 수를 표시하는 KM이라는 정태적 집합을 지정해야한다. 이 집합은 알고리듬이 허용하는 최대 반복횟수(maximum iteration)만큼의 원소(elements)를 포함하고 있다. 나머지 3개는 동태적 집합인데 각각 K(KM), KITER(KM), KLOOPCTL(KM)이다. K(KM)은 KM의 동태적 부분집합으로 알고리듬이 진행될 때 실행되고 있는 반복의 수와 동일한 수의 원소를 포함하고 있다. 반복이 진행될 때마다 K(KM)은 그 수가 증가하여 반복의 수만큼 되며 이 특성 때문에 주문제 제약조건식의 형성에 도움이 되는 집합이다. KITER(KM)은 KM의 동태적 부분집합으로 알고리듬이 진행될 때 현재의 반복과 상응한 오직 한 요소만을 포함할 뿐이다. KM에서의 서수가 현재 반복의 수와 일치할 때 한 요소만을 포함하기 때문에 하위문제에서의 우변상수 수정에 도움이 되는 집합이다. KLOOPCTL(KM)은 KM의 동태적 부분집합으로 반복과정을 전체적으로 통제하는 집합이다.

동태적 집합들인 K(KM)과 KITER(KM)은 매 반복에서 최신의 것으로 새롭게 해야한다. KITER(KM)을 새롭게 갱신하기 위해 KITER(KM)의 현재 원소를 다른 곳으로 이동시켜야 한다. 이를 수행하기 위해 반복과정중에 다음 문법규칙이 필요하다.

```
KITER(KM)=YES$(ORD(KM) EQ ORD
   (KLOOP))
```

여기서 KLOOP은 KM의 별명이다. 매 반복시마다 주문제의 제약조건식이 한개씩 추가되기 때문에 K(KM)에는 한 원소씩 추가된다. 이를 수행하기 위해 다음 문법규칙이 필요하다.

```
K(KLOOP) = YES
```

하위문제의 우변상수값을 수정하는 절차에 대해 살펴보자. 하위문제는 매 반복시 우변상수값만 변하고 문제의 규모나 크기는 같기 때문에 문제 자체는 정태적이다. 주문제에서 장소입지결정변수인 정수변수값 Y(J)을 가져와 하위문제의 우변상수값을 변화시켜야 하기 때문에 이 정수변수값을 저장해 놓아야 한다.

```
YVAL(J) = Y.L(J)
YK(J ,KITER) = YVAL(J)
```

여기서 L 확장자는 최적해값을 의미한다. 주문제는 매 반복시마다 새로운 제약조건식들이 하나씩 추가되기 때문에 하위문제에서 쌍대값을 가져와야 한다. 이들 쌍대값을 다음과 같이 저장한다.

MARG1(I, KITER)=PRIFAC.M(I)
MARG2(I, KITER)=SECFAC.M(I)
MARG3(I, J, KITER)=NOBOTH.M(I, J)

여기서 M 확장자는 쌍대값을 의미한다. 종료조건의 충족성을 검사하기 위해서 상한한계값(UPPER)과 하한한계값(LOWER)을 계산해야 한다. 하한한계값은 주문제의 최적값이 되며 하한한계값은 여기서 하위문제의 최적값이 된다. 하위문제의 최적값을 구하고 난 후에 상한한계값을 즉시 저장하고 주문제의 최적값을 구하고 난 후에 하한한계값을 구하여 즉시 저장해야한다.

UPPER=SUBOBJCOST.L + SUM(J, V(J)*
Y(J))
LOWER=MASOBJCOST.L

반복을 위한 문법규칙은 아래와 같이 표현되는데 KLOOPCTL(KM)의 값이 'YES'일 때는 이 알고리듬이 계속 반복하게 된다. 종료조건을 충족시킬 때 즉, 상한한계값에서 하한한계값을 뺀 값이 ε보다 작을 때는 종료조건을 충족시키게 되어 KLOOPCTL(KM)의 값이 'NO'가 되며 알고리듬은 종료된다. 이 문법 규칙은 다음과 같이 표현된다.

KLOOPCTL(KM)=NO$((UPPER-LOWER)
LE EPSILON)

4. 실험 결과

4.1. 실험 문제

이 실험에 사용된 문제들은 가상적인 문제들로 Pirkul이 사용한 휴리스틱 규칙과의 성능을 비교하기 위해 Pirkul이 사용한 실험문제와 같은 방식으로 생성되었다. 각각의 수요지와 예상공급입지의 좌표는 길이가 50인 정사각형에서 일양분포(Unifrom Distribution)를 가정하고 무작위적으로 추출되었다. 그리고 c_{ij}와 $\bar{c}_{ij}$의 값은 수요지와 공급지 사이의 직선거리를 계산하여 그 거리에 상수 25와 15를 각각 곱하였다. 설치비용 v_i은 상한값(u)과 하한값(l) 사이의 일양분포에서 추출하였다. 설치비용의 상한값과 하한값을 조절하여 여러 문제를 생성함으로 할당비용에 대한 설치비용과의 관계를 살펴보려고 하였다.

〈표 1〉은 실험대상 문제들의 크기에 대해 설명한 것이다. 전체문제는 문제를 분해하기 전의 문제이고 하위문제는 분해한 이후의 문제이다. 분해 이후의 주문제는 매 반복시마다 문제의 크기가 달라지므로 여기에서는 언급하지 않았다. 문제크기에서 첫번째 숫자는 제약조건식의 갯수이고 두번째는 의사결정변수의 수이다. 전체문제와 분해문제를 비교해 보면 분해문제의 크기와 문제 중의 비음의 갯수가 작아지는 것을 관찰할 수 있다. Pirkul은 위 표에서 보여준 6개 문제집단에 대하여 설치비용을 달리한 3가지 문제집단별로 다시 구분하여 총 18개 문제집단에 대해 실험을 하였는데 본 연구에서는 12개 문제집단에 대해서 실험을 하였다. 이 실험은 GAMS 386/486 PC용 Version 2.25x을 사용하였고 LP와 MIP 패키지로는 ZOOM을 이용하였다. 실험문제들은 486PC(50MHz, 8 Mega RAM)에서 실행되었으며 ε의 값은 1로 고정한 후에 실험을 한 결과는 다음 〈표 2〉와 같다.

〈표 1〉 실험문제들의 특성

(공급지 수, 수요지 수)	전 체 문 제		분해된 하위문제		
	문 제 크 기	비 음 갯 수	하위문제수	문 제 크 기	비 음 갯 수
(10, 50)	600×1200	3511	50	12×20	61
(10, 100)	1200×2010	7011	100	12×20	61
(10, 200)	2400×4010	14011	200	12×20	61
(20, 50)	1100×2020	7021	50	22×40	121
(20, 100)	220×4020	14021	100	22×40	121
(20, 200)	4400×8020	28021	200	22×40	121

〈표 2〉 GAMS 실행결과

(하한값, 상한값)	공급지	수요지	Pirkul의 실험		전체문제		분해문제	
			시 간	반복횟수	시 간	반복횟수	시 간	반복횟수
(1000, 1500)	10	50	0.97	38	8.95	690	28.69	2
		100	2.12	43	28.07	1277	56.61	2
		200	4.56	45	140.05	3317	111.90	2
	20	50	3.58	82	32.39	1435	163.36	10
		100	6.20	72	155.38	3591	387.64	12
		200	11.34	64	4549.48	35187	641.01	10
(8000, 10000)	10	50	1.22	48	19.78	1191	28.71	2
		100	4.30	93	93.48	3295	114.14	4
		200	5.84	61	485.34	8399	168.92	3
	20	50	4.14	105	122.77	3896	196.26	12
		100	11.66	152	1018.26	15513	227.05	7
		200	27.53	170	−	−	578.56	9

1. 시간은 각 기계의 CPU 시간(초)이다.
2. 전체문제중의 −표시는 GAMS의 LP패키지를 가지고 이 문제를 최대반복 허용회수(50,000번)안에 풀지 못한 경우이다.

여기서 Pirkul의 실험결과는 그 논문에서 직접 인용한 것이기 때문에 다른 두 실험 결과와 비교하기는 어려울 것이다. Pirkul의 실험은 FORTRAN IV를 이용하여 코드를 만들었고 Prime 9955 미니컴퓨터에서 실행시킨 결과이며 그 결과 중 가장 우수한 결과만을 골라서 발췌하였다. 그러나 Pirkul 결과와 분해문제의 결과가 여러 면에서 유사함을 보여 주는데 공급지를 일정하게 두고 수요지가 증가할수록 반복횟수와 시간이 늘어나는 것을 보여주고 있고, 역으로 수요지를 일정하게 두고 공급지가 2배로 증가해도 같은 결과가 발생함을 관찰할 수 있다. 또한 설치비용이 상대적으로 크면 클수록 반복회수와 시간이 증가함을 보여주고 있다.

전체문제와 분해문제를 비교해보면 문제의 규모가 작으면 문제를 분해하는 것 보다 전체문제를 푸는 것이 효율적이나 문제가 커질수록 분해문제가 더욱 우수함을 볼 수 있다. 그리고 전체문제로 형성하면 풀 수 없는 문제도 분해해법을 이용하면 풀 수가 있음을 보여 준다.

분해해법의 효율성은 처음 투입자료를 정리하는 단계에서 공급입지변수의 값을 몇 개로 고정할 것이냐에 따라 달라지는 것을 관찰할 수 있다. 이를 검사하기 위해 첫번째 문제 즉, 설치비용(1000, 1500)과 공급지와 수요지의 수가 각각 10개 및 50개인 문제에서 공급입지변수를 처음에 10, 7, 5, 2개를 설치하는 4그룹으로 나누어 실행해 보았다. 그 결과, 10개를 설치할 때는 반복횟수가 6회, 7개 설치할 때 3회, 5개 설치할 때 2회, 2개 설치할 때 5회로 나타났다.

즉, 처음 설치하는 입지가 예상공급지의 반과 같은 수준일 때 가장 효율적인 것으로 나타났다. 그래서 〈표 1〉의 실험은 공급지의 반이 처음 설치되는 것을 기준으로 실행하였다. 이 실험에서 알 수 있는 것은 최초에 고정한 공급입지변수 Y의 값에 따라 알고리듬의 효율성이 달라질 수 있다는 것이다.

5. 결 론

이 연구는 1차적이며 2차적인 공급입지를 필요로하는 '저장능력이 무한대인 장소입지문제'의 실제 사용에 실용성을 높이기 위해 고안되었다. 이 문제는 2차적인 공급입지를 꼭 필요로하는 위급상황이나 공급이 중단없이 요구되는 상황에 적합한 문제이다. 이 2차적인 입지의 필수성 때문에 1차적 입지만을 필요로 하는 문제보다 문제의 크기가 기하급수적으로 증가하게 되어 대규모의 정수계획문제로 형성된다. 대규모의 정수계획문제는 문제의 크기가 증가함에 따라 해법의 어려움이 기하급수적으로 증가하며 또 일정 규모 이상은 현재의 선형 혹은 정수계획패키지로는 해법이 불가능하다. 또한 2차적인 공급입지의 필수성 때문에 이 문제는 특수한 형태로 모형이 형성된다. 대규모 특수형태를 가진 문제는 분해해법을 적용할 때 그 효율성이 높다는 것이 일반적으로 알려져 왔다.

이 연구는 첫째, 벤더즈 분해기법을 적용하여 주문제와 하위문제들을 분해하고 전체 최적해를 구하기 위한 알고리듬을 도출하므로 벤더즈 분해기법의 적용영역을 확장하였다. 두번째, 분해된 주문제와 하위문제의 부문최적해를 구하고 이 문제사이에 필요한 자료의 이동을 자동적으로 수행하기 위해 GAMS라는 순서적 모형언어를 사용하였다. GAMS는 선형계획문제와 정수계획문제의 최적해를 구하기 위한 패키지를 활용할 수 있고 주문제와 하위문제들 사이에 필요한 자료의 이동을 수행할 수가 있다. 즉, 이 문제를 GAMS 내에서 해결하는 방법을 제시하므로 문제의 실용성을 높이게 되었고 이와 유사한 분해기법이나 동

태적인 모형에 활용할 수가 있다.

그러나, 이 문제는 다음과 같은 한계점을 가지고 있다. 최초에 몇개의 공급예상입지값을 1로 고정할 것이냐는 원칙을 제시하지 못했다는 것이다. 실행 결과에서 보여준 바와 같이 공급입지변수 몇 개를 고정하느냐에 따라 알고리듬의 효율성이 달라졌는데 이런 결정에 대한 대략의 원칙을 유도하지 못했다. 최초에 공급입지를 작은 수만 설치한다면 하위문제에서 할당변수들의 수자가 줄기 때문에 하위문제의 크기가 많이 줄어들 것이다. 대략의 선택기준은 입지를 건설하는 비용(v_j)이 우선적이며 보조적인 공급비용(c_{ij}와 $\bar{c}_{ij}$)보다 상대적으로 아주 높다면 예상공급입지를 많이 선택하는 것이 바람직하며 반대일 경우에는 예상공급입지를 작게 선택해도 되겠다. 최초에 몇개의 공급예상입지가 선택되는 것이 알고리듬의 효율성을 높일 것인가를 알아보기 위해 건설비용과 공급비용의 관계에 대한 폭넓은 컴퓨터 실행이 필요할 것이다.

참 고 문 헌

[1] 이상진, "불확실성하의 목표계획법을 적용한 무기체계 소요판단," 「국방연구」, 제37권, 제2호(1994), pp. 185-208.

[2] 이상진, "확률적 선형계획문제: 수학적 모형, 해법, 군사적 적용," 「교수논총」, 국방대학원, 제3집 1호(1995), pp. 181-208.

[3] Benders, J. K, "Partitioning procedures for solving mixed-variables programming problems," *Numerische Mathematic*, Vol. 4(1962), pp. 238-252.

[4] Brooke, A., D. Kendrick, and A. Meeraus, *GAMS: A User's Guide*, The Scientific Press, Redwood City, CA, 1988.

[5] Erlenkottler, D., "A dual based procedure for uncapacitated facility location," *Operations Research*, Vol. 26(1978), pp. 992-1009.

[6] Geoffrion, A. M. and G. W. Graves, "Multicommodity distribution system design by Benders decomposition," *Management Science*, Vol. 20(1974), pp. 822-844.

[7] Lasdon, L. S., *Optimization Theory For Large Systems*, The Macmillan Company, New York, 1970.

[8] Parker, R. G. and R. L. Rardin, *Discrete Optimization*, Academic Press, San Diego, CA, 1988.

[9] Pauler, G. E. and C. A. Floudas, "APROS: Algorithmic development methodology for discrete-continuous optimization problems," *Operations Research*, Vol. 37(1989), pp. 902-915.

[10] Pirkul, H., "The uncapacitated facility location problem with primary and secondary facility requirements," *IIE Transactions*, Vol. 21(1989), pp. 337-348.