□ 研 究 論 文 □

韓國 軍事運營分析 學會誌
第 11 券 第 2 號, 1985.12.

整數計劃法을 利用한 프로젝트 拡張順序決定에 関한 研究

姜 錫 昊*
金 智 承**

Abstract

Planning for the expansion of production capacity is of vital importance in many applications within the private and public sectors. This paper considers a sequencing expansion problem in which capacity can be added only at discrete points in time.

Given the demand forecast of each period, capacity and cost of each expansion project, we are to determine the sequence of expansion necessary to provide sufficient capacity to meet the demand in all periods at minimum cost. This problem is formulated as a pure integer programming and solved by branch and bound method using Lagrangian relaxation.

At first, simple sequencing expansion problem is presented, and in the latter part, extension to include precedence between projects is suggested.

1. 序 論

점차 늘어나는 需要에 대비하여 生産容量擴張에 관한 계획을 세우는 것은 現代 企業이 당면하고 있는 매우 중요한 문제이다. 1950년대 이후 경영과학방법이 이 分野에서 널리 사용되고 있는데, Luss〔11〕가 分類한 것에 따르면 다음과 같이 나누어 볼 수 있다.

(1) 單一 설비에 있어서의 容量 擴張 문제
(2) 한정된 프로젝트가 있을 때의 容量 擴張 문제
(3) 複數 설비에 있어서의 容量擴張문제

여기서 다룰 것은 (2)번 즉 한정된 프로젝트가 있을 때의 容量擴張 문제에 관한 것인데, 각 期間의 수요가 確定的

* 서울大學校 工科大學
** 國防管理研究所

이고, 개개의 프로젝트에 대해 추가시키는데 드는 費用과 운영하는데 드는 費用 그리고 각 프로젝트의 容量을 적절히 豫測할 수 있을 때 어떤 順序대로 프로젝트를 추가시키는 것이 매 期間의 需要를 만족시키면서 總費用을 最小化시키는 것인지를 決定하려는 것이다. 처음에는 프로젝트간에 상관관계가 없는 기본적인 模型과 解法을 제시하고 그 다음 이 기본 模型에 프로젝트간의 先後관계를 고려한 模型의 解法을 제시할것이다.

일반적으로, sequencing 문제는 動的計劃法으로 많이 풀었는데, 이때까지 硏究된 것을 간략하게 살펴보면 다음과 같다.

Dale〔7〕은 프로젝트 擴張順序 決定模型을 발전소 擴張計劃에 適用하였다. 즉, 발전소에 動力을 추가시키는데 그 時期와 어떤 容量의 프로젝트를 추가할것인가를 決定하는 것이다. 그는 이 模型에서 前進 循環式을 이용한 動的計劃法을 사용했는데, 문제 크기가 커짐에 따라 즉 可能한 狀態(state)의 數가 많아짐에 따라 컴퓨터 저장 容量(storage)의 한계때문에 近似解를 찾게 되었다.

Ackerman과 Luss〔4〕는 順序 決定模型을 전화교환기 대체 順序를 決定하는데 適用하였고, Baker와 Schrage〔6〕는 順序에 의해 作業이 연결될 때의 문제를 푸는 動的計劃法에 관해 硏究하였는데 여기서 실제 컴퓨터 사용상의 문제점에 많은 관심을 기울였다.

Erlenkotter〔8〕는 時間을 연속적으로 생각한 模型에 대해 자세히 言及했는데, 作業 順序 決定模型을 푸는데 사용되는 動的計劃法과 비슷한 動的計劃法으로 이와 같은 문제를 푸는 方法을 제시했다. 또한 그는 〔9〕 프로젝트간에 상호 연관이 있는 模型에 관해서도 硏究하였다.

이와 같이 sequencing 문제를 動的計劃法으로 많이 풀어 왔는데, Erlenkotter〔8〕 그리고 Baker와 Schrage〔6〕가 지적했듯이 動的計劃法은 프로젝트 數가 많아짐에 따라 計算上에 어려움이 따른다. 그래서 여기서는 Lagrangian relaxation을 사용한 分枝 限界法(Branch and Bound method)을 이용하여 이 문제를 解決하려는 것이 硏究 目的이라 하겠다.

2. 基本 模型

이 模型은 時間을 이산적(discrete)으로 생각했을 때 生産容量擴張 順序를 決定하는 것이다. 여기에 必要한 가정들은 다음과 같다.

(1) 각 期間에 요구되는 需要는 確定的이고 需要는 시간에 따라 減少하지 않는다.
(2) 각 프로젝트의 容量과 費用은 알려져 있다.
(3) 計劃期間은 有限하고 프로젝트 갯수도 限定되 있다.
(4) 프로젝트 容量은 한번 추가되면, 전 기간이 끝날 때까지 그 容量은 지속된다.
(5) 초기 容量은 없다고 한다.
(6) 기간 t에 사용 可能한 총 容量은 그 期間의 需要보다 적어도 같거나 커야한다. 즉 需要는 만족되어야만 한다.
(7) 프로젝트 容量은 즉시 추가될 수 있다.
(8) 한 期間에 많아야 하나의 프로젝트만 추가될 수 있다.

이 가정은 現實에 模型을 適用시킬때

많이 생기고 또한 시간을 연속적으로 생각한 模型에서도 자주 강조된다.

위와 같은 가정하에 이 문제를 다음과 같이 0-1 整數計劃法으로 模型化시킬 수 있다.

Minimize

$$\sum_{i=1}^{n}\sum_{t=1}^{T}(C_{it}+\sum_{k=t}^{T}O_{ik})\ x_{it}$$

subject to

$$\sum_{t=1}^{T} x_{it} \leqq 1 \text{ for all } i \in I \qquad (2\text{-}1)$$

$$\sum_{i=1}^{n} x_{it} \leqq 1 \text{ for all } t=1,2,\cdots,T \qquad (2\text{-}2)$$

$$\sum_{t=1}^{k}\sum_{i=1}^{n} g_i\, x_{it} \geqq d_k \text{ for all } k=1,2,\cdots,T \qquad (2\text{-}3)$$

$$x_{it}=0,\ 1 \qquad (2\text{-}4)$$

where

T : 計劃 期間 (즉, $t=1,2,\cdots,T$)

$I=\{1,2,\cdots,n\}$: 프로젝트i의 집합

g_i : 프로젝트 i의 容量(정수)

c_{it} : 프로젝트 i를 期間 t에 추가시키는데 드는 費用(정수)

o_{it} : 프로젝트 i를 기간 t에 운영하는데 드는 비용

d_t : 期間 t의 需要(정수)

x_{it} : $\begin{cases} 1 : \text{프로젝트 i가 期間 t에 추가될 때} \\ 0 : \text{o, w} \end{cases}$

위의 모형에서

제약식 (2-1)은 모든 프로젝트는 많아야 한번 추가될 수 있다는 것을 뜻하고, 제약식 (2-2)는 한 期間에 많아야 한 프로젝트가 추가될 수 있다는 것을 뜻하고, 제약식 (2-3)은 需要는 만족되어야만 한다는 것을 의미한다.

3. Lagrangian Relaxation과 Subgradient Optimization

원 문제에서 제약식 (2-3)이 문제를 풀기 어렵게 만든다. 그래서 Lagrangian relaxation을 사용하여 제약식 (2-3)을 완화시킨다. μ를 Lagrange 백터로 놓고 μ_i를 제약식 (2-3)의 i번째에 해당하는 Lagrange 승수라고 하자. 그러면 완화된 Lagrangeian 문제는 다음과 같이 된다.

$$L(\mu) = \text{Minimize } L(\mu : x)$$

$$= \text{Minimize} \sum_{i=1}^{n}\sum_{t=1}^{T}(C_{it}+\sum_{k=t}^{T}O_{ik} - g_i\sum_{k=t}^{T}\mu_k)\,x_{it} + \sum_{t=1}^{T}\mu_t\, d_t \qquad (3\text{-}1)$$

subject to

(2-1), (2-2), (2-4), $\mu_k \geqq 0$ for all $k=1,2,\cdots,T$

(3-1)과 같이 완화된 Lagrangian 문제는 배정문제 (assignment problem) 기법을 이용하여 쉽게 解를 구할 수 있다. 여기에 관한 것은 다음 절에서 상세히 다루기로 한다.

$\bar{X}$가 (3-1)의 解라고 놓고, $L(\mu : \bar{X})$를 주어진 μ에 대한 목적 함수 값이라 하자. 그러면 $L(\mu : \bar{X})$는 원 문제의 하한 경계치 (lower bound)가 된다. [14] 최적의 Lagrangian 백터 μ는 다음 문제의 최적해가 된다.

Maximize $L(\mu : X)$

subject to $\mu \geqq 0$ (3-2)

좋은 Lagrange 승수 백터를 얻기 위해서 여기서는 subgradient 알고리듬을 사용한다. subgradient 알고리듬은 초기 Lagrange 승수 백터 μ°를 가지고 시작하여 차례차례 $\mu^1, \mu^2, \cdots, \mu^s$를 구해나가는 方法인데, 一般的으로 원 문제에 適用될 때는 S를 20에서 60사이로 잡고, 그 이하의 후보 문제(candidate problem)에 適用될 때는 10이하로 잡는다. S가 커짐에 따라 μ^s는 (3-2)의 최적해에 漸近的으로 收斂한다.

$L(\mu : \bar{X})$는 分枝限界法에서 단지 후보 문제의 하한 경계치로 사용되기 때문에 (3-2)의 최적해를 반드시 얻어야만 되는 것은 아니다. 될수 있는 한 $L(\mu:\bar{X})$가 큰 Lagrange 승수 백터 μ^*를 얻어 사용하면 되는 것이다.

subgradient 최적화 알고리듬을 정리해 보면 다음과 같이 2단계로 나눌 수 있다.

(1) 초기 Lagrarge 승수 백터 $\mu^{\circ} \geqq 0$을 선택한다.

원 문제에 相應한 하한 경계치를 구할 때는 일반적으로 μ°를 0으로 잡고 그 밖의 후보 문제를 풀 경우에는 그 前 마디(previous node)의 μ^*를 초기 Lagrange 승수 백터로 잡는다.

(2) 주어진 μ^j에 대해 다음 번 Lagrange 승수 백터는 다음과 같이 구해진다.

$$\lambda_j = \rho_j(\bar{L} - L(\mu^j)) / \|V^j\|$$

로 놓으면

$$\mu_i^{j+1} = \mathrm{Max}\{0, \mu_i^j + \lambda_j V_i^j\}$$

for all $i = 1, 2, \cdots, T$

where V^j = subgradient 백터

$\varepsilon \leqq \rho_j \leqq 2$

ρ_j에 대한 초기 값으로는 一般的으로 2를 잡는다. 마지막 d단계 동안에 $L(\mu^j)$의 값이 증가하지 않으면, ρ_j의 값을 반으로 줄인다.

$\bar{L}$는 $L(\mu:\bar{X})$의 상한 경계치인데 여기서는 現 완전해의 목적함수 값을 $\bar{L}$로 잡았다.

4. Lagrangian 문제의 解法

주어진 μ에 대해 (3-1)을 푼다는것은

$$S_{it} = C_{it} + \sum_{k=t}^{T} O_{ik} - g_i \sum_{k=t}^{T} \mu_k$$

로 놓으면 다음과 같은 문제를 푸는 것과 同一하다.

Minimize $S_{it}\, x_{it}$

subject to (4-1)

$$\sum_{i=1}^{n} x_{it} \leqq 1 \quad \text{for all } t = 1, 2, \cdots, T$$

$$\sum_{t=1}^{T} x_{it} \leqq 1 \quad \text{for all } i \in I$$

$(\mu_k \geqq 0 \text{ for all } k = 1, 2, \cdots, T)$

윗 문제는 전형적인 配定 문제로 변환될 수 있다. 즉 바이퍼타이트(bipartite) 네트워크를 만들기 위해서 i에는 T개, t에는 n개의 假지점(dummy node)을 추가시켜 이 假지점에 연결된 費用을 0으로 놓으면 配定 문제와 同一한 다음 模型과 같이 (4-1)을 변환시킬수 있다.

Minimize $S'_{it}\, x_{it}$

subject to (4-2)

$$\sum_{i=1}^{n+T} x_{it} = 1 \quad \text{for } t = 1, 2, \cdots, T, \cdots$$

$n+T$

$$\sum_{t=1}^{n+T} x_{it} = 1 \text{ for } i = 1, \cdots, n, \cdots, n+T$$

where

$$S'_{it} \begin{cases} S_{it} & 1 \leqq i \leqq n,\ 1 \leqq t \leqq T \\ O & \text{other wise} \end{cases}$$

5. 分枝限界法

subgradient 최적화 알고리듬을 適用해서 얻은 解가 원 문제의 최적해는 아니다. 그래서 分枝限界法이 최적해를 얻기 위해 사용된다.

후보 문제에 Lagrangian relaxation을 適用시키는 과정에서 緩和된 문제의 최적해가 원 문제의 모든 제약식을 만족시키고 또한 補充的 餘裕條件(complementary slackness condition)인

$$\mu_k \cdot [d_k - \sum_{t=1}^{k} \sum_{i=1}^{n} g_i x_{it}] = 0, \text{ for all } k = 1, 2, \cdots, T$$

를 만족시키면 즉, 후보 문제의 하한 경계치 $L(\mu : X)$와 완화된 문제의 최적해에 相應한 원 문제의 목적함수 값이 같으면 分枝完了(fathom)가 된다. 만약 후보 문제가 分枝完了가 되지 않으면, 다시 分枝 전략이 적용되어 2개의 새로운 후보 문제를 형성하게 된다.

$x_{it} = 0$인 集合을 X_0로 놓고, $x_{it} = 1$인 集合을 X_1으로 놓으면, 分枝 나무의 어느 마디에서나 다음과 같은 제약식이 하나 더 추가된다.

$$x_{it} = 0 \text{ for all } x_{it} \in X_0$$

$$x_{it} = 1 \text{ for all } x_{it} \in X_1$$

어떤 x_{it}가 X_1에 속하게 되면, 나머지 $x_{it}(t \neq k)$들은 모두 X_0에 속하게 된다.

위의 제약식하에서 후보 문제를 配定문제로 풀기 위해서는 M을 매우 큰 양의 整數로 놓았을 때, X_0에 속하는 x_{it}에 해당되는 費用 C_{it}는 M, X_1에 속하는 x_{jk}에 해당되는 費用 C_{jk}는 $-M$으로 놓고 풀면 된다.

分枝 전략으로 $r_t (= d_t - d_{t-1})$가 처음으로 양이 되는 기간 k와 X_0나 X_1에 속하지 않은 j를 선택한다. 이와같이 하여 2개의 새로운 마디를 형성한 후 한쪽에는 $x_{jk} = 0$, 다른 한쪽에는 $x_{jk} = 1$로 놓는다.

一般的인 探索 전략은 가장 적은 하한 경계치를 갖는 마디를 선택하여 그 마디를 다시 分枝하는 것이다. 그러나 이 전략은 매우 많은 計算量을 必要로 할지 모른다. 그래서 여기서는 LIFO (last-in-first-out) 方法을 썼는데, 이 방법은 하나의 후보 문제만을 가지고있고 나머지는 스택(stack)에 저장시키는 것이다. 즉, 現 후보 문제가 分枝 끝이 되면 스택에서 새로운 후보문제를 선택한다.

LIFO 方法에서는 마지막으로 스택에 들어온 후보 문제가 선택되게 된다.[2]

이 探索 전략이 사용되면, 分枝限界法은 스택에서 후보 문제를 선택해야할때 스택이 비어 있으면 끝나게 된다. 그리고 이 단계까지의 완전해(incumbent solution)가 원 문제의 最適解가 된다.

6. 프로젝트간에 先後관계가 있는 模型

이 模型은 앞의 기본 模型에 프로젝트간의 先後관계를 고려한 것인데, 이 경우 기본 模型과 같은 方法으로 풀기가 어려우므로 조금 다르게 模型化시켰다.

만약 프로젝트 a가 추가되기 위해서는 프로젝트 b가 먼저 추가되어야 한다면, t기간 初에 남는 容量 y_t를 決定변수로 집어 넣어 다음과 같이 模型化시킬 수 있다. 先後관계에 해당되는제약식은 여러 개일수도 있다.

$$\text{Minimize} \sum_{i=1}^{n} \sum_{t=1}^{T} (C_{it} + \sum_{k=t}^{T} O_{ik})\, x_{it}$$

subject to

$$\sum_{t=1}^{T} x_{it} \leqq 1 \quad \text{for all } i \in I \qquad (6\text{-}1)$$

$$\sum_{i=1}^{n} x_{it} \leqq 1 \quad \text{for all } t = 1, 2, \cdots, T \qquad (6\text{-}2)$$

$$\sum_{i=1}^{n} g_i\, x_{it} + y_t - y_{t+1} = r_t \quad \text{for all } t = 1, 2, \cdots, T \qquad (6\text{-}3)$$

$$\sum_{i=1}^{T} t(x_{at} - x_{bt}) \geqq 0 \qquad (6\text{-}4)$$

$$x_{it} = 0 \text{ or } 1 \qquad (6\text{-}5)$$

$$y_t \geqq 0 \qquad (6\text{-}6)$$

제약식 (6-1), (6-2), (6-5)는 기본 模型과 같고, 제약식 (6-6)은 새로이 添加된 決定 변수 y_t에 관한 것이고, 제약식 (6-4)는 先後 관계를 나타내는 것이다. 그리고 제약식 (6-3)은 容量 均衡式 (capacity balance equation)인데 y_t의 정의에서 유도된다.

윗 模型은 기본 模型처럼 Lagrangian 문제를 配定 문제 기법으로 풀기가 좋지 않다. 왜냐하면, 기본 模型처럼 풀기 위해서는 2개의 제약식을 緩和시켜야 하는데, 先後 관계가 많아짐에 따라 解의 性質이 나빠지기 때문이다. 그래서 이 模型에서는 제약식 (6-1)을 緩和시켜 Lagrangian 문제를 後進循環式을 이용한動的計劃法으로 풀었는데, 이 動的計劃法에서는 프로젝트간의 先後 관계를 쉽게 處理할 수 있다.

제약식 (6-1)이 緩和된 Lagrangian 문제는 아래와 같이 된다.

$$\text{Minimize } L(\mu : X) = \sum_{i=1}^{n} \sum_{t=1}^{T} (C_{it} + \sum_{k=t}^{T} O_{ik} + \mu_i)\, x_{it} - \sum_{i=1}^{n} \mu_i$$

subject to

(6-2) ~ (6-6)

$$\mu_i \geqq 0 \quad \text{for all } i = 1, 2, \cdots, n$$

윗 문제는 t기간에 프로젝트 i를 추가시키는데 드는 費用이 $(C_{it} + \sum_{k=t}^{T} O_{ik} + \mu_i)$로 변환되었다고 볼 수 있다. 그러므로 μ에 대해 보면, 윗 문제는 변환된 總 費用을 最小化하는 프로젝트를 매 期間 決定하는 것이다. 다만 여기서는 한 프로젝트가 여러번 추가될 수 있다는 것만이 다른 條件이 된다.

이 Lagrangian 문제는 期間 t를 단계 (stage)로 생각하여 動的計劃法으로 풀 수 있다. 단계 t에서의 狀態 변수 (state variable)는 y_t가 된다. 後進循環式을 써 보면 아래와 같다.

初期 條件 $F_{T+1}(y_{T+1}) = 0$ for all $y_{T+1} \geqq 0$

$$F_t(y_t) = \min_{y_{t+1}} \{ k_t(r_t + y_{t+1} - y_t) + F_{t+1}(y_{t+1}) \}$$

subject to

$$\text{Max}\{0, y_t - r_t\} \leqq y_{t+1} \leqq y_t - r_t + g(1)$$

for $y_t = 0, 1, 2, \cdots, u(y_t)$

where

$F_t(y_t)$: t 期間 初에 남는 容量이 y_t 일 때 남은 期間의 最小 總 費用

$k_t(b)$: 期間 t 에 容量 b만큼 추가 시키는데 드는 最小 費用

$g(1)$: 가장 큰 g_i, $i=1,2,\cdots,n$

$u(y_t)$: y_t 의 상한 상계치

$$(=\sum_{i=1}^{\min(n,t-1)} g(i) - \sum_{j=1}^{t-1} r_j)$$

이 後進循環式에서 $k_t(b)$는 다음과 같이 쉽게 구할 수 있다.

$$k_t(b) = \text{Min}_i \{ C_{it} + \sum_{k=t}^{T} O_{ik} + \mu_i / g_i = b,\ x_{it} \in X_0 \cup X_1 \}$$

$$k_i(b) = \infty \text{ if } b \neq g_i \text{ for any } x_{it} \in X_0 \cup X_1$$

7. 結論 및 討議事項

이상에서 본 바와 같이, 프로젝트 擴張順序 決定模型은 整數計劃法으로 模型化하여 Lagrangian relaxation을 이용한 分枝限界法으로 풀 수 있었다. 또한 一般的인 動的計劃法으로 풀 때의 문제점인 "curse of dimensionality" 즉 프로젝트 數가 많아짐에 따라 計算量이 매우 급하게 증가하는 문제점을 어느 정도 解決할 수 있었다.

이 模型은 水資源 시스템같이 限定된 프로젝트 집합에서 어느 특정 프로젝트를 추가시킴으로써 容量을 증가시킬 수 있는 分野에서 사용되어 진다.

한편, 추후 硏究 課題로서

(1) 각 프로젝트별로 여러 容量 水準이 있을 때

(2) 容量간에 어떤 상관 관계가 있을 때

(3) 費用간에 어떤 상관 관계가 있을 때 등이 있는데

(1)번은 X에 0, 1 뿐만 아니라 2, 3, 4, … 를 취할 수 있게 함으로써 解決될 수 있을 것이다. 이 경우 計算量이 매우 增加하게 되므로 이 문제를 解決하는 方案도 아울러 硏究되야 할 것이다.

(2)번은 프로젝트간의 先後관계를 包含한다고 할 수 있는데, 예를 들어 보면, 어떤 프로젝트의 容量이 다른 특정 프로젝트가 추가되어 있지 않을 경우에는 완전히 사용 可能할 수 없는 경우를 말한다.

References

1. 강석호, Operations Research, 영지문화사, 서울, 1980.
2. 홍영식, 엄기현, 자료구조, 정익사, 서울, 1982. 53~71.
3. Geoffrion, A.M., "Lagrangian Relaxation for Integer Programming," *Mathematical Programming Study 2,* 1974, 82-114.
4. Ackerman, L.J., Luss, H. and Berkowitz, R.S., "Application of Sequencing Policies to Telephone Switching Facilities," *IEEE Trans.* SMC 7, 1977, 604-609.
5. Baker, K.R., *Introduction to Scheduling and Sequencing,* John Wiley & Sons, Inc., 1974.
6. Baker, K.R., and Schrage, L.E., "Finding an Optimal Sequence by Dynamic Programming: An Extension to Precedence-related Tasks," *Operations Research,* 26, 1978, 111-120.

7. Dale, K.M., "Dynamic Programming Approach to the Selection and Timing of Generation-Plant Additions," *Proc. Institute Electrical Engineerong.* 113, 1966, 803-811.
8. Erlenkotter, D., "Sequencing Expansion Projects," *Operations Research,* 21, 1973, 542-553.
9. Erlenkotter, D., "Sequencing of Interdependent Hydroelectric Projects," *Water Resource Research,* 9, 1973, 21-27.
10. Weingartner, H. Martin, "Capital budgeting of Interelated Projects: Survey and Synthesis," *Mgt. Science,* 7, 1966, 485-516.
11. Luss, H., "Operations Research and Capacity Expansion Problems: A Survey," *Operations Research,* 30, 1982, 907-947.
12. Murty, K., *Linear and Combinatorial Programming,* John Wiley and Sons, Inc., 1976, 437-446.
13. Manne, A.S. (ed), *Investments for Capacity Expansion, Size, Location and Time-phasing,* MIT press, Cambridge, Mass., 1967.
14. Fisher, M., "The Lagrangion Relaxation Method for Solving Integer Programming Problems," *Mgt. Science,* 27, 1981, 1-18.
15. Larson, R.E., and Casti, J., *Principles of Dynamic Programming, Marcel Dekker,* Inc., 1982.