# 難解性 組合的 最適化問題의 풀이를 위한 一般的 接近方法에 관한 硏究

安 相 炯*
李 松 根**

《目 次》



## Ⅰ. 序 論

難解性 組合的 最適化 整數計劃問題(NP-complete Combinatorial Optimization Problem)는 輸送經路選定, 立地選定, 日程計劃 등의 현실적인 有用性이 큰 문제들의 일반적인 數理模型으로 나타나지만, 線型計劃問題와는 달리 最適解를 구할 수 있는 심플렉스와 같은 一般解法이 개발되어 있지 않다. 현재까지 극히 일부의 문제에 대해서 최적해를 구하기 위한 시도가 있었으나 이러한 최적해법은 막대한 계산시간을 요하는 비효율적인 해법이 대부분이다.

난해성 조합적 최적화문제는 계산의 복잡도에 있어서 指數時間을 요하는 문제가 대부분이며, 따라서 多項時間(polynomial time) 이내에 최적해를 구하기는 불가능하다고 알려져

* 서울大學校 經營大學 敎授
** 大邱大學校 經商大學 敎授

있다. 그러므로 현재까지의 연구의 중점은 최적해법의 모색보다는 오히려 실용적인 측면에서 최적해에 대한 近似解의 탐색에 치중되어 왔으며, 이를 위한 휴리스틱해법들의 개발과 이를 평가하는 最惡境遇分析(worst-case analysis) 또는 確率的分析(probabilistic analysis)이 기존 연구의 주류를 이루고 있다.

그러나 컴퓨터 하드웨어 기술의 발달과 더불어 계산시간이 단축되고 계산용량이 계속 증가되고 있으므로, 數理理論에 기초한 分析的 方法을 통한 최적해에의 접근이 가능해지고 있다. 물론 문제의 속성상 일정한 규모이내의 최적해 탐색만이 가능하지만, 이 규모가 실용적인 규모라면, 이와같이 도출된 最適解는 다음과 같은 유용성을 가진다.

첫째, 최적해의 현실적인 적용을 통하여 현실문제의 최적자원배분을 가능하게 한다.

둘째, 기존 휴리스틱해법들의 유용성이나 최악한계, 평균형태 등을 분석하고 평가하는 기준이 된다.

셋째, 난해성 조합적 최적화문제의 풀이를 위한 컴퓨터 소프트웨어의 발달을 촉진시킨다.

본 연구는 이와같은 유용성을 가지는 최적해를 구하는 절차로서 지금까지 여러 난해성 조합적 최적화문제에 대해서 시도되었던 방법들을 검토하여 이중 가장 일반적인 방법론을 정리하여 제시하고자 한다.

또한 이 방법을 아직 효율적인 최적해법의 탐색이 시도된 바 없는 전형적인 조합적 최적화문제인 병렬처리기계문제(parallel processor problem)에 적용하여 최적해법의 구성과정을 사례로서 제시하고자 한다. 그리고 이와같이 일반화된 방법론을 토대로 컴퓨터 실행을 위한 실제 프로그래밍 작성과정을 예시하기 위하여, 병렬처리기계문제의 최적해를 구하는 단계별 절차를 FORTRAN 프로그램으로 구성하였다.

## Ⅱ. 難解性 組合的 最適化問題의 槪念

### 2.1. 難解性 組合的 最適化問題의 屬性에 관한 理論的 考察

난해성 조합적 최적화문제(NP-complete problems)의 존재에 관한 이론적 기초는 Stephen Cook(2)의 논문 "The Complexity of Theorem Proving Procedure"에서 구축되었다. Cook은 이 논문에서 다음 이슈를 제시하였다.

#### 2.1.1. 多項時間解法의 重要性

多項時間解法(polynomial time algorithm)은 指數時間解法(exponential time algorithm)과 구별되는 개념으로서, 이 구분은 時間複雜度函數(time complexity function)에 기초하고 있

다. 특정문제에 대한 어떤 해법의 時間複雜度函數란, 가능한 모든 크기의 입력자료에 대해서 해당문제를 풀려고 할 때 소요되는 최대 계산시간을 함수로 표현한 것이다. 어떤 하나의 문제가 주어졌을 때 이 문제를 풀이하는 해법들이 다양하게 존재할 수 있다. 그러나 각 해법의 시간복잡도함수는 해법의 논리적 구조에 따라 서로 다르게 나타나며, 특정해법이 계산소요시간의 관점에서 충분히 효율적(efficient)인지, 아니면 비효율적(inefficient)인지의 판단기준은 일반적으로 그 해법이 多項時間解法인지 아니면 指數時間解法인지에 두고 있다.

예를 들어 어떤 함수 $f(n)$에서 입력자료의 크기를 나타내는 $n>0$에 대하여 $|f(n)|\leq C|g(n)|$을 만족시키는 상수 $C$가 존재한다면 함수 $f(n)$의 時間複雜度函數는 $0(g(n))$이 된다. 多項時間解法은 어떤 다항식함수(polynomial function) $P$에 대해서 時間複雜度函數가 $0(P(n))$으로 나타나는 경우의 해법으로 정의된다. 이에 반해서 時間複雜度函數가 위와 같은 다항식함수로 나타낼 수 없는 경우의 해법을 指數時間解法이라 한다.

이 두 형태의 해법의 구분은 문제의 크기가 대규모인 경우에 대단히 중요한 의미를 가진다. 예를 들어 문제의 규모를 $n$으로 나타낼 때, $n$이 증가하는 경우에 다항함수인 $n^5$에 비해서 지수함수인 $5^n$의 계산량은 폭발적으로 증가하게 된다. $n=30$만 되어도 지수함수문제는 아무리 최첨단 컴퓨터를 동원하더라도 최적해를 구하는 데는 수 세기가 소요될 수도 있다. 이러한 이유에서 多項時間解法은 指數時間解法에 비해서 훨씬 바람직한 해법으로 간주된다. 일반적으로 어떤 문제에 대해서 多項時間解法이 발견되면 그 문제는 잘 풀렸다고 보며, 그렇지 못한 경우에는 그 문제를 追跡不可能(intractable)한 문제로 분류한다.

2.1.2. NP 系列問題와 P系列問題의 구분

Cook은 또한 NP계열의 의사결정 문제에 대한 관심을 야기시켰는데, NP계열의 문제란 非決定的컴퓨터(nondeterministic computer), 즉 무한 개수의 독립적인 계산절차를 竝列(parallel)로 처리할 수 있는 컴퓨터를 사용해서야만 비로소 다항시간이내에 풀 수 있는 비교적 어려운 문제집단을 의미한다. 반면에 P계열의 문제란 決定的컴퓨터(deterministic computer), 즉 매 단계마다 유한 수의 작업을 순서대로 처리하는 컴퓨터를 사용해서 다항시간 이내에 풀 수 있는 비교적 쉬운 문제집단을 의미한다.

현재 비결정적컴퓨터는 기술적으로 개발이 안되어 있는 상태이므로 일반적인 결정적 컴퓨터를 전제로 했을 때, 다항시간해법이 존재하는 문제집단을 P계열문제로 간주하고 아직 다항시간해법이 존재하는지 아닌지의 여부가 확인되지 않은 문제집단을 NP계열문제로 간주할 수 있다.

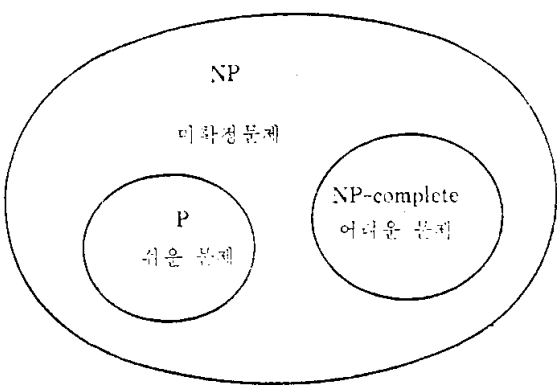
2.1.3. 滿足性問題(satisfiability problem)의 존재

Cook은 NP중에서 하나의 특정문제, 즉 만족성(satisfiability) 문제는 NP에 속하는 모든 문제들이 이 문제로 다항시간 이내에 귀속(reduce)될 수 있는 속성을 가진다는 것을 증명하였다. 따라서 만약 만족성 문제가 多項時間解法으로 풀려질 수 있다면 NP에 속하는 다른 모든 문제도 多項時間解法으로 풀 수 있다는 것을 의미하게 되고, 반면에 NP에 속하는 어떤 문제가 追跡不可能(intractable)하다면 만족성 문제도 당연히 追跡不可能해 진다. 그러므로 만족성 문제는 NP문제 중에서 가장 難度가 높은 핵심적인 문제라고 하였다.

2.1.4. NP-complete 問題의 槪念

Cook은 NP에 속하는 또 다른 문제들 중에는 NP의 가장 어려운 속성을 만족성 문제와 동등한 정도로 가지고 있는 문제들이 있음을 제시하였다. 한편, Richard Karp(20)은 Cook의 연구를 계승하여 일련의 연구 결과를 발표하였는데, 그에 의하면, 외판원 문제(traveling salesman problem)를 포함한 많은 유명한 조합적 최적화의 문제가 만족성 문제와 동등한 정도의 난해성을 가짐을 증명하였다.

이후에도 다양한 문제들이 이러한 문제와 그 難度에 있어서 동등함이 증명되었는데, NP 문제 중에서 이러한 문제들을 난해성 조합적 최적화문제(NP-complete problem)로 분류하게 되었다. 다시말해서 만족성문제만큼 다항시간해법의 발견이 어렵다고 간주되는 문제집단을 NP-complete 계열문제라고 한다. 이와 같이 다양한 문제집단들을 NP-complete 계열문제로 분류함으로써 어떤 하나의 NP-complete 문제가 풀려질 수 있느냐의 여부를 묻거나 증명하기보다는 NP-complete 문제집단 전체가 풀려질 수 있느냐 없느냐를 증명하는 것이 중요과제로 대두되었다. 현재로서는 NP-complete 문제가 모두 追跡不可能(intractable)한지 또는 多項時間解法이 존재할 수 있는지에 대한 증명 또는 반대 증명이 확립되어 있지 못한




〈그림 2-1〉 NP 문제의 구성

상태이며, 많은 연구자들이 NP-complete 문제는 모두 追跡不可能할 것이라고 추정만 하고 있을 뿐이다. 그러므로 어떤 문제가 NP-complete 문제로 분류된다는 사실만으로도 그 문제를 多項時間解法으로 풀기 위해서는 최소한 획기적인 연구의 진전이 있어야 함을 의미한다.

P와 NP 및 NP-complete의 관계는 〈그림 2-1〉 같이 나타낼 수 있다.

### 2.2. 難解性 組合的 最適化問題의 接近方法

난해성 조합적 최적화문제로 증명된 문제에 직면하였을 때, 문제해결을 위한 접근방법은 다음 두가지 범주로 나누어 생각해 볼 수 있다.

첫번째 범주의 접근방법은 최적해의 발견에 집착하지 않고 허용 가능한 계산시간 범위 이내에 훌륭한 準最適解를 탐색하는 방법이다. 準最適解를 구하는 해법을 휴리스틱해법이라 한다. 이러한 해법을 구성하는 방법은 문제마다 고유한 구조적 특징이나 속성을 이용해야 하므로 문제의 유형에 따라 다양한 방법들이 존재하게 된다. 휴리스틱 접근방식은 비교적 간단한 절차에 의거해서 짧은 계산 시간으로 최적해에 가까운 실행가능해를 구할 수 있기 때문에 지금까지의 대부분의 난해성 조합적 최적화문제에 대한 연구는 보다 실용성을 가지는 효율적인 휴리스틱해법의 발견에 치중해왔다.

두번째 접근방법은 난해성 조합적 최적화문제임을 감안하여 지수시간복잡도의 필요성을 어느 정도 인정하면서 최적해를 탐색하는 것이다. 이러한 접근방식 중에서 가장 널리 사용되는 방법은 동적계획법(dynamic programming)과 분단탐색법(branch-and-bound)이다. 동적계획법으로 구성된 최적방법은 거의 지수시간해법으로서 문제의 규모가 조금만 커지면 계산시간이 엄청나게 증가하는 비효율적인 해법이 대부분이다. 한편 분단탐색방법은 나무구조의 탐색양식내에서 부분적인 해(partial solutions)들을 생성시키면서, 실행가능해가 될 수 없는 부분해를 삭제하기 위한 강력한 상한가와 하한가를 활용하여 탐색가지들을 잘라가는 방법이다. 이때 상한가는 주로 휴리스틱해법으로 구한 최선의 실행가능해가 이용되고 하한가는 보다 풀이가 용이한 이완문제를 풀어서 구한다. 분단탐색의 효율을 결정하는 가장 중요한 요소는 어느정도 최적해에 근접한 상한가와 하한가를 구할 수 있느냐 하는 것이다. 분단탐색으로 구성된 최적해법은 일종의 유사다항시간해법(pseudo-polynomial algorithm)이 되므로 상당한 규모의 문제까지 최적해를 적정시간이내에 구할 수 있게 된다. 대표적으로 Gavish와 Pirkul(8)의 다수제약식 0~1 배낭문제와 Gavish와 Srikanth(9)의 복수외판원 문제 등을 위시하여 입지문제(location)와 일정계획(scheduling) 분야의 많은 NP-complete 문제들에 대한 최적해탐색에 이와 같은 방법론이 시도되고 있고 상당한 성공을 거두고 있는

것으로 보고되고 있다. 본 논문에서는 난해성 조합적 최적화문제에 대한 효율적인 최적해법의 구성방법론에 초점을 맞추고 이를 체계화하고자 하므로 다음 장에서는 분단탐색에 의한 최적해법의 구성에 필요한 이론적 배경을 정리하여 제시한다.

## Ⅲ. 分段探索에 의한 最適解法의 구성을 위한 理論的 背景

### 3.1. 휴리스틱해법의 개발

분단탐색의 방법에 의해서 난해성 조합적 최적화문제의 최적해를 효율적으로 구하기 위해서는 최적해에 가능한 가까이 접근하는 실행가능해를 구하여 이를 상한가로서 이용하여야 한다. 이러한 실행가능해를 구하기 위해서는 효율적인 휴리스틱해법를 개발하여야 한다, 바람직한 휴리스틱해법은 다음 특징을 가져야 한다. 첫째 적정계산시간 이내에 실행될 수 있어야 하고, 둘째 산출된 휴리스틱해가 평균적으로 최적해에 가까이 존재해야 하며, 세째 최악경우의 해가 최적해로부터 일정비율 이상 벗어나지 않는다는 것이 보장되어야 하고, 네째 해법의 설계가 단순해야 한다.

이러한 휴리스틱해법을 구성하기 위한 설계방법으로서 Fisher등(5)과 Foulds(6)는 현재까지 여러 연구에서 시도된 방법들을 정리하여 다음 유형으로 구분하였다.

1. 단일경로방식(single pass method)

하나의 절차에 의거해서 문제의 자료를 한차례 한번씩 전체적으로 검토함으로써 순차적으로 개선된 실행가능해를 도출하는 방법. 그리디(greedy)방식이 여기에 해당하며 매 단계마다 목전의 이익추구를 계속하여 나감.

2. 局地的 개선법(local improvement method)

일단 하나의 실행가능해를 구하여 이를 최초해로 하고 미리 결정되어 있는 일련의 수정과정을 거쳐서 점진적으로 해를 개선하여 局地的 最適解(local optimum)에 도달할 때까지 반복계산하는 방법. 이 방법은 개선과정에서 일시적인 해의 악화도 허용하는 것이 단일경로 방식과는 다름.

3. 요소분석법(component analysis method)

문제의 규모가 너무 크고 복잡한 경우에 이를 분해하여 보다 다루기 쉬운 부문제를 만들고, 각 부문제에 대한 해를 순차적으로 구하여 이를 결합함으로써 전체문제의 해를 구하는 방법이다. 이때 분할된 부문제들간의 상호작용성을 잘 고려하여야 한다.

4. 학습방법 (learning method)

유사한 유형의 문제로부터 원리나 규칙을 추출하여 이를 보다 복잡한 문제의 해를 찾는 나무가지탐색과정상의 가지선택기준으로 삼는 방법.

5. 해구성법 (construction method)

문제의 자료를 검토하여 최종해의 한 구성요소로서 가치있다고 판단되는 요소들을 하나씩 추출하여 실행가능해를 완성시켜 가는 방법.

이러한 접근방법들은 상호배타적인 것이 아니기 때문에 실제 휴리스틱해법을 개발할 때는 이러한 접근방법중의 하나 이상을 결합하여 사용하게 된다.

### 3.2. 라그랑지안 이완기법

1970년대에 들어와서 이룩한 수학적 프로그래밍 분야의 중요 업적중의 하나는 라그랑지안이완기법의 발견이다. 라그랑지안이완기법은 어떤 정수 계획 문제가 일부 난해한 제약조건식 때문에 풀이가 어려워진 경우에 일부 제약조건식을 이완시켜 목적함수에 포함시킴으로써 비교적 풀이가 용이한 문제로 재구성하는 기법이다. 70년대 이전에도 정수계획 최적화 문제에서 라그랑지안 방법을 사용한 몇 가지 시도가 Lorie와 Savage(22), Everett(4)에 의해 이루어진 바 있으나, 본격적인 라그랑지안 접근방식의 탄생은 Held와 Karp(14, 15)가 최소걸침나무에 기초한 라그랑지안 문제를 사용하여 외판원 문제 (traveling salesman problem)에 대한 성공적인 해법을 극적으로 고안해낸 이후부터이다. Held와 Karp의 성공에 자극을 받아서 1970년대 초기에는 일정계획 (scheduling) 문제와 기타 일반 정수계획문제들에 대한 라그랑지안 방법의 성공적인 적용사례가 다수 개발되었다.

Geoffrion(10)은 그의 논문에서 이러한 접근방식들을 통칭하여 라그랑지안이완기법 (Lagrangian relaxation)이라고 하는 명칭을 부여하여 이것이 일반화되기에 이르렀다.

#### 3.2.1. 라그랑지안이완의 전개

우선 라그랑지안 이완의 개념을 설명하기 위하여 다음과 같은 정수계획식으로 공식화된 하나의 조합적 최적화 문제를 예를 들어 보기로 한다.

$$(P)\quad Z_{IP}=\min C_x$$

$$\text{s.t}\quad Ax \leq b$$

$$Dx \geq e$$

$$x \geq o \text{ and integer}$$

여기서 $x$는 $n\times 1$, $b$는 $m\times 1$, $e$는 $k\times 1$인 벡터이며 다른 모든 행렬(matrix)은 이에 상응하는 차원(dimension)을 가진다.

원문제 (P)의 제약조건식은 두 개의 제약조건식의 집합 $Ax \leq b$와 $Dx \geq e$로 나누어 지는데 $Ax \leq b$가 난해한 제약식(complicating constraints)이므로 이 제약식이 없다면 상대적으로 문제풀이가 쉬워진다고 가정하자.

원문제 (P)를 라그랑지안 문제로 변환시키기 위해서 $m$차원의 비음벡터(non negative vector)인 라그랑지안 승수(Lagrangian multipliers) $u$를 정의하여, 원문제의 목적함수에 陰의 함수식 $u(Ax-b)$을 추가한다.

$$(P)' \quad \min \; Cx+u(Ax-b)$$
$$\text{s.t.} \quad Ax \leq b$$
$$Dx \geq e$$
$$x \geq o \text{ and integer}$$
$$u \geq o$$

이 문제의 최적목적함수 값은 원문제의 최적해 $Z_{IP}$에 대해서 하한가(lower bound)가 된다. 왜냐하면 원문제의 목적함수에 단순히 음의 함수식만 추가하였고 제약조건식은 변화가 없기 때문이다.

여기서 제약조건식 중에서 $Ax \leq b$를 제거하면 라그랑지안 문제 (LRu)가 만들어진다.

$$(\text{LRu} \quad Z_D(u) = \min \; Cx+u(Ax-b)$$
$$Dx \geq e$$
$$x \geq o \text{ and integer}$$
$$u \geq o$$

제약조건식 $Ax \leq b$를 제거하면 (LRu)의 최적목적함수 값이 어떤 경우에도 (P)′ 최적목적함수값 이하로 내려오므로 $Z_D(u)$는 역시 $Z_{IP}$에 대한 하한가가 된다. 그리고 가정에 의해서 라그랑지안 문제는 원문제 보다 상대적으로 풀이가 용이한 문제가 된다.

3.2.2. 최적 라그랑지안 승수의 탐색

다음 $Z_D(u)$가 $Z_{IP}$에 가장 가까운 하한가(tight lower bound)가 되도록 하는 라그랑지안 승수 $u$를 찾는 문제를 고려해 보기로 한다. 최상의 승수 $u$를 선택하는 문제는 다음과 같은 쌍대문제 $(D)$의 최적해를 구하는 것이다.

$$(D) \quad Z_D = \max_{u} \; Z_D(u)$$

앞에서 $X=\{x \mid Dx \leq e, \; x \geq 0$이며 整數$\}$는 문제(LRu)의 실행가능해의 집합이며 유한하다고 가정하였다.

따라서 $X$는 $X=\{x^t, \; t=1, 2, \cdots, T\}$로 나타내는 것도 가능하다.

이것을 이용하여 문제 $(D)$를 다음과 같은 다수제약조건식을 가지는 선형계획문제$(\bar{D})$로 표현할 수 있다.

$$(\bar{D})\quad Z_D=\max\ w$$

$$\text{s.t}\quad w\leq Cx^t+u(Ax^t-b),\quad t=1,2,\cdots,T$$

$$w:\text{자유변수},\quad u\geq 0$$

이것은 $\text{Max}\ Z_D(u)=\text{Max}\{\text{Min}\ Cx+u(Ax-b)\}$이고 $w=\min\ Cx+u(Ax-b)$임을 생각하면 쉽게 이해할 수 있다.

문제 $(\bar{D})$의 쌍대문제는 다수의 列(columns)을 가지는 또 하나의 일반선형계획문제 $(\bar{P})$가 된다.

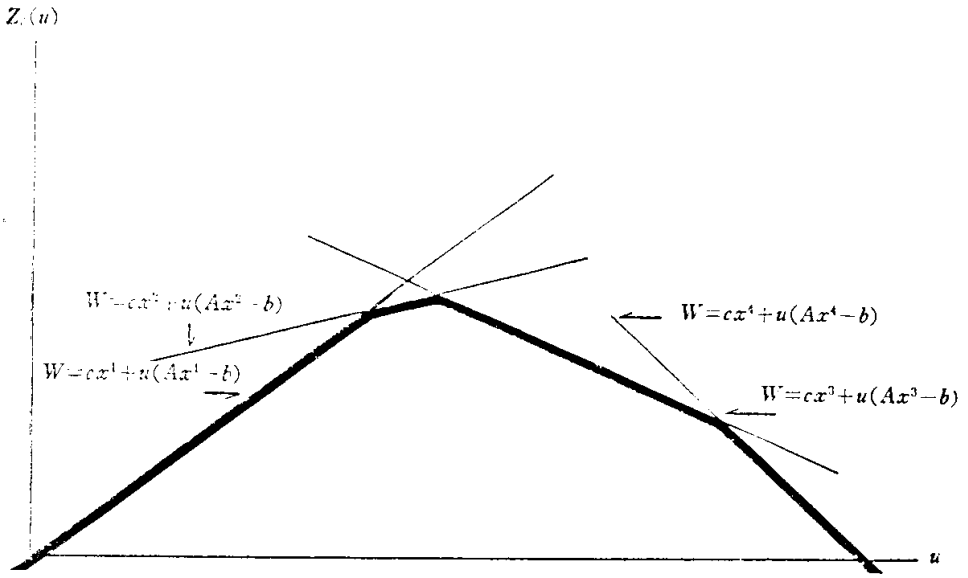
$$(\bar{P})\quad Z_D=\min_{\lambda}\sum_{t=1}^{T}\lambda_t\ Cx^t$$

$$\text{s.t}\sum_{t=1}^{T}\lambda_t\ Ax^t=b$$

$$\sum_{t=1}^{T}\lambda_t=1$$

$$\lambda_t\geq 0,\quad t=1,2,\cdots,T$$

문제 $(\bar{P})$와 $(\bar{D})$는 문제 $(D)$에 대한 해법을 모색하는데 있어서 중요한 단서를 제공한다. 문제 $(\bar{D})$는 $Z_D(u)$가 유한개수의 선형함수들로 구성된 분할선형 오목함수(piecewise linear concave function)임을 나타낸다. 다음 〈그림 3-1〉은 $m=1$이고 $T=4$인 경우에 어떤 $Z_D(u)$




〈그림 3-1〉 $Z_D(u)$ 함수의 형태

의 형태를 나타내고 있다.

이와 같이 $Z_D(u)$ 함수는 연속성(continuity)과 오목성(concavity)의 속성을 가지고 있기 때문에, 하나의 예외인 미분가능성(differentiability)의 문제만 해결되면 상승해법(hill-climbing algorithm)을 쉽게 적용할 수 있게 된다. $Z_D(u)$ 함수는 대부분의 지역에서 미분가능하지만, 최적해에서는 일반적으로 미분불가능하다. 그러므로 최적 라그랑지안 승수 $u$를 구하기 위해서는 미분가능함수에 적용하는 그래디언트 기법 대신에 이 방법을 다소 수정한 서브그래디언트 기법을 사용하여야 한다.

### 3.3. 서브그래디언트의 개념

서브그래디언트 기법은 라그랑지안이완기법과 분단탐색법을 연결하는 수리계획 분야에서 가장 활용도가 높은 도구 중의 하나다.

다음 문제를 예시하여 서브그래디언트의 개념을 설명하기로 한다.

$$W=\max f(y)$$

$$\text{s.t. } y\in Y\subseteq R^m$$

여기서 함수 $f$는 모든 $y$에 대해서 정의되는 오목실수이지만 반드시 모든 점에서 미분가능할 필요는 없는 어떤 함수로 가정한다. 이때 하나의 벡터 $r$가 다음 조건을 만족시키면 이 벡터를 함수 $f$의 $\bar{y}$에서의 서브그래디언트라고 한다.

$$f(y)\leq f(\bar{y})+r(y-\bar{y}) \text{ for all } y\in Y$$

그리고 $\bar{y}$에서의 서브디프렌셜(subdifferential) $\partial f(\bar{y})$는 $\bar{y}$에서 함수 $f$의 모든 서브그래디언트의 집합이라고 정의한다.

즉 $\partial f(\bar{y})=\{r\in R^m : f(y)\leq f(\bar{y})+r(y-\bar{y}) \text{ for all } y\}$이다.

여기서 서브디프렌셜 $\partial f(\bar{y})$는 다음과 같은 성질을 가진다.

(i) $\partial f(\bar{y})$는 폐쇄 볼록 집합(closed convex set)이며 $m$차원 공간 $R^m$의 부분집합이다(subset of $R^m$).

(ii) $\partial f(\bar{y})$는 함수 $f$가 모든 지역에서 유한(finite)하다면 모든 $\bar{y}\in R^m$에 대해서 항상 존재한다(non-empty).

(iii) 함수 $f$가 $\bar{y}$에서 미분가능하다면 (differential), $\bar{y}$에서의 서브디프렌셜은 $\bar{y}$에서의 그래디언트와 일치한다.

즉 $\partial f(\bar{y})=\nabla f(\bar{y})$

(iv) 함수 $f$는 $\bar{y}$에서 0(zero)이 서브디프렌셜에 포함될 때 $\bar{y}$에서 최대화된다. ($f$ is maximized at $\bar{y}$ if $0\in\partial f(\bar{y})$)

이와 같은 서브디프렌셜과 서브그래디언트의 성질에 기초하여 Poljak(25, 26)은 다음과 같은 定理를 제시하였다.

(定理 1) $y^0 \in Y$인 $y^0$에서부터 출발하여 일련의 과정 $y^{k+1}=y^k+t_k.r^k$ (여기서 $t_k=\frac{\|r^k\|}{\lambda^k}$, $\lambda^k \geq 0$이며 $r^k$는 $y^k$에서의 서브그래디언트이다)을 거치고, $\lambda^k \to 0$, $\sum_{k=0}^{\infty} \lambda_k = \infty$의 두 조건만 만족시킨다면 $n \to \infty$일 때 $y^n \to y^*$ (최적해)가 성립한다. 즉 $y^n$은 최적해에 수렴한다.

上記 定理에서 $t_k$의 결정방식을 수정하여 1969년에 定理 2를 발표하였다.

(定理 2) $t_k=\lambda_k.\frac{f^*-f(y^k)}{\|r^k\|^2}$이고 (이때 $f^*$는 실행가능해 중에서 최선의 목적 함수값이며 $f(y^k)$는 $y^k$에서의 목적함수값), $0<\varepsilon_1 \leq \lambda_k \leq 2-\varepsilon_2$, $\varepsilon_2>0$일 때 $n \to \infty$이면 $y^n \to y^*$(최적해)이고 $f(y^n) \to Z$(최적해에서의 목적함수값)로 수렴한다.

### 3.4. 서브그래디언트와 라그랑지안이완의 결합

이와 같은 서브그래디언트의 일반적인 이론이 라그랑지안이완기법과 결부되어 어떻게 사용되는가를 보기 위하여 앞절에서 언급한 문제 (LRu)의 최적 목적함수값의 함수 $Z_D(u)$를 다시 살펴보기로 한다.

다음 수식이 만족되면 $m$차원 벡터 $r$는 함수 $Z_D(u)$의 $\bar{u}$에서의 서브그래디언트가 된다.

$$Z_D(u) \leq Z_D(\bar{u}) + r(u-\bar{u}) \text{ for all } u$$

여기서 $Z_D(u)$는 모든 지역에서 서브디프렌셜을 가짐은 자명하다. 그리고 〈그림 3-1〉에서 나타난 바와 같이 문제 (LRu)의 최적해 $x^t$에 대해서, 벡터$(Ax^t-b)$는 어떤 $u$값에 대한 서브그래디언트가 된다. 함수 $Z_D(u)$에 대한 다른 모든 서브그래디언트는 이러한 원시 서브그래디언트들(primitive subgradients)의 볼록결합(convex combination)이다. 이러한 관점에서, $u^*$와 $\lambda^*$가 실행가능하고 상보여유 조건(complementary slackness condition)을 만족시킨다면 각각 문제$(\bar{D})$와 $(P)$에 대한 최적해가 되고, 이 결과는 다시 0(zero)이 $u^*$에서 $Z_D(u)$의 서브그래디언트중의 하나에 속하면 $u^*$는 문제 $(D)$에서 최적이 됨을 의미한다. 그러므로 이와 같은 속성을 가지는 문제 $(D)$의 최적해를 구하기 위해서 Poljak의 정리에 따른 서브그래디언트기법을 이용하면 그 절차는 다음과 같이 진행된다.

$u^0$를 연속점 $\{u^k\}$의 초기치라고 가정하면 연속점 $\{u^k\}$는 다음 규칙에 의해 생성된다.

$$u^{k+1}=u^k+t_k(Ax^k-b)$$

여기서 $x^k$는 (LRu$^k$)에 대한 최적해이며, $t^k$는 양수의 스칼라 값을 가지는 전진거리(step size)를 나타낸다. 그리고 서브그래디언트$(Ax^k-b)$는 방향벡터를 의미한다. 그러므로 서브그래디언트 기법도 그래디언트 기법과 마찬가지로 목적함수값을 증가시키는 방향벡터와 적

당한 전진거리를 구하여 최적해에 도달하고자 하는 일종의 상승해법(ascent algorithm)이다. 그러나 서브그래디언트 기법에서는 목적함수값이 매 반복계산마다 반드시 증가하지는 않기 때문에 이를 준상승해법(approximate ascent algorithm)이라고도 한다. 전진거리 $t^k$는 Poljak의 정리에 따라 다음과 같이 구해진다.

$$t^k = \frac{\lambda_k(Z^* - Z_D(u^k))}{\|Ax^k - b\|^2}$$

여기서 $Z^*$는 원문제 $(P)$의 최적목적함수값 $Z_{IP}$의 近似値로서, $Z_{IP}$에 대한 상한가가 되며, 일반적으로 휴리스틱해법으로 구한다. 서브그래디언트기법의 효율성은 어느 정도 최적해와 가까운 $Z^*$를 구할 수 있느냐에 많이 의존한다. $\lambda_k$는 사용자가 결정하는 스칼라값으로서 이 값은 일정수의 반복계산(보통 10~20번 사이) 동안 $Z_D(u)$의 개선이 이루어지지 않았을 때마다 1/2로 줄여 나간다. 이 공식에 대한 타당성은 Held, Wolfe와 Crowder(13)에 의해 입증된 바 있다.

이와 같이 라그랑지안 승수 $u$의 값을 계속적인 반복계산을 통하여 개선시켜 나감으로써 원문제$(P)$의 최적해 $Z_{IP}$에 도달하거나 아니면 최소한 선형계획이완의 목적함수값 $Z_{LP}$보다는 더 최적해에 가까운 (엄밀한) 하한가 $Z_D$를 도출하게 된다.

이와 같은 반복계산에 대한 정지조건은 다음 네 가지 경우에 도달했을 때이다.

a) $Z^* = Z_D(u^k)$이면 $(P)$의 최적해에 도달(상한가=하한가이므로)

$$Z_D = Z^* = Z_D(u^k)$$

b) $Ax^k - b = 0$이면 $(P)$의 최적해 도달

즉 $Z = Cx^* \geq Cx^* + u(Ax^* - b) = Z_D(u)$

따라서 $Z = Z_D(u)$

c) $Z_D(u^k)$가 어떤 값 $Z_D$에 수렴할 때

d) 반복횟수가 일정한계(보통 150번 이하)를 초과할 때

로서 a), b)는 최적해에 도달한 경우이고 c), d)의 경우는 최적해에 도달했다는 보장이 없다. 그리고 여기서 도출된 $Z_D$는 선형계획이완기법에 의한 목적함수값 $Z_{LP}$에 대해서 $Z_D \geq Z_{LP}$가 성립함은 Geoffrion(10)에 의해서 다음과 같이 증명된 바 있다.

$$Z_D = \max_{u \geq 0} Z_D(u)$$

$$= \max_{u \geq 0} \{\min_{x} (Cx + u(Ax - b))\}$$

$$Dx \geq e$$

$$x \geq 0 \text{ 이며 정수}$$

$$\geq \max_{u\geq 0}\{\min_{x}(Cx+u(Ax-b))\} \quad (\text{선형이완})$$

$$Dx\geq e$$

$$x\geq 0$$

$$=\max_{u\geq 0}\{\min_{x}(C+uA)x-ub\}$$

$$Dx\geq e$$

$$x\geq 0$$

$$=\max_{u\geq 0}\{\max_{v\geq 0}(ve-ub)\} \quad (\text{쌍대문제})$$

$$VD\leq C+uA$$

$$=\max_{u,v\geq 0}(ve-ub)$$

$$VD-uA\leq c$$

$$=\min\ Cx \quad (\text{쌍대문제})$$

$$Ax\leq b$$

$$Dx\geq e$$

$$x\geq 0$$

$$=Z_{LP}$$

그러므로 라그랑지안이완기법과 서브그래디언트기법에 의해 도출된 목적함수값은 원문제의 최적해에 대해서 선형계획이완기법에 의한 목적함수값보다 항상 우수한(tight) 下限價를 제공하여 줌으로써 분단탐색법과 결합하여 보다 효율적인 해법의 개발을 가능케 한다.

### 3.5. 분단탐색법

휴리스틱해법에 의해 도출된 최상의 上限價와 라그랑지안弛緩 및 서브그래디언트기법에 의해 도출된 가장 엄밀한 下限價를 분단탐색에 편입시켜 최적해를 구하는 일반적인 절차는 다음과 같다.

#### 3.5.1. 上限價와 下限價 및 라그랑지안 乘數의 計算

分段探索法을 적용하였을때 최적마디에서 이루어지는 上限價 및 下限價의 계산 절차는 아래 순서로 진행된다.

1) 휴리스틱 절차를 사용하여 원문제 (P)에 대한 實行可能解를 구하고, 이 값을 서브그래디언트 절차에서 요구되는 최적해에 대한 上限價 $z$로 사용한다. 라그랑지안 乘數값을 초기화시킨다.

2) 라그랑지안 문제에 대한 최적해를 계산한다.

3) 下限價가 (上限價−1)을 초과하면 그 上限價가 바로 최적해이므로 계산을 마침. 아

니면 다음 단계로 계속됨.

4) 서브그래디언트 절차에 의거하여 라그랑지안 乘數 $u$를 갱신한다.

5) 단계 2에서 단계 4까지 더이상 上限價의 값이 개선되지 않을 때까지 반복.

6) 최종 라그랑지안문제에 대한 敏感度 分析(sensitivity analysis)을 실시하여, 일부 가능한 변수 $x_{ij}=0$이나 1로 고정시켜 문제의 규모를 줄여본다.

7) 최종 라그랑지안 문제에 대한 최적해를 분석하여 대부분의 작업이 어느 한 기계에만 배정되고, 일부작업만 배정이 되지 않거나 혹은 두 대 이상의 기계에 중복배정이 된 경우, 이것을 각 기계당 한 개의 작업만 배정되게끔 내장된 휴리스틱 절차에 의해 새로운 上限價의 도출을 시도해 본다. 만약 개선된 새로운 上限價가 나온다면 $z$를 갱신(update) 한다.

8) 下限價와 實行可能解가 일치하면 ($Z_{LR(u^*)}>\bar{Z}-1$) 반복계산을 종료시킨다. 아니면 최적해를 확인하기 위한 分段探索法상의 새로운 마디로 진행한다.

3.5.2. 反復計算 過程

최초마디에서 최적해가 발견되지 않으면 從優先(depth-first) 分段探索法 절차를 사용하여 하위마디에서 최적해를 탐색한다.

이 절차는 다음과 같은 과정을 거친다.

분단탐색나무의 모든 단계에서 하나의 分離變數(separation variable)가 선택되고, 두 개의 下位問題가 생성된다. 하나의 하위문제는 分離變數가 0으로 고정되고, 다른 하위문제는 分離變數가 1로 고정된다. 각 하위문제마다 제한된 횟수의 서브그래디언트 반복계산이 수행되고, 제한된 문제(restricted problem)에 대한 下限價의 개선을 모색한다.

分技(branching)를 위한 다음 마디는 두 개의 하위문제중 가장 낮은 下限價를 가지는 마디가 선택된다. 마디 삭제가 발생한 경우, 방금 삭제된 마디로 내려오는 가지 경로를 따라서, 거슬러 올라가 가장 낮은 단계(level)의 마디까지 逆追跡(backtracking) 절차를 수행한다.

## Ⅳ. 難解性 組合的 最適化問題의 最適解法 構成事例

이 장에서는 전형적인 난해성 조합적 최적화문제인 병렬처리기계문제를 事例問題로 하여 최적해탐색을 위한 제반 방법론들을 적용해보고 그 효과를 측정해 본다.

### 4.1. 竝列處理機械問題의 定義 및 模型樹立

竝列處理機械상에서 總作業完了時間의 最小化문제에 대한 보다 명확한 定義를 위하여 어

떤 시스템 내에 $n$개의 작업($i=1,\cdots,n$)과 $m$대의 기계($j=1,\cdots,m$)가 대기하고 있다고 가정하고 작업 $i$가 기계 $j$에 배정되었을 때의 가공시간을 $t_{ij}$라고 한다.

$P_j(j=1,\cdots m)$를 $j$번째 기계에 배정된 작업 $i$들의 집합이라 하면, 그 기계에 배정된 작업을 모두 가공완료하는 데 소요되는 시간은 $M_j=\sum_{i\in P_j} t_{ij}$가 된다. 이때 總作業完了時間은 $M=\max_j\{M_j\}$가 된다.

이와 같은 가정을 토대로 하여 이 문제에 대한 모형은 다음과 같은 整數計劃問題로 정립될 수 있다.

$$\text{(P)}\quad \min\ M \tag{1}$$

$$\text{s.t.}\quad \sum_{i=1}^{n} t_{ij}\, x_{ij}\le M \qquad \text{for } j=1,2,\cdots,m \tag{2}$$

$$\sum_{j=1}^{m} x_{ij}=1 \qquad \text{for } i=1,2,\cdots,n \tag{3}$$

$$x_{ij}=0 \text{ or } 1$$

여기서 첨자집합(index set)을 다음과 같이 정의한다.

$I=\{1,2,\cdots,n\}$ : 작업(job)들의 집합

$J=\{1,2,\cdots,m\}$ : 기계(processor)들의 집합

그리고 각 변수들은 다음과 같은 의미를 갖고 있다.

$x_{ij}=\begin{cases}1, & \text{만약 작업 } i\text{가 기계 } j\text{에서 가공된다면}\\ 0, & \text{그렇지 않으면}\end{cases}$

$t_{ij}$=작업 $i$의 기계 $j$에서의 가공시간

$M$=總作業完了時間

이 모형은 자원의 배분과정에서 어떤 처리기(processor), 즉 사람이나 기계 또는 어떤 부서에 부과되는 최대작업부하를 최소화하는 방향으로 자원을 할당해야 하는 경우의 문제이다.

### 4.2. 병렬처리기계문제의 휴리스틱해법 개발

병렬처리기계문제중에서 作業分割可能問題의 경우에는 기계의 同質性여부에 관계없이 최적해법이 존재하지만 作業分割不可能問題로 넘어오면 가장 간단한 형태인 等速機械 2대 이상인 문제부터 NP-complete에 속하는 문제가 되어 최적해를 구하는 多項式限定解法(polynomial bounded algorithm)을 개발할 수 있는 희망은 거의 없다. 그러므로 기존의 연구는 準最適解(near-optimal solution)를 구하는 휴리스틱해법의 개발과 더불어서 이 해법들의 成果分析을 통하여 주로 計算複雜度(computational complexity)를 낮추면서 計算解의 質을 높이고자 한 것들이다.

〈표 4-1〉 問題類型別 解法研究者

| 기계종류 | 作業分割可能(preemptive) | 作業分割不能(non-preemptive) |
|---|---|---|
| 等速機械 (equal) | McNaughton(최적해법) | Sahni(지수시간 최적해법)<br>Graham(LPT 휴리스틱)<br>Coffman(Multifit 휴리스틱)<br>Hochbaum과 Shmoys(多項近似解法) |
| 比例速機械 (uniform) | Horvath(최적해법)<br>Gonzalez와 Sahni(최적해법) | Hochbaum과 Shmoys(多項近似解法) |
| 異速機械 (unequal) | Lawler와 Labetoule(최적해법) | Ibarra와 Kim(A-E 휴리스틱)<br>De와 Morton(H휴리스틱) |

원문제 (P)에 해당하는 작업분할이 불가능한 이속병렬처리기계문제에 대한 총작업완료시간의 최소화문제에서 Ibarra와 Kim(18) 및 De와 Morton(3)의 기존 휴리스틱해법들은 다음 세가지 목표들중 일부 또는 전부를 고려하고 있다.

① 가능한한 여러 기계에 부과되는 작업의 負荷를 均等化시킨다.

② 마지막에 가공시간이 너무 긴 작업이 남게 되면 이 작업이 어느 기계에 배정되더라도 전체적으로 總作業完了時間이 길어지므로, 이와 같은 突出性(lumpiness)을 피하기 위해서 最長加工時間의 작업부터 우선적으로 처리한다.

③ 각 작업은 그 작업에 대해서 比較優位를 가지는 기계(소요시간이 가장 적은 기계)에 우선적으로 배정한다.

그러나 위 세가지 목표는 상호 배타적일 때가 많기 때문에, 이를 동시에 추구하기가 어려운 경우가 많이 나타난다. 따라서 이들 목표를 어떻게 적절히 조화시키느냐가 해법의 효율을 좌우한다고 볼 수 있다. 본 연구에서 제시하고자 하는 새로운 휴리스틱해법은 위 세가지 목표를 보다 효과적으로 절충하기 위해서 다음 정의를 이용하기로 한다.

(定義)

모든 未配定作業 $i$에 대해서 각 기계별로 部分作業完了時間(partial completion time)과 각 작업의 처리시간의 합을 계산하여, 이 합이 가장 짧은 기계(最善의 機械 : minimum machine)와 그 다음 짧은 기계(次善의 機械 : next minimum machine)를 선택하여 두 처리시간의 차이를 구한다. 이 차이를 작업 $i$의 後悔(regret)라고 할 때, 이 차이가 가장 큰 작업이 最大後悔作業(maximum regret job)이다.

새로운 휴리스틱은 이 정의에 입각하여 後悔(機會費用)를 최소화하는 방향으로 다음 작업을 선택하여 적정기계에 배정해 나감으로써 매 단계마다 部分作業完了時間의 증가분을

최소화시키는 일종의 近視眼的 휴리스틱(myopic heuristic) 접근방식을 취한다. 앞으로 새로운 휴리스틱을 휴리스틱 I로 명명한다.

이 해법의 초기단계에서는 最大後悔基準에 의해 다음 처리작업을 선택하여 部分作業完了時間의 증가분을 최소화시키고자 하는데, 部分作業完了時間의 증가분을 최소화시킨다는 것은 작업부하를 균등하게 한다는 첫번째 목표에 충실한 것이고, 동시에 最大後悔作業을 차기처리작업으로 선택한다는 것은 比較優位追求의 세번째 목표에 충실하다는 의미이다. 휴리스틱 I 에서 최장시간작업 우선처리목표의 추구는 해법의 제 3 단계에 이를 때까지 유보된다. 구체적으로 $\beta xn$개의 작업이 할당된 이후에 비로소 돌출성의 문제를 해결하고자 한다. 후기단계로 돌입하면 남은 작업들에 대해서 기계별 작업시간의 평균치를 기준으로 최장시간작업순서로 정렬하여, 작업부하의 균등화를 고려하면서 적정기계에 배정해 나간다. 그러므로 이 해법의 초기단계에서는 첫번째 목표와 세번째 목표를 동시에 추구하고 작업의 適正部分(the proportion of jobs)이 할당된 이후에는 두번째 목표와 첫번째 목표를 동시추구하게 된다는 점에서 다른 해법들에 비해 3가지 목표를 보다 균형있게 고려한다고 볼 수 있다.

휴리스틱 I

0. 여러가지 다른 $\beta$ 값에 대해서 제 1 단계부터 제 5 단계까지 반복적용하여 최상의 결과를 선택한다. $(0<\beta\leqq 1)$

1. 모든 기계 $j$에 대해서 部分作業完了時間 $S_j \leftarrow 0$으로 초기화시킨다.

2. 모든 미배정작업 $i, i\in T$에 대해서

   $K1_{ij}^{*}=\min_{j} \{S_j+t_{ij}\}$

   $K2_{ij}^{**}=\min_{j\neq j^{*}}\{S_j+t_{ij}\}$를 구하고

   $K_i=K2_{ij}^{**}-K1_{ij}^{*}$를 구한다.

   각 작업 $i$에 대해서 해당하는 $j^{*}$를 $j_{(i)}$로 규정함.

3. (a) $K=\max_{i\in T}\{K_i\}$를 발견한다.

   해당 $i$를 $i$로 정의한다.

   만약 max $K_i$가 둘 이상의 작업에서 동률이 발생하면

   $K1_{ij}^{*}$가 작은 작업 $i$부터 배정한다.

   (b) 작업 $i$를 기계 $j_{(i)}$에 배정함.

   $S_{j(i)} \leftarrow S_{j(i)}+t_{ij(i)}$

   $T \leftarrow T-\{i\}$로 갱신한다.

(c) 만약 $|T|>\beta \cdot n$이면 제 2 단계로 돌아간다.

아니면 제 4 단계로 간다.

4. 미배정된 모든 작업 $i, i\in T$에 대해서 $K_i=\frac{1}{m}\cdot\sum_{j=1}^{m} t_{ij}$를 계산하여, $K_i$를 내림차순으로 정렬하여, 각 작업에 새로운 순위를 부여한다.

5. 모든 미배정작업 $i\in T$에 대해서

(a) $\underset{j}{\text{Min}}\{S_j+T_{ij}\}$를 발견하여 해당기계 $j$를 $j_{(i)}$로 규정한다.

(b) 작업 $i$를 기계 $j_{(i)}$에 배정한다.

$S_{j(i)}\leftarrow S_{j(i)}+t_{ij(i)}$로 갱신한다.

(c) 미배정작업집합 $T$가 공집합이 되면($T=\phi$) 해법진행이 종료됨.

아니면 제 5 단계로 되돌아감.

### 4.3. 라그랑지안 下限價의 導出

原問題(IP)의 라그랑지안弛緩식은 어느 제약조건식을 弛緩시키느냐에 따라서 두 가지 방법을 고려해 볼 수 있다. 이 중에서 적정한 계산시간 이내에 최적해에 가장 가까운 下限價를 도출하는 기법을 선택하여 分段探索法에 편입시키고자 한다.

原問題(IP)의 제약식($\sum_{i=1}^{n} t_{ij}\ x_{ij}\leq M$ for all $j$)을 背囊制約式(knapsack constraints)라고 하고, 제약식($\sum_{j=1}^{m} x_{ij}=1$ for all $i$)을 割當制約式(assignment constraints)이라고 한다면 우선 背囊制約式을 弛緩시킨 라그랑지안 弛緩式을 (LR Ⅰ)이라고 하고, 割當制約式을 弛緩시킨 것을 (LR Ⅱ)라고 한다.

이때 (LR Ⅰ)는 제약조건식이 整數屬性(integrality property)을 가지는데, 이것은 제약조건식 $x_{ij}\in\{0,1\}$이 $0\leq x_{ij}\leq 1$로 대체되더라도 최적해에 영향을 미치지 않음을 의미한다. 또한 이것은 라그랑지안弛緩問題의 최적목적함수값 $Z_{LRI}$이 원문제의 선형계획이완문제의 최적목적함수값 $Z_{LP}$와 동일한 값을 가짐을 의미한다. 따라서 보다 엄밀한 하한가를 구하기 위해서는 할당제약식을 이완시키는 방식을 채택해야 한다.

割當制約式을 弛緩시켜 구성한 라그랑지안 문제(LR Ⅱ)는 다음과 같다.

$$(\text{LR}\,\text{Ⅱ})\quad Z_{\text{LR}\,\text{Ⅱ}}=\text{Min}\Big[M+\sum_{i=1}^{n} v_i\Big(\sum_{j=1}^{m} x_{ij}-1\Big)\Big]$$

$$=\text{Min}\Big[\sum_{i=1}^{n}\sum_{j=1}^{m} v_i\ x_{ij}+\Big(M-\sum_{i=1}^{n} v_i\Big)\Big]$$

$$\text{s.t.}\ \sum_{i=1}^{n} t_{ij}\ x_{ij}\leq M \qquad \forall\ j\in J$$

$$x_{ij}\in\{0,1\} \qquad \forall\ i\in I,\ j\in J$$

$$v_i, M \text{ 자유변수}$$

割當制約式을 이완시켜 본 결과, 이 이완문제가 多數制約式 0～1 背囊問題(multiple constraints 0～1 knapsack problem)가 되며, 이 문제 자체가 또 하나의 NP-complete 문제로서 나타났다. 최근 다수제약식 0～1 배낭문제의 最適解를 구하기 위한 해법의 연구가 Gavish와 Pirkul(8) 등에 의해서 이루어지고 있으나, 그것은 이 문제의 제약식을 다시 이완시켜 분단탐색을 시도하는 방법으로서 最適解를 구할 수 있는 문제의 규모도 제한되어 있고 막대한 계산시간이 소모된다. 그러므로 이 해법들을 이용하게 되면 下限價를 구하기 위해서 NP-complete인 原問題를 또 다른 NP-complete인 弛緩問題로 전환시키는데 불과하므로 적정계산시간 이내에 적절한 下限價를 구하기가 사실상 힘들어진다. 뿐만 아니라 이 문제의 제약식 우변의 값 $M$도 고정된 常數가 아니고 自由變數로 남아있으며 라그랑지안 乘數조차도 확정되어 있지 않기 때문에, 외견상으로 이 라그랑지안문제의 最適解(下限價)를 구한다는 것은 대단히 어려운 일로 보인다.

그러나 이완문제의 目的函數와 制約條件의 구조를 자세히 분석해 보면 이 문제가 여러 개의 單一制約式 0～1 背囊問題(single 0～1 knapsack problem)로 분해됨을 발견할 수 있다. 단일제약식 0～1 배낭문제는 Nauss(24)에 의해 변수 개수 약 10,000개까지 풀어 낼 수 있는 類似多項時間解法이 개발되어 있다. 그러므로 우변의 변수 $M$을 하나의 整數값에 고정시키고, 乘數를 서브그래디언트의 일정단계에서 고정시키면, Nauss의 해법을 이용하여 분해된 단일제약식 0～1 배낭문제들을 풀고, 이를 결합하면 주어진 $M$과 승수값에서의 이완문제의 최적해값을 구할 수 있다. 다음 과제는 $M$의 값을 변화시켜 이완문제의 최적해값의 증감을 계산하여 이 중에서 이완문제를 최소로 하는 $M$값을 결정하는 것이다. 이때 $M$의 범위가 크면 $M$의 정수값 하나하나에 대해서 상기과정을 반복해야 하므로 막대한 계산시간이 소모된다. $M$값의 가능한 범위를 최소한으로 축소시키기 위하여, $M$에 대한 上限價와 下限價를 구하여 이완문제의 제약조건식에 추가시켰다. $M$에 대한 上限價는 휴리스틱 I의 값을 이용하고, 下限價는 앞에서 언급한 背囊制約式弛緩 라그랑지안下限價를 이용하였다.

즉 배낭제약식이완문제(LR Ⅰ)의 최적해값인 $Z_{D(LRI)}$을 구하고 이 값을 $M$의 1차 下限價로 보아 $\underline{M}$라고 하고 휴리스틱으로 구한 上限價를 $\bar{M}$라 하여 $M$의 범위를 $\underline{M} \leq M \leq \bar{M}$로 한정하여 (LR Ⅱ) 문제에 추가시키기로 한다.

새로운 (LR Ⅱ)문제는 다음과 같다.

$$(\text{LR}\,\text{Ⅱ})\quad Z_{\text{LR}\,\text{Ⅱ}} = \text{Min}\left[M + \sum_{i=1}^{n} v_i \left(\sum_{j=1}^{m} x_{ij} - 1\right)\right]$$

$$= \text{Min}\left[\sum_{i=1}^{n}\sum_{j=1}^{m} v_i\, x_{ij} + \left(M - \sum_{i=1}^{n} v_i\right)\right]$$

$$\text{s.t.} \sum_{i=1}^{n} t_{ij}\, x_{ij} \leq M \quad \forall\ j \in J$$

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

$$\underline{M} \leq M \leq \bar{M},\ M\text{은 정수}$$

$$v_i \text{ 자유변수}$$

앞에서 설명한 바와 같이, 분해된 각 背囊問題의 최적해를 구하여 이를 總合(aggregate)한 결과가 주어진 $v_i$와 $M$에서의 최적해가 된다. 다음 $M$값을 下限價에서 上限價까지 整數單位로 증가시켜가며 각 $M$값에서의 최적해를 구하고, 이 중에서 최소값을 내는 해를 주어진 $v_i$에서의 최적해 $Z_{LR\,II}$로 삼는다. 그리고 서브그래디언트 절차에 따라서 $v_i$값을 변화시켜 가며 $Z_D(_{LR\,II})=\max v\{Z_{LR\,II}\}$를 구한다.

이 과정에 대한 단계별 알고리즘은 다음과 같다.

(단계 1) 문제구성

原問題의 制當制約式을 雙對變數 $v_i$로 弛緩시켜 LRⅡ식을 구성한다.

(단계 2) 초기화

$\underline{M}=Z_{D(LRI)}$

$\bar{M}=Z_H$

$v_i=0 \quad \forall\ i \in I$

$\lambda_1=2$

$M_1^1=1+\text{INT}[\underline{M}]$ if $\underline{M}$ is not integer, o.w. $M_1^1=\underline{M}$

여기서 INT는 整數化 函數임

(단계 3) $K$번째 반복계산시의 LRⅡ 풀이

a) (LRⅡ)를 제약조건식의 개수 만큼 분해하여 $m$개의 單一제약식 배낭문제로 만든다.

b) 각각의 單一背囊問題를 풀어 각 제약식의 최적해를 구한다.
(Nauss의 유사다항시간해법이용)

c) 이 결과를 총합하여 $Z_{LR\,II}$를 구한다.

d) $M$을 $M^k{}_1$에서 1씩 증가시켜 가면서 $\bar{M}$를 초과하기 전까지, 각 $M$에 대한 $Z_{LR\,II}$를 구하여 비교하고 이중 $M^k{}_R$에서 $Z_{LR\,II}$가 최소가 된다면 이 값은 주어진 $v^k{}_i$에서의 최적 목적함수값 $Z_{LR\,II}(v^k)$라 하고, 이때의 $\chi_{ij}=\chi_{ij}{}^k$, $M=M^k{}_R$이 최적해가 된다.

$$Z_{LR\,II}(v^k)=\min\left[\sum_{i=1}^{n}\sum_{j=1}^{m} v_i{}^k\ x_{ij}{}^k+(M_R{}^k-\sum_{i=1}^{n} v_i{}^k)\right]$$

〈단계 4) $K$번째 반복계산시의 方向벡터 $(Ax^k-b)$와 前進距離 $t_k$ 및 새로운 雙對變數 $v_i{}^{k+1}$을 구함.

$$Ax^k-b=\sum_{j=1}^{m} x_{ij}{}^k-1 \qquad A\ i=1,2,\cdots,n$$

$$t_k=\frac{\lambda_k(Z^*-Z_{LR\,II}(v_k))}{\|Ax^k-b\|^2}$$

따라서 $v_i{}^{k+1}=v_i{}^k+t_k\ (Ax^k-b)$

참고 : $Z^*$는 $Z_{LR\,II}$에 대한 上限價로서, 여기서는 $Z^*=Z_H$로 한다.

〈단계 5) λ값의 조정

$Z_{LR\,II}(v^k)$가 정해진 수의 반복계산(예 : 15번 반복계산)에서 그 값이 개선되지 않았다면

$\lambda_{k+1}=1/2\ \lambda_k$

〈단계 6) 다음 조건을 만족시키면 반복계산을 마친다.

(i) $Z_H=Z_{LR\,II}(v^k)\rightarrow$최적해 발견

(ii) $Ax^k-b=\sum_{j=1}^{m} x_{ij}{}^k-1=0\rightarrow$최적해 발견

(iii) 반복계산 횟수가 일정한계를 초과할 때

(예 : $k>150$)

(iv) $Z_{LR\,II}$가 어떤 수에 수렴할 때

(v) (i)~(iv)의 어느조건도 충족시키지 못하면 $k=k+1$이 되고 단계 3으로 돌아간다.

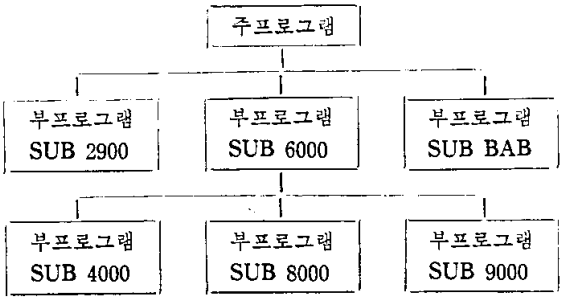
### 4.4. 분단탐색에 의한 최적해탐색

並列處理機械問題의 最適解를 구하기 위한 분산탐색 FORTRAN 프로그램은 다음 〈그림 4-1〉과 같이 구성된다.

### 4.5. 실험결과

CYBER-835를 이용하여 실험한 결과 이산적 일양분포 U(1, 100)인 문제에 대해서 다음 〈표 4-2〉와 같은 결과를 얻었다.

이 실험에서 變數의 개수가 80개 이하인 경우에는 거의 100% 최적해가 발견되나 그 이상 400개 이하인 경우에는 약 74% 정도 최적해가 발견됨을 파악하였다. 최적해가 발견된 문제중에서 라그랑지안 下限價와 초기 휴리스틱 上限價의 엄밀성으로 1차 分段探索(라그랑




1) 주프로그램 : 일반적인 분단탐색절차(마디의 생성과 탐색, 上下限價의 비교를 통한 최적해 확인)
2) 부프로그램 SUB 2900 : 휴리스틱 I (上限價 도출)
3) 부프로그램 SUB 6000 : (LR Ⅱ)에 기초한 라그랑지안 이완 및 서브그래디언트기법(下限價 도출)
4) 부프로그램 SUB 4000 : (LR I )에 기초한 라그랑지안弛緩 및 서브그래디언트기법(下限價의 초기화)
5) 부프로그램 SUB 8000~9000 : 단일제약식배낭문제의 풀이
6) 부프로그램 SUB BAB : 최초마디(root node)에서의 라그랑지안해를 중심으로 한 민감도분석(라그랑지안휴리스틱 분단탐색절차)

〈그림 4-1〉 主프로그램과 副프로그램의 관계

지안 휴리스틱)에 들어가기 전에 최적해가 발견된 경우는 28.6%이며, 1차 分段探索을 거쳐서 역시 초기휴리스틱해가 최적해로 확인된 경우까지 합치면 31.6%에 이른다. 반면에 초기휴리스틱해와 다른 새로운 최적해가 1차 分段探索을 거쳐 도출된 경우는 66.3%이다. 分段探索에 들어가는 경우에 1차 分段探索(라그랑지안 휴리스틱) 과정에서 탐색마디수 3,000개 이내에서 최적해가 발견되지 못하면, 2차 分段探索으로 넘어가게 되는데, 문제의 특성상 2차 分段探索에서는 마디별 下限價의 계산소요시간이 너무 크기 때문에 사실상 適正計算時間(2,000 CPU second) 이내에 최적해탐색이 불가능하였다. 그러나 이 결과는 De와 Morton(3)이 단순열거에 의존한 분단탐색방법으로 변수 40개(기계대수 4대 * 작업개수 10개)까지 밖에 풀지 못한 것과 비교하면 풀이가능한 문제의 규모를 상당히 확장시켰다고 볼 수 있다.

## Ⅴ. 結　論

본 연구에서는 난해성 조합적 최적화문제의 최적해를 구하는 일반적인 방법론을 모색하고자 하였다. 난해성 조합적 최적화문제는 일반적으로 이 문제들의 NP-complete 속성 때문에 다항시간 최적해법(polynomial-time exact algorithm)을 구한다는 것은 사실상 힘든 것으로 알려져 있다. 최근 여러가지 수학적 이론의 발전에 힘입어 지금까지 효율적인 최적해법을 구성하기 힘들었던 일부 난해성 조합적 최적화문제들에 대해서, 비록 제한된 규모이내

〈표 4-2〉 離散的一樣分布 U(1, 100) 實驗問題에 대한 最適解 探索結果

| 문제의 규모 | | a | b | c | d | e | f | 휴리스틱 上限價의 최적해와의 격차(%) | | | 최종라그랑지안 下限價의 최적해와의 격차(%) | | | 탐색마디수 | | | CUP 계산시간(단위 : 초) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 기계 대수 | 작업 개수 | | | | | | | 최소 | 평균 | 최대 | 최소 | 평균 | 최대 | 최소 | 평균 | 최대 | 최소 | 평균 | 최대 |
| 2 | 5 | 5 | 5 | 4 | 1 | 5 | 9 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1 | 1.2 | 2 | 3.51 | 5.24 | 29.674 |
| | 10 | 5 | 5 | 5 | 0 | 5 | 0 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 1 | 1.0 | 1 | 8.23 | 19.119 | 33.747 |
| | 15 | 5 | 5 | 2 | 3 | 2 | 3 | 0.00 | 0.81 | 2.27 | 0.00 | 0.00 | 0.00 | 1 | 1.6 | 2 | 21.463 | 69.510 | 114.582 |
| | 20 | 5 | 5 | 3 | 2 | 4 | 1 | 0.00 | 0.31 | 1.57 | 0.00 | 0.00 | 0.00 | 1 | 1.4 | 2 | 23.412 | 77.942 | 155.000 |
| 3 | 10 | 5 | 5 | 3 | 2 | 4 | 1 | 0.00 | 0.58 | 2.91 | 0.00 | 0.00 | 0.00 | 1 | 1.4 | 2 | 10.400 | 63.116 | 149.364 |
| | 15 | 5 | 5 | 3 | 2 | 4 | 1 | 0.00 | 0.11 | 0.56 | 0.00 | 0.00 | 0.00 | 1 | 1.4 | 2 | 15.129 | 32.736 | 47.148 |
| | 20 | 5 | 5 | 1 | 4 | 2 | 3 | 0.00 | 2.94 | 7.33 | 0.00 | 0.00 | 0.00 | 1 | 1.8 | 2 | 49.013 | 108.414 | 188.000 |
| | 25 | 5 | 5 | 1 | 4 | 1 | 4 | 0.00 | 2.93 | 6.57 | 0.00 | 0.42 | 1.06 | 1 | 2.8 | 7 | 37.644 | 161.781 | 241.492 |
| 4 | 15 | 5 | 5 | 1 | 4 | 1 | 4 | 0.00 | 2.75 | 5.62 | 0.00 | 0.43 | 1.81 | 1 | 1.8 | 2 | 22.227 | 70.670 | 120.526 |
| | 20 | 5 | 5 | 0 | 5 | 0 | 5 | 4.29 | 6.77 | 11.11 | 0.15 | 0.61 | 1.84 | 2 | 5.2 | 13 | 170.271 | 182.179 | 189.025 |
| | 25 | 5 | 4 | 0 | 4 | 0 | 4 | 2.14 | 5.81 | 9.93 | 0.00 | 0.48 | 1.51 | 2 | 7.0 | 17 | 204.081 | 233.620 | 277.013 |
| | 30 | 5 | 5 | 0 | 5 | 0 | 5 | 1.63 | 4.81 | 10.05 | 0.02 | 0.99 | 2.06 | 2 | 16.2 | 35 | 150.233 | 224.754 | 352.713 |
| 5 | 20 | 5 | 5 | 1 | 4 | 1 | 4 | 0.00 | 3.17 | 7.48 | 0.00 | 0.39 | 1.86 | 1 | 18.5 | 31 | 40.796 | 206.464 | 449.693 |
| | 25 | 5 | 5 | 1 | 4 | 1 | 4 | 0.00 | 5.90 | 14.47 | 0.00 | 0.36 | 1.26 | 1 | 9.25 | 31 | 106.507 | 206.524 | 330.626 |
| | 30 | 5 | 3 | 0 | 3 | 0 | 3 | 0.89 | 4.19 | 7.27 | 0.01 | 0.76 | 1.28 | 2 | 7.0 | 16 | 138.441 | 260.790 | 443.156 |
| | 35 | 5 | 4 | 1 | 3 | 1 | 3 | 0.00 | 4.27 | 6.42 | 0.01 | 0.20 | 0.77 | 1 | 236.5 | 409 | 151.891 | 380.178 | 583.693 |
| 8 | 25 | 5 | 3 | 0 | 3 | 0 | 3 | 3.92 | 7.85 | 11.62 | 0.02 | 0.18 | 0.47 | 11 | 32.0 | 85 | 233.333 | 343.275 | 438.534 |
| | 30 | 5 | 3 | 0 | 3 | 0 | 3 | 10.20 | 18.79 | 31.37 | 0.06 | 0.99 | 1.40 | 2 | 40.75 | 75 | 603.759 | 768.133 | 957.198 |
| | 35 | 5 | 2 | 0 | 2 | 0 | 2 | 13.33 | 15.27 | 17.21 | 0.08 | 0.55 | 1.01 | 67 | 124.50 | 182 | 567.165 | 695.090 | 823.015 |
| | 40 | 5 | 4 | 0 | 4 | 0 | 4 | 6.78 | 12.35 | 16.90 | 0.00 | 0.23 | 0.82 | 2 | 173.25 | 619 | 315.392 | 545.924 | 695.328 |
| 10 | 30 | 5 | 4 | 2 | 2 | 2 | 2 | 0.00 | 7.91 | 20.00 | 1.13 | 1.77 | 2.68 | 1 | 483.00 | 1,673 | 123.902 | 420.792 | 716.838 |
| | 35 | 5 | 3 | 0 | 3 | 0 | 3 | 7.89 | 10.73 | 14.29 | 0.01 | 0.58 | 1.08 | 3 | 29.0 | 81 | 349.827 | 527.340 | 660.493 |
| | 40 | 5 | 3 | 0 | 3 | 0 | 3 | 11.36 | 15.92 | 23.25 | 0.41 | 0.53 | 0.77 | 61 | 199.7 | 425 | 492.119 | 627.858 | 873.818 |
| | 50 | 5 | 0 | 0 | 0 | 0 | 0 | — | — | — | — | — | — | — | — | — | — | — | — |
| | | | | | | | | | 5.83 | | | 0.41 | | | | | | | |

(참고) a : 實驗問題의 수　　b : 최적해가 발견된 문제의 수　　c : 分段探索法에 들어가기전 최적해가 발견된 문제의 수
d : 1차 分段探索(라그랑지안 휴리스틱)에서 최적해가 발견된 문제의 수　　e : 최적해가 휴리스틱 上限價와 일치하는 경우의 수
f : 최적해가 휴리스틱 I 의 해와 다른 경우

에서나마 효율적인 최적해법의 구성에 성공한 바 있다. 그러나 이러한 최적해법들은 각 문제의 속성에 따라서 여러가지 다른 방법론들이 적용되었기 때문에 체계적인 통일성을 갖지 못하고, 문제마다 고유한 방법론을 개발해야 하는 것으로 인식되어 왔다.

본 논문에서는 이러한 방법론들의 체계화와 일반화를 시도해 보고자 지금까지 개발된 수학적기법들을 분석해 본 결과, 난해성 조합적 최적화문제의 최적해법 구성을 위해서는 분단탐색법을 기본적인 틀(framework)로 하여 최선의 실행가능해(상한가)를 구하는 적절한 휴리스틱해법의 개발과 최상의 초과최적해(하한가)를 구하는 라그랑지안이완과 분해기법 및 서브그래디언트기법을 적용하는 것이 가장 이상적인 방법론임을 밝히고 그 이론적 배경을 제시하였다.

그리고 이 방법론을 전형적인 NP-complete 문제인 병렬처리기계문제에 적용하여 효율적인 최적해법을 구성하는 과정을 사례로서 제시하였다.

본 연구의 한계로서는 아직도 효율적인 최적해법을 구하지 못한 다양한 NP-complete 문제들이 많이 존재하고 있기 때문에, 이러한 방법론을 적용해서 모든 NP-complete 문제에 대한 효율적인 최적해법을 구성할 수 있는지, 아니면 어떤 동질의 특징을 지닌 일부 유사한 문제집단에만 그것이 가능한지는 지금까지의 연구결과로서는 단정할 수 없다는 점이다. 이 문제는 향후 난해성 조합적 최적화문제들에 대한 보다 많은 후속연구에 의해서 밝혀질 수 있을 것이다.

이러한 한계에도 불구하고 본 연구에서 개발한 프로그램은 분단탐색에 기초한 일반적인 방법론의 원형을 포함하고 있기 때문에, 아직 효율적인 최적해법을 개발하지 못한 많은 난해성 조합적 최적화문제를 푸는 유익한 모형이 될 수 있을 것이다.

## 參 考 文 獻


1. Coffman, E., M.R. Garey and D.S. Johnson, "An Application of Bin Packing to Multiprocessor Scheduling," *SIAM J. Comput.*, 7, 1978, pp. 1-17.
2. Cook, S., "The Complexity of Theorem Proving Procedures," *Proc. 3rd Ann. ACM Symp. on Theory of Computing*, Association for Computing Machinery, New York, 1971, pp. 151-158.
3. De, P. and T.E. Morton, "Scheduling to Minimize Makespan on Unequal Parallel Processors," *J. ACM*, 26, 1979.

4. Everett, H., "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Res.*, Vol. 11, 1963, pp. 399-417.
5. Fisher, M.L. and Alexander H.G. Rinnooy Kan," The Design, Analysis and Implementation of Heuristics", *Management Science*, Vol. 34, No. 3, 1988, pp. 263-265.
6. Foulds, L.R., "The Heuristic Problem-Solving Approach," *J. of Operational Research Society*, Vol. 34, No. 10, pp. 927-934.
7. Garey, M. and David S. Johnson, *Computers and Intractability*, W.H. Freeman and Company, San Francisco, 1979, pp. 96-99.
8. Gavish, B. and Hasan Pirkul, "Efficient Algorithms for Solving Multiconstraint Zero-one Knapsack Problems to Optimality" *Mathematical Programming* 31, 1985.
9. Gavish, B. and Kizhanathan Srikanth, "An Optimal Solution Method for Large-scale Multiple Traveling Salesman Problems," *Operations Research*, Vol. 34, No. 5, Sep.-Oct., 1986, pp. 698-717.
10. Geoffrion, A., "Lagrangean Relaxation and its Uses in Integer Programming," *Math. Programming Study*, Vol. 2 (1974), pp. 82-114.
11. Gonzales, T. and S. Sahni, "Preemptive Scheduling of Uniform Processor," *J. of ACM*, Vol. 25, No. 1, Jan., 1978, pp. 92-101.
12. Graham, R., "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Applied Math*, 17, 1969, pp. 416-417.
13. Held, M., P. Wolfe and H.D. Crowder, "Validation of Subgradient Optimization," *Mathematical Programming*, Vol. 6, 1974, pp. 62-88.
14. Held, M. and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees: Part II", *Math. Programming* 6, 1971, pp. 62-88.
15. Held, M. and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees," *Operations Res.*, Vol. 18 (1970), pp. 1138-1162.
16. Hochbaum, D. and David B. Shmoys, "A Polynomial Approximation Scheme for Scheduling on Uniform Processors: Using the Dual Approximation Approach", *SIAM J. Comput.*, Vol. 17, No. 3, June, 1988.
17. Hochbaum, D. and David B. Shmoys, "Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results", *J. of ACM*, Vol. 34, No.

1, Jan. 1987, pp. 144-162.

18. Ibarra, O. and C.E. Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors," *J. of ACM*, Vol. 24, 1979, pp. 280-289.
19. Karp, R., "Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in Plane," *Math. Oper. Res.* 2, 1977, pp. 209-224.
20. Karp, R., "Reducibility among Combinatorial Problems," in R.E. Miller and J.W. Thatcher(eds.) ***Complexity of Computer Computations***, Plenum Press, New York, 1972, pp. 85-103.
21. Lawler, E. and J. Labetoule, "On Preemptive Scheduling of Unrelated Parallel Processors by Linear Programming", *J. ACM*, Vol. 25, No. 4, Oct., 1978, pp. 612-619.
22. Lorie, J. and L.J. Savage, "Three Problems in Capital Rationing," *J. Business*, 1955, pp. 229-239.
23. McNaughton, R., "Scheduling with Deadlines and Loss Functions," *Management Science*, Vol. 6, No. 1, 1959.
24. Nauss, R., "An Efficient Algorithm for the 0∼1 Knapsack Problem," *Management Science*, Vol. 23, No. 1, September, 1976.
25. Poljak, B., "A General Method of Solving Extremum Problems," *Soviet Math.* 8 (1967), pp. 593-597.
26. Poljak, B.,"Minimization of Unsmooth Functionals," *U.S.S.R. Computational Math. and Math. Phys.* 9(1969), pp. 14-29.
27. Sahni, S., "Algorithms for Scheduling Independent Tasks," *J. of ACM*, Vol. 23, No. 1, Jan. 1976, pp. 116-127.