Journal of the Korean
OR/MS Society
Vol. 16, No.2, Dec., 1991

並列處理機械상에서 總作業完了時間의 最小化解法에 관한 硏究

안상형* • 이송근**

A Study on Approximate and Exact Algorithms to Minimize Makespan on Parallel Processors

Sang Hyung Ahn* • Song Kun Lee**

Abstract

The purpose of this study is to develop an efficient exact algorithm for the problem of scheduling n independent jobs on m unequal parallel processors to minimize makespan. Efficient solutions are already known for the preemptive case. But for the non-preemptive case, this problem belongs to a set of strong NP-complete problems. Hence, it is unlikely that the polynomial time algorithm can be found. That is the reason why most investigations have been directed toward the fast approximate algorithms and the worst-case analysis of algorithms.

Recently, great advances have been made in mathematical theories regarding Lagrangean relaxation and the subgradient optimization procedure which updates the Lagrangean multipliers. By combining these mathematical tools with branch-and-bound procedures, there have been some successes in constructing pseudo-polynomial time algorithms for solving previously unsolved NP-complete problems. This study applies similar methodologies to the unequal parallel processor problem to find the efficient exact algorithm.

*서울대학교 경영학과

**대구대학교 경영학과

(이 논문은 서울대학교 1988년도 대학발전 연구기금에 의하여 연구되었음.)

1. 序 論

日程計劃(scheduling)을 수립하는 데 있어서의 기본적인 문제중의 하나는 竝列處理機械(parallel machine)상에서 總作業完了時間(makespan)을 最小化하는 문제이다. 여기서 竝列處理機械問題라 함은, 1회 가공(single operation)을 요하는 다수의 작업들이 시스템 내에 대기하고 있고 다수의 기계가 준비되어 있는 경우 각 작업들이 이 중 어느 기계를 통해서도 1회 가공으로 완료될 수 있는 狀況에서, 일정한 작업배정을 통하여 특정 成果尺度(measure of performance)[1]를 最小化하거나 最大化하는 경우의 문제를 의미한다.[2] 그리고 여기서 成果尺度가 되는 總作業完了時間(makespan)은 최초작업의 착수로부터 마지막 작업이 완료되는 시점까지의 加工所要時間을 일컫는다. 본 硏究는 특히 작업의 分割加工이 불가능하고 각 작업의 기계간 처리속도가 다른 異速竝列處理機械의 問題狀況에서 작업의 最適配分을 통한 總作業完了時間의 最小化를 달성할 수 있는 最適解法을 구성하고자 한다.

이 문제는 생산현장에서 일단의 作業群을 일단의 機械群에 효율적으로 배분하는 문제로 나타나기도 하고, 현재 개발이 추진 중인 多數의 處理機(processor)가 네트워크로 연결된 병렬컴퓨터를 이용하여 대규모 작업군을 처리할 때 負荷의 均等化(load balancing)를 추구하면서 최소시간내에 작업처리를 완료하기 위한 作業配分의 問題로 나타나기도 한다.[3] 지금까지 生産管理分野와 電算學分野에서 이에 대한 많은 연구가 진행되어 왔으나, 이 문제가 가지고 있는 기본적인 屬性, 즉 NP−completeness의 屬性[4]으로 말미암아 효과적으로 最適解를 구하기가 지극히 힘들기 때문에 대부분의 연구는 最適解의 근사치, 즉 準最適解를 구하는 휴리스틱해법의 개발이나 準最適解의 엄밀성을 분석하는 最惡限界分析(worst-case analysis), 平均行態分析(average behavior analysis), 確率的分析(probabilistic analysis)등이 주류를 이루고 있다.

作業分割이 불가능한 異速竝列處理機械問題의 最適解를 구해보려는 지금까지의 시도로서는 크기가 극히 작은 문제(기계대수4×작업개수10이하)에 대한 De와 Morton(1979)의 초보적인 列擧法이 있을 뿐이다. 본 연구에서는 보다 큰 규모의 문제에 대해 휴리스틱해의 最適解로부터 離脫程度를 평가하기 위한 比較基準으로서의 필

1) 일정계획(scheduling)에서 성과척도(measure of performance)는 특정 해법의 효율을 평가하는 기준으로 작업장의 성격과 목적에 따라 평균체류시간(mean flow time), 평균납기지연시간(mean tardiness), 납기지연 작업수(number of tardy job), 총작업완료시간(makespan)등의 여러 기준이 이용된다.

2) K.R.Baker, *Introduction to Sequencing and Scheduling*, New York, John Wiley and Sons, 1974, pp.114−115.

3) Eliezer Dekel and Sartaj Sahni, "Parallel Scheduling Algorithms," *Operations Research*, Operations Research, Vol.31, No. 1, Jan−Feb. 1983, pp.24−29.

4) M.R. Garey and D.S.Johnson, *Computers and Intractability*, San Francisco, W.H.Freeman and Company, 1979, p. 238.

요성과 아울러 실제 經營現場에서 부딪힐 수 있는 현실적인 크기의 문제에 대한 직접적인 最適解導出의 필요성을 감안하여 본 문제에 대한 효율적인 最適解法을 구성하였다.

異速竝列處理機械상에서 總作業完了時間의 最小化問題에 대한 最適解를 구하기 위한 기본적인 방법으로서 分段探索法을 이용한다. NP-complete문제를 分段探索法으로 해결하고자 할 때 가장 중요한 문제는 最適解에 근접하는 강력한 上限價(upper bound)와 下限價(lower bound)를 구하는 것이다. 이를 감안하여 분단탐색법의 틀(framework)속에서 최적해를 효율적으로 도출하기 위해서 硏究의 目的을 다음과 같이 설정하였다.

첫째, 지금까지 개발된 어떠한 휴리스틱해법보다 最適解와의 격차를 훨씬 줄일 수 있는 새로운 휴리스틱해법을 개발한다. 기존의 휴리스틱해법에서 추구해온 作業配定의 優先順位를 보다 적절히 조합하고 새로운 槪念에 입각한 優先順位를 추가함으로써 기존의 휴리스틱보다 平均的인 관점에서나 最惡限界의 관점에서 우수한 휴리스틱해법을 개발하여 최적해에 보다 근접한 上限價를 제공하고자 한다.

둘째, 分段探索法에서 탐색마디의 수를 효과적으로 줄여나가기 위해서는 上限價 뿐만 아니라 下限價도 엄밀해야 하므로 본 문제의 構造的 特性에 기초한 강력한 下限價를 구하는 방법을 모색한다. 이를 위해서 原問題에 대한 여러가지 弛緩問題를 구성하고 각 이완문제의 쌍대최적해를 비교분석하여, 이중에서 원문제의 최적해에 가장 근접하는 해를 도출하는 이완방식을 선택한다. 그리고 여기서 산출된 이완문제의 최적목적함수값을 下限價로 설정하여 분단탐색법에 편입시킨다.

셋째, 이 기법들을 分段探索法에 편입시켜 最適解를 구하는 해법을 구성하고, 이를 프로그램화하여 실제 현실문제에 적용할 수 있도록 한다.

2. 問題의 定義및 模型의 構成

2.1 問題의 槪要

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 어떤 시스템 내에 n개의 작업(i=1,……, n)과 m대의 기계(j=1,……, m)가 대기하고 있다고 가정하고 작업 i가 기계 j에 배정되었을 때의 가공시간을 t_{ij}라고 한다.

P_j(j=1,……, m)를 j번째 기계에 배정된 작업 i들의 집합이라 하면, 그 기계에 배정된 작업을 모두 가공완료하는 데 소요되는 시간은$M_j = \sum_{i \in Pj} t_{ij}$가 된다. 이때 總作業完了時間은 $M=\max\{M_j\}$가 된다. 이 과정을 예시하면 <표 2-1>과 같다.

결국 이 문제는 n개의 작업집합을 m대의 기계에 대한 部分集合(subset)으로 분할하되 總作業完了時間을 最小化하는 分割方式을 구하는 문제라고 할 수 있다. 이 문제의 경우의 수를 구하기 위해 n개의 작업을 $n_1, n_2, \cdots, n_m$개의 부분집합으로 분할한다면 다음과 같이 나타낼 수 있다.

표 2-1 작업시간행렬의 예

기계 \ 작업	1	2	3	4	5
1	77	18	91	89	39
2	25	14	19	79	72

$P_1 = \{2, 4\}$, $M_1 = 18 + 89 = 107$

$P_2 = \{1, 3, 5\}$ $M_2 = 25 + 19 + 72 = 116$

$\therefore M = \max\{M_1, M_2\} = \max\{107, 116\} = 116$

$n_1 + n_2 + n_3 + \cdots + n_m = n$

이 식을 만족시키는 각각의 경우에 대해서

$nCn_1 \cdot (n - n_1)Cn_2 \cdot (n - n_1 - n_2)Cn_3 \cdots$

$\cdots(n - n_1 - \cdots\cdots - n_{m-1})Cn_m$

의 경우를 따지는 m-partition 문제가 되므로 문제의 규모가 커짐에 따라서 경우의 수가 급격히 증가하여 최적해의 발견이 어렵게 된다.

2.2 問題의 類型

總作業完了時間의 最小化라는 목적은 동일하더라도 작업이나 기계의 特性에 따라서 다음과 같은 基準으로 문제의 類型을 구분할 수 있다.

(1) 작업의 先行構造(precedence structure)의 存在與否

a. 獨立作業(independent job) : 작업들 간에 先行作業과 後續作業이 존재하지 않는 경우

b. 從屬作業(dependent job) : 작업들 간에 先行作業과 從屬作業이 존재하는 경우

(2) 同質等速機械와 異質機械(homogeneous-equal machine and heterogeneous machine)와의 구분

a. 同質機械(等速機械:equal machine):각 작업의 가공시간이 어느 기계를 거쳐 가공되든지에 관계없이 일정한 경우($t_{ij}=t_i$)

b. 異質機械

- 比例速機械(uniform machine):모든 작업에 대해서 기계들 간의 상대적인 속도가 작업에 관계없이 비례적으로 정해지는 경우($t_{ij} = \alpha_i t_i$)
- 異速機械(general machine) : 각 작업에 대한 기계들 간의 상대적인 속도가 작업마다 전부 다른 경우

(3) 단위작업의 分割加工의 可能性 與否

a. 分割加工이 可能한 경우(preemptive case):임의의 작업을 어떤 기계에서 가공도중에 중단시켰다가 다른 기계에서 다시 그 나머지 부분을 가공하여 완료할

수 있는 경우

b. 分割加工이 不可能한 경우(non－preemptive case):분할가공이 허용되지 않는 경우

(4) 作業着手時點에 모든 작업의 시스템 到着與否

제로시점에 모든 작업이 시스템 내에 이미 도착해서 대기하고 있는 경우와 계속해서 하나씩 도착해 오는 경우로 구분할 수 있으나, 여기서는 모든 작업이 시스템 내에 대기하고 있고 어떠한 작업도 제로시점부터 가공이 가능하다고 가정한다.

(5) 作業準備時間(set－up time)

여기서 작업준비시간은 0으로 간주한다. 독립작업에 대해서는 작업준비시간을 가공시간에 포함시킬 수 있기 때문이다.

본 연구에서 다루고자 하는 문제는 다음과 같은 假定下의 問題로 그 범위를 한정하기로 한다.

표 2－2 연구문제의 假定과 範圍

機械의 類型	異速竝列處理機械
目的	總作業完了時間의 最小化
先行構造	獨立作業
分割加工의 可能與否	分割不可能
作業의 시스템 到着	0(제로)시점에 作業到着完了
作業準備時間	加工時間에 포함

3.3 模型의 樹立

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

$$(P)\ \text{Min} \quad M \qquad \cdots (2-1)$$

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

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

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

여기서 첨자집합(index set)을 다음과 같이 정의한다.

I={1, 2, …, n} : 작업(job)들의 집합

J={1, 2, …, m} : 기계(processor)들의 집합

그리고 각 변수들은 다음과 같은 의미를 갖고 있다.

x_{ij} = 1 만약 작업 i가 기계 j에서 가공된다면

0 그렇지 않으면

t_{ij} = 작업 i의 기계 j에서의 가공시간

M = 總作業完了時間

모형의 목적함수식 (2-1)은 각 기계에서 할당된 작업을 처리하는 데 소요되는 시간이 가장 큰 기계에서의 소요시간을 最小化하는 것을 나타낸다. 제약조건식(2-2)는 모든 기계에서는 總作業完了時間의 범위 내에서 할당된 모든 작업들의 처리를 완료해야 함을 의미한다. 제약조건식(2-3)은 하나의 개별작업은 반드시 어느 한 대의 기계를 통해서만 가공이 되어야 함을 의미한다. 이 조건은 작업분할이 허용되지 않는 경우에는 타당하지만, 작업분할이 허용되는 경우에는 x_{ij}가 반드시 0 이나 1로 고정될 필요가 없다. 이때 x_{ij}는 각 기계에서의 가공비율을 나타낸다. 따라서 작업분할이 허용되는 경우에도 제약조건식(2-3)은 여전히 유효하고, 다만 x_{ij}의 整數條件만 완화시킨 셈이 된다.

3. 휴리스틱解法의 開發

3.1 휴리스틱解法의 構造

병렬처리기계문제중에서 作業分割可能問題의 경우에는 기계의 同質性여부에 관계없이 최적해법이 존재하지만 作業分割不能問題로 넘어오면 가장 간단한 형태인 等速機械 2대이상인 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

표 3-1 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) Graham(LPT 휴리스틱) Coffman(Multifit 휴리스틱) Hochbaum 과 Shmoys(多項近似解法)
比例速機械 (uniform)	Horvath(최적해법) Gonzalez 와 Sahni(최적해법)	Hochbaum 과 Shmoys(多項近似解法)
異速機械 (unequal)	Lawler 와 Labetoule(최적해법)	Ibarra 와 Kim (A-E 휴리스틱) De 와 Morton (H 휴리스틱)

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.

작업분할이 불가능한 이속병렬처리기계문제에 대한 총작업완료시간의 최소화문제에서 Ibarra와 Kim(1979)및 De 와 Morton(1979)의 기존 휴리스틱해법들은 다음 세 가지 목표들 중 일부 또는 전부를 고려하고 있다.

① 가능한 한 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 세 가지 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 세 가지 목표를 보다 효과적으로 절충하기 위해서 다음 정의를 이용하기로 한다.

(定義)

모든 未配定作業 i에 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 i의 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 I에서 최장시간작업 우선처리목표의 추구는 해법의 제3단계에 이를 때까지 유보된다. 구체적으로 βxn개의 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 첫

번째 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

○. 여러가지 다른 β값에 대해서 제1단계부터 제5단계까지 반복적용하여 최상의 결과를 선택한다.($0 < \beta <= 1$)

1. 모든 기계 j에 대해서 部分作業完了時間 $S_j \leftarrow 0$으로 초기화시킨다.
2. 모든 미배정작업 i, $i \in T$에 대해서

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

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

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

   각 작업 i에 대해서 해당하는 j*를 $j_{(i)}$로 규정함.
3. (a) $K = \max_{i \in T}\{K_i\}$를 발견한다.

   해당 i를 $\bar{i}$로 정의한다.

   만약 max K_i가 둘 이상의 작업에서 동률이 발생하면 $K1_{ij}^{*}$가 작은 작업 i부터 배정한다.

   (b) 작업 $\bar{i}$를 기계 $j_{(\bar{i})}$에 배정함

   $S_{j_{(\bar{i})}} + S_{j_{(\bar{i})}} + t_{\bar{i}j_{(\bar{i})}}$

   $T \leftarrow T - \{\bar{i}\}$로 갱신한다.

   (c) 만약 $|T| > \beta \cdot n$이면 제2단계로 돌아간다.

   아니면 제4단계로 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_i를 내림차순으로 정렬하여, 각 작업에 새로운 순위를 부여한다.
5. 모든 미배정작업 $i \in T$에 대해서

   (a) $\min_{j}\{S_j + t_{ij}\}$를 발견하여 해당기계 j를 $j_{(i)}$로 규정한다.

   (b) 작업 i를 기계$j_{(i)}$에 배정한다.

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

   (c) 미배정작업집합 T가 공집합이 되면($T = \emptyset$) 해법진행이 종료됨.

   아니면 제5단계로 되돌아감.

여기서 휴리스틱 I의 단계별 과정을 세부적으로 설명한다.

(0단계) T를 현재까지의 未配定作業들의 집합이라고 한다. β는 次期配定作業을 선택하는 優先順位를 결정하는 데 있어서, 最大後悔에 입각한 선택에서 最長作業時間優先(Longest Processing Time First : LPT)에 입각한 선택으로 전환하는 分岐點을 결정하는 기준으로서, 전체 작업개수를 n개라고 할 때, 旣配定作業數가 $\beta \times n$에 이르기 전까지는 최대후회에 입각한 선택을 하며, $\beta \times n$개를 초과한 후부터는 LPT에 입각한 차기작업의 선택이 이루어지도록 한다. 일반적으로 β는 0.1에서 1까지 0.1단위로 점증시키면서 휴리스틱의 과정을 적용한 결과를 비교하고, 이중 최상의 결과를 휴리스틱 I의 해로 간주한다.

(1단계) 각 기계별로 현재까지의 기배정작업시간의 합을 部分作業完了時間 S_j라고 하고, 처

음에 이 값을 0으로 初期化시킨다.

(2단계) 모든 미배정작업 i에 대해서, 각 작업별로 현재까지의 기계별 부분작업완료시간 S_i와 작업 i의 기계 j에서의 작업소요시간 t_{ij}를 합한 $S_i + t_{ij}$를 계산해서 최소치를 나타내는 최선의 기계를 j*라고 하고 해당기계에서의 새로운 부분작업완료시간 $S_{i*} + t_{ij*}$를 $K1_{ij*}$라 한다. 또한 그 다음 최소치를 나타내는 차선의 기계를 j**라 하며, 해당기계에서의 새로운 부분작업완료시간을 $K2_{ij**}$라고 한다. 그러면 $K_i = K2_{ij**} - K1_{ij*}$는 각 작업별 후회(기회비용)값의 되며, 각 작업 i에 대한 最善의 기계 j*를 j(i)라고 한다.

(3단계)

(a) 모든 미배정작업중에서 후회값이 가장 큰 작업 i를 선택하여, 해당작업 i를 $\bar{i}$라고 한다.

(b) 작업 $\bar{i}$를 최선의 기계 $j(\bar{i})$에 배정한다.

이때 미배정작업의 집합 T는 한 개가 줄어든다.

(c) 미배정작업개수 T가 $\beta \times n$개 이상이면, 제2단계로 되돌아가고 아니면 제4단계로 넘어간다.

(4단계) 미배정작업집합 T에 남아있는 모든 작업 i에 대해서, 작업 i의 기계별 작업시간 t_{ij}를 합계해서 기계대수 m으로 나누면, 각 작업의 平均作業時間이 산출된다. 이 시간을 기준으로 내림차순으로 정리하여, 평균작업시간이 큰 작업부터 새로운 배정우선순위를 부여한다.

(5단계)

(a) 새로운 우선순위에 입각해서, 각 작업별로 새로운 부분작업완료시간 $S_i + t_{ij}$를 계산해서, $S_i + t_{ij}$가 최소인 기계 j를 j(i)로 한다.

(b) 작업 i를 기계 j(i)에 배정한다.

(c) 이 과정을 마지막 남은 작업까지 반복한다.

휴리스틱 I의 計算複雜度에 대해서 다음 定理를 제시한다.

(定理)

휴리스틱 I의 計算複雜度(time complexity)는 $0(kmn^2 + nlogn)$이다.

(證明)

제 1 단계 : $0(m)$

제 2, 제 3 단계 : $0(mn_1^2)$

여기서 n_1은 $n_1 \leq \beta n$을 만족시키는 最大整數

제 4, 제 5 단계 : $n_2 = n - n_1$이라면 $0(n_2 \log n_2 + n_2 m)$

따라서 1회 반복시의 계산의 복잡도는 $= 0(mn_1^2 + n_2 \log n_2 + n_2 m) \leq 0(mn^2 + nlogn)$, (여기서 $n_1, n_2 < n$)

k회 반복계산이 있었다면 $0(k(mn^2 + nlogn))$

실제로 각 β에 대해서 매 반복시마다 순서분류결정(sorting)을 할 필요가 없으므로 最終計算複雜度는 $0(kmn^2 + nlogn)$이다.

휴리스틱 I의 計算複雜度는 Ibarra와 Kim의 휴리스틱보다는 크고 De 와 Morton의 휴리스틱과는 동등한 정도이다. 그러나 기본적으로 계산시간이 多項式限定時間(polynomial time)의 범위이내이므로 컴퓨터 CPU 시간상으로 큰 차이가 없다고 볼 수 있다.

표 3-2 휴리스틱해법들의 最適解와의 比率隔差 比較

실험문제		B		C		D		E		F		G		H		I		최적
기계 대수	작업 개수	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b	해
2	5	110	2.80	110	2.80	107	0.00	110	2.80	110	2.80	107	0.00	110	2.80	107	0.00	107
	10	190	22.58	168	8.39	183	18.06	190	22.58	190	2.26	168	8.39	155	0.00	155	0.00	155
	20	394	11.30	389	9.89	362	2.21	509	43.79	383	8.19	362	2.26	363	2.54	354	0.00	354
	30	544	11.93	583	19.96	490	0.82	694	42.80	514	5.76	490	0.82	507	4.32	486	0.00	486
3	5	52	0.00	52	0.00	52	0.00	63	21.15	72	38.46	52	0.00	52	0.00	52	0.00	52
	10	131	16.96	124	10.71	112	0.00	130	16.07	124	10.71	112	0.00	123	9.82	112	0.00	112
	20	248	11.71	263	18.47	258	16.22	354	59.46	271	22.07	248	11.71	223	0.45	225	1.35	222
	30	346	20.56	317	10.45	314	9.41	481	67.60	307	6.97	307	6.97	307	6.97	297	3.48	287
4	5	61	0.00	63	3.28	63	3.28	61	0.00	61	0.00	61	0.00	61	0.00	63	3.28	61
	10	104	15.56	92	2.22	90	0.00	137	52.22	111	23.33	90	0.00	91	1.11	90	0.00	90
	20	145	11.54	185	42.31	151	16.15	218	67.69	189	45.38	145	11.54	149	14.62	137	5.38	130
	30	173	20.14	187	29.86	164	13.89	255	77.08	156	8.33	156	8.33	146	1.39	144	0.00	144
8	5	24	0.00	25	4.17	25	4.17	24	0.00	25	4.17	24	0.00	25	4.17	24	0.00	24
	10	29	0.00	48	65.52	49	68.97	29	0.00	29	0.00	29	0.00	29	0.00	29	0.00	29
	20	53	20.45	49	11.36	53	20.45	68	54.55	52	18.18	49	11.36	50	13.64	46	4.55	44
	30	60	22.45	57	16.33	58	18.37	88	79.59	59	20.41	57	16.33	57	16.33	54	10.20	49
평 균			11.75		15.98		12.00		37.96		13.56		4.86		4.89		1.77	

(참고) a : 각 휴리스틱해의 값

b : 최적해와의 比率隔差(%) b=100 × (휴리스틱해 − 최적해)/최적해

c : 휴리스틱 G는 휴리스틱 B − F중 최상의 결과를 선택한 것임

3.2 휴리스틱 I의 成果測定 및 기존해법과의 比較

휴리스틱해의 최적해와의 비율격차를 기준으로 하는 성능평가를 위해서 다음과 같은 成果尺度를 설정한다.

$$Z = \frac{Z_1 - Z_2}{Z_2} \times 100$$

Z_1:휴리스틱해의 값

Z_2:최적해의 값

여기서 Z는 특정 휴리스틱해의 최적해로부터 比率隔差(percentage deviation)를 나타낸다. 그리고 최적해는 제5장에서 설명하는 分段探索法(branch and bound)에 의해서 구한다.

<표 3-2>에서는 이와같은 成果尺度에 따라 이산적일양분포 U(0, 100)에 의거하여 산출한 16개의 실험문제에 대해서 최적해를 구하고 휴리스틱의 최적해와의 比率隔差를 구하여, 휴리스틱해법들간의 성과를 비교해 보았다.

이 실험결과에 의하면 Ibarra와 Kim의 휴리스틱해법 중에서 單一解法으로서는 휴리스틱 B, D 및 F가 비교적 양호한 결과를 산출하고 있으며 그 중에서도 특히 휴리스틱 B와 D가 최적해로부터 평균 12%의 격차를 보이고 있어 다른 해법에 비해서 우수하다는 것을 알 수 있다. 휴리스틱 G의 경우 최적해로부터 평균 4.86%의 격차를 보이고 있어 단일해법만 사용하는 경우보다 계산소요시간은 길어지지만 계산해는 상당히 개선시킬 수 있음을 나타내고 있다. 한편 De와 Morton의 휴리스틱 H는 최적해와의 격차가 4.89%로 Ibarra와 Kim의 單一解法에 비해서는 상당한 진전을 이룩하였으나 집단휴리스틱 G에 비해서는 거의 차이가 없음을 알 수 있다. 한편 機會費用의 개념에 입각한 휴리스틱 I는 최적해로부터의 평균격차가 불과 1.77%로 기존의 휴리스틱해와는 비교할 수 없을 정도로 최적해에 근접한 우수한 준최적해를 도출할 수 있음을 보여 준다. 그러나 여기서 실험문제의 수가 제한되어 있기 때문에 최적해로부터 比率隔差가 평균 1.8% 수준이라고 일반화하기는 힘든다. 比率隔差의 평균수준을 알기 위해서는 보다 다양한 규모의 문제에 대해서 다수의 실험을 해야 하므로 이 과제는 제5장으로 미루도록 한다.

4. 라그랑지안 下限價의 導出

1970년대에 들어와서 이룩한 수학적 프로그래밍 분야의 중요 업적중의 하나는 라그랑지안弛緩技法의 발견이다. 라그랑지안弛緩技法은 어떤 整數計劃問題가 일부 난해한 제약조건식 때문에 풀이가 어려운 경우에 일부 제약조건식을 弛緩시켜 목적함수에 포함시킴으로써 비교적 풀이가 용이한 문제로 재구성하는 기법이다. 라그랑지안 문제는 풀이가 용이할 뿐만 아니라 선형이완문제를 풀어서 구한 하한가보다 원문제의 최적해값에 근접한 엄밀한 하한가를 제공한다. 그러므로 이 기법은 分段探索法에서 일반적으로 下限價를 구하는 데 사용되는 線型計劃으로의 弛緩 대신에 보다 유용하게 이용될 수 있다.

原問題(P)의 라그랑지안弛緩式은 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(P)의 제약식($\sum_{i=1}^{n} t_{ij}x_{ij} \leq M$ for all j) 을 背囊制約式(knapsack constraints)이라 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.

$I=\{1, 2, \cdots, n\}$가 작업집합, $J=\{1, 2, \cdots, m\}$가 기계집합, 그리고 t_{ij}가 작업시간 행렬이라면, 두 이완문제는 다음과 같다.

(LR Ⅰ)

$$Z_{LRI} = \text{Min}[M + \sum_{j=1}^{m} u_j(\sum_{i=1}^{n} t_{ij}x_{ij} - M)]$$
$$= \text{Min}[\sum_{i=1}^{n}\sum_{j=1}^{m} u_j t_{ij} x_{ij} + M - \sum_{j=1}^{m} M \cdot u_j]$$
$$= \text{Min}[\sum_{i=1}^{n}\sum_{j=1}^{m} u_j t_{ij} x_{ij} + M(1 - \sum_{j=1}^{m} u_j)]$$

s.t. $\sum_{j=1}^{m} x_{ij} = 1 \quad \forall i \in I$

$x_{ij} \in \{0, 1\}$

M은 自由變數(free variable)

(LR Ⅱ)

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

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

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

v_i, M 자유변수

(LR Ⅰ)은 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 이 기법을 이용해서 구한 하한가는 Z_{LP} 이상으로 原問題의 최적해 Z_{IP}와의 쌍대격차를 크게 개선할 수 없기 때문에 分段探索法에 도입하기는 힘들 것으로 예상된다. 그러므로 보다 최적해에 근접한 하한가를 도출하기 위해서는(LR Ⅱ)의 풀이를 시도해 보는 수밖에 없다.

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−complete문제로서 나타났다. 최근 다수제약식 0−1 배낭문제의 最適解를 구하기 위한 해법의 연구가 Shih(1979), Gavish 와 Pirkul(1985)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 위해 NP−complete인 原問題를 또 다른 NP−complete인 弛緩問題로 전환시키는데 불과하므로 적정계산시간이내에 적절한 下限價를 구하기가 사실상 힘들어진다. 뿐만 아니라 이 문제의 제약식 우변의 값 M도 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(1976)에 의해 변수개수 약 10,000개까지 풀어 낼 수 있는 類似多項時間解法이 개발되어 있다. 그러므로 우변의 변수 M을 하나의 整數값에 고정시키고, 乘數를 서브그래디언트의 일정단계에서 고정시키면, Nauss의 해법을 이용하여 분해된 단일제약식 0−1배낭문제들을 풀고, 이를 결합하면 주어진 M과 승수값에서의 이완문제의 최적해 값을 구할 수 있다. 다음 과제는 M의 값을 변화시켜 이완문제의 최적해값의 증감을 계산하여 이 중에서 이완문제를 최소로 하는 M값을 결정하는 것이다. 이때 M의 범위가 크면 M의 정수값 하나하나에 대해서 상기과정을 반복해야 하므로 막대한 계산시간이 소모된다. M값의 가능한 범위를 최소한으로 축소시키기 위하여, M에 대한 上限價와 下限價를 구하여 이완문제의 제약조건식에 추가시켰다. M에 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 M의 1차 下限價로 보아 $\underline{M}$라고 하고 휴리스틱으로 구한 上限價를 $\overline{M}$라 하여 M의 범위를 $\underline{M} \le M \le \overline{M}$로 한정하여 (LR Ⅱ)문제에 추가시키기로 한다.

새로운 (LR Ⅱ)

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

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

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

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

$$\underline{M} \le M \le \overline{M},\ M\text{은 정수}$$

$$v_i \quad \text{자유변수}$$

앞에서 설명한 바와 같이, 분해된 각 背囊問題의 최적해를 구하여 이를 總合(aggregate)한 결과가 주어진 v_i와 M에서의 최적해가 된다. 다음 M값을 下限價에서 上限價까지 整數單位로 증가시켜가며 각 M값에서의 최적해를 구하고, 이 중에서 최소값을 내는 해를 주어진 v_i에서의 최적해 $Z_{LR\,II}$로 삼는다. 그리고 서브그래디언트 절차에 따라서 v_i값을 변화시켜 가며 $Z_{D(LR\,II)} = \mathrm{Max}_v \{Z_{LR\,II}\}$를 구한다.

이 과정에 대한 단계별 알고리즘은 다음과 같다.

(단계 1) 문제구성

原問題의 割當制約式을 雙對變數 v_i로 弛緩시켜 LR Ⅱ 식을 구성한다.

(단계 2) 초기화

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

$\overline{M} = Z_H$

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

$\lambda_1 = 2$

$M^1{}_1 = 1 + \mathrm{INT}[\underline{M}]$ if $\underline{M}$ is not integer, o.w $M^1{}_1 = \underline{M}$

여기서 INT는 整數化 函數임

(단계 3) K번째 반복계산시의 LR Ⅱ 풀이

a) (LR Ⅱ)를 제약조건식의 개수 만큼 분해하여 m개의 單一제약식

배낭문제로 만든다.

b) 각각의 單一背囊問題를 풀어 각 제약식의 최적해를 구한다.

(Nauss의 유사다항시간해법 이용)

c) 이 결과를 총합하여 $Z_{LR\,II}$를 구한다.

d) M을 M^k_1에서 1씩 증가시켜 가면서 $\bar{M}$를 초과하기 전까지, 각 M에 대한 $Z_{LR\,II}$를 구하여 비교하고 이중 M^k_R에서 $Z_{LR\,II}$가 최소가 된다면 이 값은 주어진 v^k_i에서의 최적 목적함수값 $Z_{LR\,II}(v^k)$라 하고, 이때의 $x_{ij}=x_{ij}^k$, $M=M^k_R$이 최적해가 된다.

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

(단계 4) K번째 반복계산시의 方向벡터(Ax^k-b)와 前進距離 t_k및 새로운 雙對變數 v_i^{k+1}을 구함

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

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

따라서 $v_i^{k+1}=v_i^k+t_k(A^k-b)$

참고:Z^*는 $X_{LR\,II}$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.

a) $Z_H=Z_{LR\,II}(v^k)$ → 최적해 발견

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

c) 반복계산 횟수가 일정한계를 초과할 때

(예 : k>150)

d) $Z_{LR\,II}$가 어떤 수에 수렴할 때

e) a)-d)의 어느조건도 충족시키지 못하면 k=k+1이 되고 단계3으로 되돌아간다.

(LR I), (LR II)로 구한 각 하한가의 최적해와의 비율격차가 <표 4-1>에 제시되어 있다.

이 표에서 (LR II)에 의한 하한가는 최적해에 거의 수렴하는 강력한 한계치(bound)임을 알 수 있다.

5. 竝列處理機械問題의 分段探索 프로그램 構成

5.1 프로그램의 構成

竝列處理機械問題의 最適解를 구하기 위한 분단탐색 프로그램은 <그림 5-1>과 같이 구성된다.

표 4-1 下限價들의 最適解와의 比率隔差 比較

문제의 규모		실험	최적해	* 최적해와의 격차(%) *					
기계	작업	문제의	발견	背囊制約式弛緩라그랑지안下限價(Z_{LRI})			割當制約式弛緩라그랑지안下限價($Z_{LR\,II}$)		
대수	개수	개수	개수	최소	평균	최대	최소	평균	최대
2	5	5	5	1.97	10.80	16.22	0.00	1.91	8.04
2	10	5	5	3.17	6.14	10.07	0.00	0.00	0.01
2	15	5	5	1.27	3.08	4.56	0.00	0.42	0.81
2	20	5	5	1.33	2.01	3.62	0.00	0.35	0.86
3	10	5	5	4.30	12.73	16.40	0.00	1.40	7.29
3	15	5	5	3.13	5.41	9.42	0.00	0.45	1.87
3	20	5	5	3.59	4.96	6.39	0.00	0.87	1.40
3	25	5	5	2.10	3.34	4.59	0.00	0.42	1.06
4	15	5	5	6.21	9.91	13.17	0.00	1.03	2.81
4	20	5	5	4.72	5.97	6.64	1.15	1.61	1.84
4	25	5	4	2.03	5.57	7.55	0.00	1.18	2.51
4	30	5	5	2.71	3.87	6.06	0.02	0.99	2.06
5	20	5	5	9.11	11.92	13.59	0.00	2.09	3.86
5	25	5	5	5.23	8.12	10.48	0.00	0.91	1.24
5	30	5	3	4.69	5.65	6.90	0.01	0.76	2.28
5	35	5	4	5.27	6.36	7.26	0.01	0.62	2.08
평 균				4.05	6.62	8.73	0.07	0.94	2.50

(참 고) 최적해가 발견된 문제에 한해서 계산하였음.

比率隔差(percent gap) = $100(Z^* - \text{각 下限價})/Z^*$

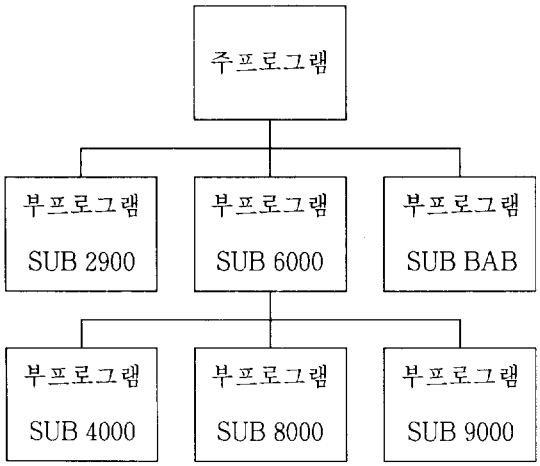

그림 5-1 主프로그램과 副프로그램의 관계

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 : (LRⅡ)에 기초한 라그랑지안 이완 및 서브그래디언트 기법(下限價 도출)
4) 부프로그램 SUB 4000 : (LRI)에 기초한 라그랑지안 이완 및 서브그래디언트 기법(下限價의 초기화)
5) 부프로그램 SUB 8000-9000 : 단일제약식 배낭문제의 풀이
6) 부프로그램 SUB BAB : 최초마디(root node)에서의 라그랑지안해를 중심으로 한 민감도분석(라그랑지안휴리스틱 분단탐색절차)

5.2 프로그램의 段階別 細部節次

(1)최초마디에서 上限價는 휴리스틱Ⅰ(서브루틴 SUB 2900)로 계산하고, 下限價는 라그랑지안 이완문제(LRⅡ)(서브루틴 SUB 6000)로 계산하되 (LRI)은 (LRⅡ)를 위한 총작업완료시간변수 M의 초기하한치(M)를 계산하는 데 이용한다(이 과정이 서브루틴 SUB 4000 임). 서브루틴 SUB 6000 속에서는 하위문제로 풀어야 하는 단일제약식배낭문제(single knapsack problem)에 대한 프로그램이 서브루틴 SUB 8000과 SUB 9000으로 구성되어 포함되어 있다. 이 서브루틴은 Nauss에 의해서 개발된 프로그램을 채택하여 삽입한다. 계산결과

1) 최초마디에서 下限價와 上限價의 격차가 1 이하이면 휴리스틱해의 값이 곧 최적해가 된다.
2) 上限價와 下限價의 간격이 1이상 5이하이고, 이 범위내에서 下限價가 어떤 특정치에 수렴하면 下限價의 바로 위 정수값을 잠정적인 최적해로 간주하고 上限價=下限價가 이루어질 때까지 마디 탐색.
3) 위의 간격이 5이상이면 人爲上限價를 下限價와 上限價사이의 적당한 중간치로 선정하여 下限價가 어떤 특정치에 수렴할 때까지 계속 개선시켜 나감.

(2) 최초 마디에서의 마지막 라그랑지안 해 $x_{ij}(v^*)$를 토대로, 이 해 자체가 실행가능한지, 아니면 이 해를 攪亂分析(perturbation analysis)했을 때 가까운 위치에 개선된 實行可能解가 존재

하는지를 분석하여 上限價의 개선을 시도해 본다. 이 과정은 별도의 縱優先 分段探索法인 서브루틴 BAB에서 수행되며, 이 프로그램은 다음과 같이 진행한다. 이 과정은 라그랑지안해를 기초로 하여 보다 나은 實行可能解를 찾는 절차로서 일종의 라그랑지안 휴리스틱 절차로 볼 수 있다.

1) 分離優先順位의 決定(separation order)

① 마지막 라그랑지안해에서 $x_{ij}=1$로 결정된 변수를 $x_{ij}=0$로 바뀌었을 때 벌과치(penalty) $= Z_{LR\,II}' - Z^* > 0$인 변수를 우선적으로 고정시킴 → 문제축소과정

[$Z_{LR\,II}'$: 下限價
  Z^* : 上限價]

② 한 대의 기계에만 배정된 작업에 해당하는 변수

③ 두 대 이상의 기계에 배정된 작업에 해당하는 변수 x_{ij}중에서 $x_{ij}=0$으로 고정된 변수

④ ③의 변수 중 $x_{ij}=1$로 고정된 변수

⑤ 아무 기계에도 배정되어 있지 않은 작업에 해당하는 변수의 순으로 분리됨

2) 分枝優先順位의 결정(branching order)

下位段階(level)에 해당하는 마디부터 分枝시키되 동일단계에 두 개 이상의 마디가 존재하면 下限價가 작은 마디부터 선택

3) 逆追跡(backtracking)節次

從優先探索(depth first)의 원칙에 의하여 한 가지의 탐색이 완전히 끝난 뒤 다음 가지로 넘어감.

4) 이와 같은 탐색과정에서 모든 작업이 하나의 기계에 배정되는 實行可能解로서 初期上限價보다 개선된 해가 나오면 이를 새로운 上限價로 설정한다.

(3) 이와 같은 과정을 거친 후 上限價와 下限價의 격차가 1이하이면 해당 上限價를 도출하는 實行可能解가 最適解이며, 그렇지 않으면 변수 x_{ij}를 하나씩 순서대로 분리변수로 선택하여(i=1, 2, …, n순으로, j=1, 2, …, m 순으로) 0과 1로 고정시켜보고, 下限價가 낮은 마디부터 우선적으로 분기시키는 主프로그램(main program)에서의 분단탐색을 수행한다. 여기서도 가지삭제(fathoming)는 縱優先探索(depth-first)방식을 따르도록 한다.

(4) 최대탐색 마디수를 3,000개로 제한하여 이 범위내에서 최적해 탐색이 이루어지지 않으면 최적해탐색이 불가능한 문제로 분류된다.

5.3 실험문제의 구성 및 분석

앞에서 제시한 最適解法으로 풀이할 수 있는 문제의 크기와 계산소요시간, 탐색마디의 수 등을 파악하기 위하여 우선 1차적으로 24개의 豫備實驗問題를 만들었다. 각 實驗問題의 작업시간행렬 t_{ij}는 $1 \le t_{ij} \le 100$의 整數를 離散的 一樣分布(discrete uniform distribution)에 의거하여 無作爲 抽出하였다.

문제의 규모는 기계대수를 2대에서 10대까지, 작업개수는 5개에서 50개까지 차례로 증가시키면서 양자를 적당히 조합하여 구성하였다. 각 문제에 대해서 휴리스틱I에 의한 上限價와 LR

표 5-1 離散的一樣分布 U(0,100) 豫備實驗問題에 대한 最適解 探索結果

문제의 규모		휴리스틱I(上限價)		최종 LR ‖(下限價)		최적해	分段探索法에서의 마디수	CPU시간[b] (단위:초)	최적해탐색 불능 이유
기계대수	작업개수	Z_H	比率隔差(%)	Z_{LR} ‖	比率隔差(%)				
2	5	112	0.00	111.9999	0.00	112	1	29.674	
2	10	158	0.00	157.9999	0.00	158	1	19.119	
2	15	292	0.34	290.9999	0.00	291	2	67.878	
2	20	367	0.00	366.9999	0.00	367	1	23.412	
3	10	101	0.00	100.9999	0.00	101	1	35.578	
3	15	181	0.56	179.9999	0.00	180	2	31.249	
3	20	161	7.33	149.9999	0.00	150	2	113.569	
3	25	276	3.37	266.9999	0.00	267	2	140.545	
4	15	90	1.12	88.9873	−0.01	89	2	55.323	
4	20	128	4.92	121.9208	−0.06	122	3	170.271	
4	25	157	5.37	148.9927	−0.00	149	17	277.013	
4	30	125	1.63	122.8748	−0.10	123	21	164.836	
5	20	115	7.48	106.8705	−0.12	107	31	449.693	
5	25	85	3.66	81.9664	−0.04	82	34	106.876	
5	30	96	1.05	94.8124	−0.20	95	3	321.762	
5	35	113	8.65	103.3476	−0.63	104	41	323.151	
8	25	48	11.63	42.9899	−0.02	43	11	233.233	
8	30	54	10.20	48.9709	−0.06	49	19	603.759	
8	35	68	13.33	58.9937	−1.68	60	67	567.165	
8	40	69	16.95	58.5144	−0.82	59	2	512.367	
10	30	48	20.00	39.5471	−1.13	40	1,673	716.838	
10	35	45	7.14	41.8725	−0.30	42	757	572.322	
10	40	49	11.36	43.8183	−0.41	44	61	517.636	
10	50	59	—	48.0115	—	—	—	—	a
평 균			5.92		−0.24				

(참 고) a : 1차 分段探索에서 탐색마디의 한계 3000개를 초과하고, 2차 分段探索에서 CPU 시간의 한계 2000 CPU second를 초과한 경우

b : CPU second는 CYBER-835의 계산시간임

표 5-2 離散的一樣分布 U(0, 100) 實驗問題에 最適解 探索結果

문제의규모			휴리스틱上限價의 최적해와의 격차(%)			최종라그랑지안下限價의 최적해와의 격차 (%)			탐색마디수			CPU 계 산 시 간 (단위:초)		
기계 대수	작업 개수	a b c d e f	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	1673	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	—	—	—	—	—	—	—	—	—	—	—	—
평 균				5.83			0.41							

(참 고) a : 實驗問題의 수
b : 최적해가 발견된 문제의 수
c : 分段探索法에 들어가기 전 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 휴리스틱I의 해와 다른 경우

Ⅱ에 의한 下限價및 최적해와 탐색마디수, CPU 시간등을 계산하여 <표 5-1>과 같이 정리하였다.

1차 실험결과 適正探索마디수(3,000마디)와 適正計算時間(2,000 CPU second)이내에 최적해가 발견되는 문제는 대개 變數의 개수(기계대수×작업개수)가 400개(기계10대×작업40개)이내인 경우이며 그 이상은 적정시간이내에 탐색이 힘든다는 결론에 도달하였다. 이 수준은 De 와 Morton이 單純列擧에 의한 分段探索法으로 變數 40개(기계 4대×작업10개)까지 최적해를 발견할 수 있었던 것과 비교하면 상당한 개선을 이룩하였다고 볼 수 있다. 1차실험에서 휴리스틱I로 구한 上限價의 최적해와의 比率隔差(percentage gap;100(上限價-최적해)/최적해)는 기계대수가 증가함에 따라 커지는 경향이 있으며 평균 5.92% 수준이지만 라그랑지안 下限價의 比率隔差는 문제의 크기에 별로 영향을 받지않고 평균 0.24% 수준으로서, 이것은 下限價가 거의 최적해에 수렴함을 보여주고 있다. 이 수치가 <표 4-1>에서 나타난 라그랑지안 下限價의 平均比率隔差 0.94% 보다 더욱 개선된 것은 下限價가 수렴될 때까지 M(총작업완료시간의 하한가)을 계속 상승시키면서 서브그래디언트를 반복하여 최종적으로 얻어진 수치이기 때문이다. 이와같이 최종 라그랑지안 下限價가 최적해값과 거의 일치하고, 최종 라그랑지안해가 최적해와 대단히 가까운 속성이 있기 때문에 라그랑지안 휴리스틱이라는 약간의 變數조정으로 최적해에 도달할 수 있게 된다. 그리고 라그랑지안 휴리스틱의 절차를 통해서도 최적해를 발견하기 힘든 문제들에 대해서는 최적해값이 최종라그랑지안 下限價에서 크게 벗어나지 않을 것이라는 확신을 가질 수 있게 해준다.

1차 실험결과를 토대로 최적해발견비율이 비교적 높은 기계대수 10대 작업개수 50개 미만인 實驗問題를 각 경우별로 5개씩 만들어 2차실험을 하여 <표 5-2>와 같이 정리하였다.

2차 실험에서는 總實驗問題중 최적해가 발견된 문제의 비율, 分段探索法이용전에 최적해가 발견된 문제의 비율, 分段探索에서 최적해가 발견된 문제의 수, 탐색마디의 수, CPU계산시간 등을 조사하였다. 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(라그랑지안 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.

6. 結 論

본 연구에서는 分割이 不可能한(non-preemptive) 作業群을 異速竝列處理機械(unequal parallel processor)상에서 적절히 할당하여 總作業完了時間(makespan)을 最小化하기 위한 最適解法을 구하고자 하였다. 作業分割이 可能한 경우의 竝列處理機械問題는 기본적으로 決定變數의 整數制限이 없는 線型計劃問題(linear programming problem)의 範疇에 속하므로 심플렉스등의 해법을 동원하면 最適解를 실행가능시간 내에 구할 수 있다. 그러나 현실상황에서 보다 一般性을 가지는 작업분할이 불가능한 문제에 대해서는 等速機械, 比例速機械 및 異速機械의 모든 유형에서 아직 實用的인 最適解法이 개발되어 있지 않다. 일반적으로 작업분할이 불가능한 경우 병렬처리기계상에서 총작업완료시간을 최소화하는 문제는 強 NP--complete 整數計劃問題로 분류되며, 따라서 多項時間이내에 最適解를 구하기는 사실상 불가능하다고 보고 있다.

최근에 難解한 제약조건식을 이완시켜 原問題의 最適解에 접근해 가는 여러가지 弛緩技法과 探索技法에 관한 수학적 이론이 발달함으로써, 이를 이용하여 일부 NP-complete문제들에 대해서 비록 多項時間 내에 수렴하는 最適解法은 아니라고 하더라도 類似多項時間解法(pseudo-polynomial time algorithm)을 구성하는 데는 성공한 바 있다.

본 연구도 이러한 硏究趨勢를 반영하여 強 NP-complete 屬性을 가지는 作業分割不能 異速竝列處理機械上의 總作業完了時間 最小化問題에 대해서 기존의 휴리스틱 연구영역을 벗어나 最適解法의 개발을 시도하였다. 이러한 시도는 라그랑지안이완, 서브그래디언트기법 및 분단탐색등의 限界値理論(bounding theory)에 기초를 두고 있는 바, 이 시도를 통해서 다음과 같은 연구성과를 거두었다.

첫째, 後悔(機會費用)의 개념을 이용하여 기존 휴리스틱해법보다 平均的인 觀點에서나 最惡限界의 觀點에서 最適解와의 比率隔差가 크게 줄어든 準最適解(實行可能解)를 산출하는 새로운 휴리스틱해법을 개발하였다.

둘째, 原問題의 일부 制約條件式을 目的函數로 이완시켜 라그랑지안 이완문제를 구성하고 서브그래디언트기법을 적용하여 이완문제의 최적해를 구해본 결과 이 값이 원문제의 최적해값에 수렴되는 성질을 지닌 엄밀한 下限價로서 나타남을 파악하였다.

셋째, 위에서 구한 엄밀한 上限價(準最適解)와 下限價를 토대로 한계치의 폭을 좁혀가면서 最適解를 인식하는 효율적인 分段探索의 方法을 개발하였다.

넷째, 이러한 해법들의 개발로 인하여, 最適解를 구할 수 있는 문제의 規模와 比率이 크게 향상되었다.

다섯째, 본 연구에서 개발한 새로운 휴리스틱해법이나 최적해법은 병렬처리형태의 시스템환경에 응용되어 시스템의 성과를 제고시키는 데

기여할 수 있다.

최근 竝列機械에서의 日程計劃에 대한 중요성이 크게 부각되고 있는데 그것은 情報通信技術의 발달로 인해 개별적으로는 한정된 능력을 지닌 處理機(processor)들을 네트워크로 연결하여 이들 處理單位들 간에 신속하고 긴밀한 정보교환이 이루어지도록 할 수 있고 따라서 多量多數의 작업군이 시스템에 부과되었을 때 이를 적절히 分散處理하여 전체작업의 處理規模를 확장하고 處理効率을 높이는 것이 가능해졌기 때문이다. 특히 오늘날 활발하게 추진되고 있는 병렬컴퓨터의 하드웨어적 개발과 더불어, 병렬구조하에서 작업의 효율적인 배분을 도모하기 위한 소프트웨어적인 연구 역시 필연적으로 요구되고 있는 상황이다.

作業分割不能 異速竝列處理機械의 定義에서 설정한 諸 假定들은 이러한 시스템들의 環境과 상당히 유사하기 때문에, 본 연구에서 개발한 해법들은 다양한 시스템환경에 응용되어 시스템의 효율을 증진시킬 수 있을 것으로 기대된다.

참 고 문 헌

[1] Baker, K., *Introduction to Sequencing and Scheduling*, New York, John Wiley and Sons, 1974, pp.114-115.

[2] 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.

[3] De, P. and T.E. Morton, "Scheduling to Minimize Makespan on Unequal Parallel Processors," *J. ACM*, 26, 1979.

[4] Garey, M. and David S. Johnson, *Computers and Intractability*, W.H.Freeman and Company, San Francisco, 1979, pp.96-99.

[5] Gavish, B. and Hasan Pirkul, "Efficient Algorithms for Solving Multiconstraint Zero-one Knapsack Problems to Optimality", *Mathematical Programming* 31, 1985.

[6] Geoffrion, A., "Lagrangean Relaxation and its Uses in Integer Programming, "*Math programming Study*, Vol. 2,1974, pp.82-114.

[7] Gonzales, T. and S.Sahni, "Preemptive Scheduling of Uniform Processor," *J. of ACM*, Vol.25, No.1, Jan., 1978, pp.92-101.

[8] Graham, R., "Bounds on Multiprocessing Timing Anomalies," *SIAM J. Applied Math*, *17*, 1969, pp. 416-417.

[9] Held, M, and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees:Part Ⅱ",

Math. Programming 6, 1971, pp.62－88.

[10] Held, M. and R.M. Karp, "The Traveling Salesman Problem and Minimum Spanning Trees, "*Operations Res.*, Vol. 18,1970, pp. 1138－1162.

[11] 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.

[12] Hochbaum, D. and David B. Shmoys, "Using Dual Approximation Algrorithms for Scheduling Problems:Theoretical and Practical Results", *J. of ACM*, Vol.34, No.1, Jan. 1987, pp.144－162.

[13] Ibarra, O. and C.E.Kim, "Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors," *J. of ACM*, V.24, 1979, pp.280－289.

[14] 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.

[15] McNaughton, R., "Scheduling with Deadlines and Loss Functions," *Management Science*, Vol.6, No.1, 1959

[16] Nauss, R., "An Efficient Algorithm for the 0－1 Knapsack Problem," *Management Science*, Vol.23, No.1, September, 1976.

[17] Poljak, B., "A General Method of Solving Extremum Problems," *Soviet Math.* 8,1967, pp. 593－597.

[18] Poljak, B., "Minimization of Unsmooth Functionals," *U.S.S.R. Computational Math, and Math. Phys.* 9, 1969, pp.14－29.

[19] Sahni, S., "Algorithms for Scheduling Independent Tasks.", *J. of ACM*, Vol. 23, No.1, Jan. 1976. pp. 116－127.

[20] Shih, W., "A Branch and Bound Method for the Multiconstraint Zero－one Knapsack Problem,"*Journal of the Operational Research Society*, Vol.30, No.4, 1979.