韓國經營科學會誌
第23卷 第2號
1998年 6月

U라인 라인밸런싱을 위한 분지한계법*

김여근** · 김재윤** · 김동묵*** · 송원섭**

A Branch-and-Bound Algorithm for U-line Line Balancing

Yeo Keun Kim** · Jae Yun Kim** · Dong Mook Kim*** · Won Seop Song**

Abstract

Assembly U-lines are increasingly accepted in industry, especially just-in-time production systems, for the efficient utilization of workforce. In this paper, we present an integer programming formulation and a branch-and-bound method for balancing the U-line with the objective of minimizing the number of workstations with a fixed cycle time. In the mathematical model, we provide the method that can reduce the number of variables and constraints. The proposed branch-and-bound method searches the optimal solution based on a depth-first-search. To efficiently search for the optimal solutions to the problems, an assignment rule is used in the method. Bounding strategies and dominance rules are also utilized. Some problems require a large amount of computation time to find the optimal solutions. For this reason, some heuristic fathoming rules are also proposed.

Extensive experiments with test-bed problems in the literature are carried out to show the performance of the proposed method. The computational results show that our method is promising in solution quality.

1. 서 론

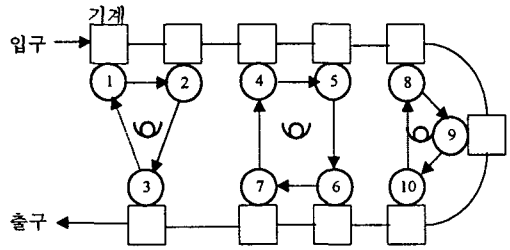
조립라인밸런싱(assembly line balancing)문제는 조립라인에 부과된 여러 제약들을 어기지 않고, 하나 또는 그 이상의 목적들이 최적이 되도록 라인 내의 작업장에 작업들을 할당하는 문제이다. 전통적인 조립라인밸런싱문제는 작업장이

* 이 연구는 1997년도 전남대학교 학술연구비 지원에 의해 연구되었음
** 전남대학교 산업공학과
*** 동신대학교 산업공학과

직선형태로 배치된 생산라인을 고려하며, 밸런싱은 선행제약과 사이클타임제약을 만족하도록 작업장에 작업들을 할당함으로써 결정된다.

JIT(Just-In-Time) 생산시스템에서는 수요변동에 유연하게 대응할 수 있는 U자형 조립라인(이하 U라인이라 함)을 흔히 사용하고 있다. U라인은 [그림 1]과 같이 라인의 출구와 입구가 같은 위치에 있는 라인을 말한다[13]. 본 연구에서는 U라인의 조립라인 밸런싱문제를 다룬다.

U라인은 입·출구 작업이 동일한 작업자에 의해 이루어지므로 제품 한 단위가 출구에서 떠나면 한 단위의 재료가 공정의 입구에 투입되어 이른바 '당기기 방식에 의한 적시생산(just-in-time pulling system)'이 가능하다. 이와 같이 출구와 입구의 작업이 동일한 작업자에 의해 이루어지므로 라인내의 재공품 수량은 항상 일정하게 유지된다. 또한 각 기계마다 재공품을 표준 보유량만 유지하게 함으로써 작업자간의 작업 불균형이 눈에 드러나게 되어 공정의 개선이 촉구될 수 있다[13].



[그림 1] U자형 조립라인의 예

작업할당의 관점에서 U라인과 직선라인(straight line)을 비교하여 보자. 직선라인의 작업할당에서는 모든 선행작업들이 할당된 작업집합에서 할당할 작업을 선택한다. 반면에 U라인에서는 모든 선행작업들이 할당된 작업집합과 모든 후행작업들이 할당된 작업집합의 합집합이 할당가능한 작업집합이 된다. 따라서 가능해 집합이 커진다. 그러므로 U라인은 직선라인에 비해 작업할당에 대한 유연성이 높아 좋은 작업할당이 가능하며, 요구되는 작업장수 역시 직선라인의 작업장수보다 크지 않다[12].

본 연구에서는 U라인밸런싱(U-Line Balancing: ULB)문제에서 사이클타임이 주어진 작업장수 최소화문제의 최적해를 구할 수 있는 분지한계법을 제안한다. 지금까지의 라인밸런싱문제는 대부분 직선라인에 관하여 연구되었으며, U라인에 관해서는 많은 연구가 이루어지지 않았다. ULB문제에 관한 연구는 Miltenburg와 Wijngaard [12]에 의해 처음 시작되었다. 그들은 U라인에서 사이클타임이 주어진 작업장수 최소화 문제의 최적해를 구하기 위한 동적계획 모형과 발견적 기법을 연구하였다. 동적계획 모형은 문제의 크기가 커지면 많은 계산시간이 소요되어 주어진 시간내에 최적해를 찾지 못한다는 단점을 갖는다. 그들은 발견적 기법으로 직선라인밸런싱(Straight Line Balancing: SLB)문제의 최대순위위치가중치(Maximum Ranked Positional Weight)규칙을 U라인의 특성에 맞게 수정한 할당규칙을 제안하였다. Hwang *et al.*[7]은 U라인에서 작업시간과 작업자의 작업장내 작업간 이동시간을 동시에 고려한 작업장수 최소화 문제를 발견적 기법을 이용하여 해결하였다. 또한 SLB문제에 대한 분지한계법의 적용은 여러 연구[5,6,8,10]가 있으나, ULB에 분지한계법을 적용한 연구는 아직 이루어지지 않았다.

ULB를 위한 분지한계법에서 좋은 초기해를 구하기 위하여 할당규칙에 의해 작업을 작업장에 할당하는 가능해 열거방법을 제안하고, 적절한 분지전략과 한계전략, 그리고 분지완료규칙

들을 다룬다. 분지한계법을 이용하여 최적해를 구하는데 소요되는 계산시간은 상대적으로 적거나 또는 아주 많이 소요되는 경향이 있다[6]. 따라서 최적해를 구하는데 상대적으로 계산시간이 많이 소요되는 특성을 갖는 문제들을 해결하기 위한 발견적 분지완료규칙들을 제안한다. 또한 ULB문제에 맞는 0-1정수계획의 모형화를 통하여 변수와 제약식의 수를 감소시킬 수 있는 방법을 추가적으로 다룬다.

본 논문의 구성은 다음과 같다. 2절에서는 ULB문제에서 작업장수 최소화문제에 대한 0-1 정수계획모형을 제시한다. 3절은 U라인에서 주어진 목적에 대한 가능해 열거법을 설명하고, 4절은 분지한계법에 대하여 다룬다. 5절에서는 실험을 통하여 제안한 알고리듬의 성능을 분석하고, 6절은 결론으로 구성된다.

2. U라인밸런싱의 0-1정수계획 모형

2.1 기호정의

먼저 수리모형의 전개를 위하여 아래와 같이 기호를 정의한다.

I : 작업의 집합; $I=\{1,2,\ldots,i,\ldots,M\}$.

J : 작업장의 집합; $J=\{1,2,\ldots,j,\ldots,N\}$.

$P(i)$: 작업 i의 직선행작업 집합.

$P_a(i)$: 작업 i의 모든 선행작업 집합.

$S(i)$: 작업 i의 직후행작업 집합.

$S_a(i)$: 작업 i의 모든 후행작업 집합.

E_i : 작업 i의 할당가능한 가장 이른 작업장.

L_i : 작업 i의 할당가능한 가장 늦은 작업장.

$J(i)$: 작업 i의 할당가능 작업장 집합.

$I(j)$: 작업장 j에 할당될 수 있는 작업집합.

CT : 사이클타임.

t_i : 작업 i의 작업시간.

u : 아주 큰 양의 정수.

$[x]^+$: x보다 크거나 같은 최소의 정수.

E_i와 L_i는 $J(i)$와 $I(j)$를 구하기 위한 것이다. 보다 적은 $J(i)$와 $I(j)$를 구하면 수리모형에서 변수와 제약식수를 줄일 수 있으므로 E_i와 L_i의 결정은 중요하다. 모든 선행작업들이 할당되어 새로운 작업을 할당하는 것을 전방향 작업할당, 모든 후행작업들이 할당되어 새로운 작업을 할당하는 것을 후방향 작업할당이라 할 때, E_i^f와 L_i^f는 전방향 작업할당시 작업 i의 할당가능한 가장 이른 작업장과 가장 늦은 작업장, E_i^b와 L_i^b는 후방향 작업할당시 작업 i의 할당가능한 가장 이른 작업장과 가장 늦은 작업장으로 각각 나타낸다. SLB문제에서 Patterson과 Albracht [14]가 정의한, 작업 i가 할당될 수 있는 가장 이른 작업장과 가장 늦은 작업장은 전방향만을 고려했기 때문에 본 연구에서는 E_i^f와 L_i^f에 해당된다. 그러므로 E_i^f와 L_i^f는 각각 $[(t_i+\sum_{h\in P_a(i)} t_h)/CT]^+$과 $N+1-[(t_i+\sum_{h\in S_a(i)} t_h)/CT]^+$로 주어진다[14]. 만약 후방향부터 작

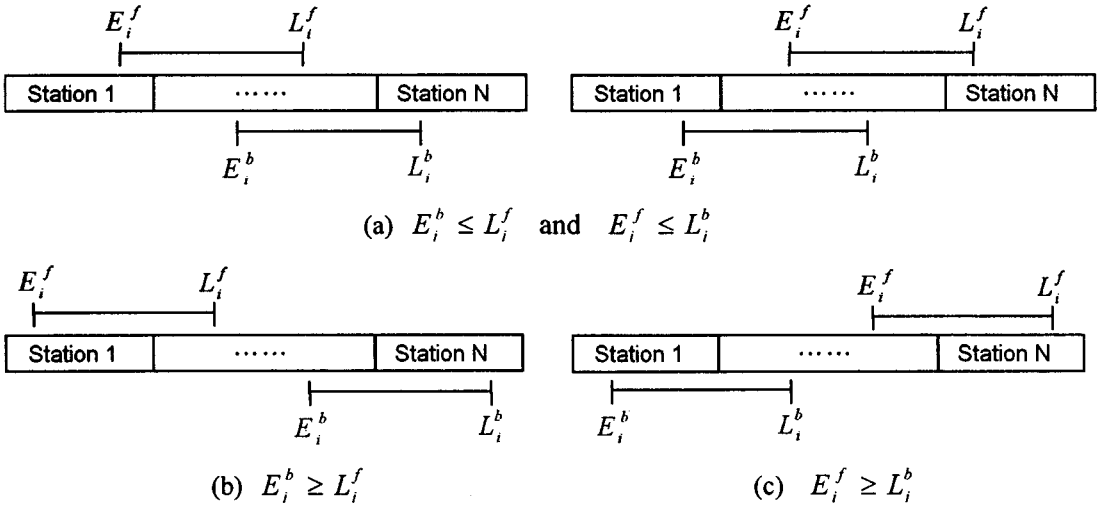
업할당을 한다면 E_i^b와 L_i^b는 $[(t_i+\sum_{h\in s_a(i)} t_h)/CT]^+$과 $N+1-[(t_i+\sum_{h\in p_a(i)} t_h)/CT]^+$로 각각 나타낼 수 있다. 이것들은 작업 i를 기준으로 후방향으로부터 시작하여 현재까지 할당된 작업을 작업 i의 후행작업으로 두고 E_i^f와 L_i^f를 간단히 변형한 것이다. 따라서 U라인에서는 E_i와 L_i는 아래와 같이 된다.

$$E_i = \min(E_i^f, E_i^b)$$

$$L_i = \max(L_i^f, L_i^b)$$

직선라인에서 작업 i의 할당가능 작업장 집합 $J(i)=\{j\in J \mid E_i\le j\le L_i\}$가 된다. 그러나 U라인에서는 전방향 작업할당과 후방향 작업할당에 대한 가장 이른 작업장과 가장 늦은 작업장을 구하면 [그림 2]와 같은 네가지 경우가 발생한다. (a)는 전방향의 할당가능한 작업장 범위와 후방향의 할당가능한 작업장 범위사이에 교집합이 존재하는 경우이며, (b)와 (c)는 그렇지 않은 경우이다. [그림 2]를 통하여 알 수 있듯이 $J(i)$는 전방향의 할당가능한 작업장 범위와 후방향의 할당가능한 작업장 범위사이에 교집합 존재여부에 따라 구분되므로 다음과 같이 구할 수 있다. 그리고 모든 i에 $J(i)$로부터 모든 j에 대한 $I(i)$를 쉽게 구할 수 있다.

$$J(i)=\begin{cases}\{j\in J \mid \min(E_i^f,E_i^b)\le j\le \max(L_i^f,L_i^b)\},\\ \qquad\{j\in J \mid E_i^f\le j\le L_i^f\}\cap\{j\in J \mid E_i^b\le j\le L_i^b\}\neq\phi\text{이면},\\ \{j\in J \mid E_i^f\le j\le L_i^f\}\cup\{j\in J \mid E_i^b\le j\le L_i^b\},\ \text{그외의경우}\end{cases}$$



(a) $E_i^b \le L_i^f$ and $E_i^f \le L_i^b$

(b) $E_i^b \ge L_i^f$ (c) $E_i^f \ge L_i^b$

[그림 2] 할당방향에 따른 가장 이른 작업장과 가장 늦은 작업장

2.2 0-1정수계획모형

ULB문제를 위한 0-1정수계획모형에는 다음과 같은 결정변수와 지시변수를 사용한다.

〈결정변수〉

$$x_{ij}=\begin{cases}1, & \text{작업 } i\text{가 작업장 } j\text{에 할당되면,}\\ 0, & \text{그렇지 않으면.}\end{cases}$$

$$y_j=\begin{cases}1, & \text{작업장 } j\text{에 작업이 할당되면,}\\ 0, & \text{그렇지 않으면.}\end{cases}$$

〈지시변수〉

$$z_i=\begin{cases}1, & \text{작업 } i\text{가 선행작업이 모두 할당되어 작업장에 할당되면,}\\ 0, & \text{작업 } i\text{가 후행작업이 모두 할당되어 작업장에 할당되면.}\end{cases}$$

앞에서 언급하였듯이 직선라인과는 다르게 U라인에서는 선행공정도상의 양방향에서 동시에 작업들을 작업장에 할당할 수 있다. 이러한 특성을 반영하여 개발한 ULB문제에서 작업장수 최소화 문제의 0-1정수계획모형은 다음과 같다.

$$\text{Min.} \quad \sum_{j=1}^{N} y_j \tag{0}$$

$$\text{s.t.} \quad \sum_{j\in J(i)} x_{ij}=1 \qquad \forall\, i\in I \tag{1}$$

$$\sum_{i\in I(j)} t_i x_{ij}\le CT \qquad \forall\, j\in J \tag{2}$$

$$\sum_{j\in J(i)} jx_{ij}-\sum_{q\in J(h)} qx_{hq}+u(1-z_i)\ge 0 \quad \text{for all } i\in I,\ h\in P(i) \tag{3}$$

$$\sum_{j\in J(i)} jx_{ij}-\sum_{g\in J(k)} gx_{kg}+uz_i\ge 0 \quad \text{for all } i\in I,\ k\in S(i) \tag{4}$$

$$\sum_{i\in I(j)} x_{ij}-uy_j\le 0 \qquad \forall\, j\in J \tag{5}$$

$$y_j\ge y_{j+1} \qquad \forall\, j\in J-\{N\} \tag{6}$$

$$x_{ij}=0 \text{ or } 1 \qquad \forall\, i\in I, j\in J(i) \tag{7}$$

$$y_j=0 \text{ or } 1 \qquad \forall\, i\in I, j\in J(i) \tag{8}$$

$$z_i=0 \text{ or } 1 \qquad \forall\, i\in I \tag{9}$$

식 (0)는 작업장수 최소화를 나타내는 목적함수이다. 각 작업이 하나의 작업장에 반드시 할당되도록 하는 할당제약(occurrence constraints)은 식 (1)로 표현된다. 식 (2)는 각 작업장의 사이클타임 제약이며, 식 (3)과 (4)는 선행제약(precedence constraints)이다. U라인에서 작업 i의 할당은 작업 i의 모든 선행 또는 후행작업이 할당된 경우에 이루어진다. 식 (3)에서 작업 i의 할당작업장은 직선행 작업 $h(\in P(i))$가 할당된 작업장과 같거나 그 이후 작업장임을 나타내며, 식 (4)에서 작업 i는 직후행 작업 $k(\in S(i))$가 할당된 작업장 또는 그 이후 작업장에 할당가능하다는 것을 의미한다. 그리고 U라인에서 작업장 j에 할당가능한 작업집합은 모든 선행작업들이 이미 할당된 작업집합과 모

든 후행작업들이 이미 할당된 작업집합의 합집합이므로 식 (3)과 (4)는 이러한 관계를 지시변수 z_i를 도입하여 수식화한 것이다. 식 (3)은 전방향으로 작업 i의 할당이 고려될 때만 유효하다. 그렇지 않은 경우 아주 큰 양의 정수 u에 의해 식 (3)은 제약이 되지 않고, 후방향으로 작업 i의 할당이 고려되는 경우는 식 (4)가 유효하다. 식 (5)는 각 작업장에 작업이 할당되면 y_j가 반드시 1이어야 한다는 제약이다. 이 식에서 작업장 j에 작업이 할당되면(즉, $\sum_{i\in I(j)} x_{ij} > 0$ 이면) 부등식을 만족하기 위하여 $y_j = 1$이 되며, 그렇지 않으면(즉, $\sum_{i\in I(j)} x_{ij} = 0$ 인 경우), 목적함수식을 최소화하기 위해서 $y_j = 0$이 된다. 식 (6)은 작업장 j에 적어도 하나의 작업이 할당된 후 다음 작업장 $j+1$에 작업이 할당되게 함으로써 작업이 할당되지 않은 작업장이 없도록 하는 제약이다. 식 (7), (8), (9)는 결정변수와 지시변수를 의미한다. 여기서 구하고자 하는 작업장수는 전체 작업수보다 클 수 없으므로 N은 전체 작업수보다 작지않게 임의로 정한다.

만약 U라인 설계시 라인의 출구와 입구 작업을 동일인이 담당한다는 특성을 고려한다면 수리모형에 첫번째 작업(작업번호 1)과 마지막 작업(작업번호 M)은 첫번째 작업장에 할당한다는 식 (10)을 제약식에 첨가하면 된다.

$$x_{11} = x_{M1} = 1 \qquad (10)$$

3. U라인밸런싱을 위한 열거법

ULB를 위한 분지한계법의 개발을 위하여 먼저 해 공간의 탐색을 위한 열거법을 다룬다. 해의 탐색은 탐색 방법에 따라 크게 깊이우선탐색(depth-first search)과 너비우선탐색(breadth-first search)이 있다. 깊이우선탐색은 한 노드에서 단지 하나의 노드만을 분지하여 해를 찾고 이로부터 역추적(backtracking)하여 다른 해를 탐색하는 방법이고, 너비우선탐색은 현 노드에서 가능한 모든 경우를 분지하면서 해를 탐색해 가는 방법이다. 깊이우선탐색은 가능해를 빨리 구할 수 있고, 한 경로에 있는 노드만을 저장하므로 메모리를 절약할 수 있어 최근 연구에서는 깊이우선탐색이 널리 이용되고 있다[6,8,10].

SLB문제에서 작업장수 최소화를 위한 가능해 열거법으로 FABLE(Fast Algorithm for Balancing Lines Effectively)이 있다[8]. FABLE은 깊이우선탐색에 따라 작업을 작업번호순으로 작업장에 할당한다. 그러나 작업번호순 작업할당은 선행공정도상의 처음 또는 마지막 작업으로부터 시작하여 전방향 또는 후방향 중 한 방향으로 작업을 할당해가는 SLB문제에는 적합하지만, 양방향의 작업할당을 동시에 고려하는 ULB문제에는 적합하지 않다. 따라서 본 연구에서는 ULB문제에서 이러한 문제점을 해결하고, 초기에 좋은 해를 찾기 위하여 할당규칙에 의하여 작업을 할당하는 열거법을 제안한다.

ULB를 위한 열거법은 깊이우선탐색을 이용하며 열거나무(enumeration tree)를 만들 때, 각 노드마다 선후행제약과 사이클타임제약을 만족하는 할당가능 작업집합을 생성한다. 생성된 집

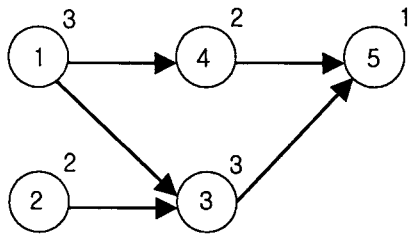
합내에서 할당규칙에 의하여 작업을 선택하여 할당하고, 하나의 가능해가 구해지면 최근 할당된 작업으로부터 역추적하여 다른 가능해를 탐색한다. 이와 같은 방법으로 모든 가능한 경우를 열거하면 종료한다. 할당규칙에 의한 작업할당은 초기에 좋은 해를 탐색할 가능성이 높아, 다음 절에서 기술할 분지한계법을 이용하면 좋지 않은 해들을 조기에 분지를 종료하여 짧은 시간내에 최적해를 탐색할 수 있는 장점을 갖는다.

먼저 열거법에 사용되는 기호를 아래와 같이 정의하기로 하자.

k	: 열거트리에서 하나의 가능해를 나타내는 경로의 깊이
$n(k)$	: 탐색중인 가능해에서 깊이가 k인 노드
$a(k)$	: 노드 $n(k)$에 할당된 작업
$F(k)$	: 노드 $n(k)$의 할당가능 작업집합

또한 어떤 부모노드 $n(k\text{-}1)$에서 분지되는 모든 자식노드들을 분지되는 순서에 따라 $n(k_0)$, $n(k_1)$, $n(k_2)$, ...로 표현하면 이들은 모두 깊이가 k인 노드이다. 만약, 노드 $n(k_0)$에서 역추적한다면 $n(k\text{-}1)$에서 새로 분지되는 노드는 $n(k_1)$가 되며, 이 노드를 '역추적노드'로 부르기로 한다. 그리고 노드 $n(k_1)$이 역추적노드가 될 때 $a(k_0)$는 '역추적된 작업'이라 부르기로 한다. $n(k\text{-}1)$에서 $n(k_2)$도 생성되었다면 이 노드는 $n(k_1)$에서 새로 분지된 역추적노드이며 노드 $n(k_2)$의 역추적된 작업집합은 $\{a(k_0), a(k_1)\}$이다.

[그림 3]의 선행공정도를 이용하여 작업장수 최소화를 목적으로 하는 ULB문제의 가능해 열거법을 설명하기로 한다. [그림 3]에서 원 안의 숫자는 작업번호를 나타내며, 원 밖의 숫자는 작업시간을 나타낸다.



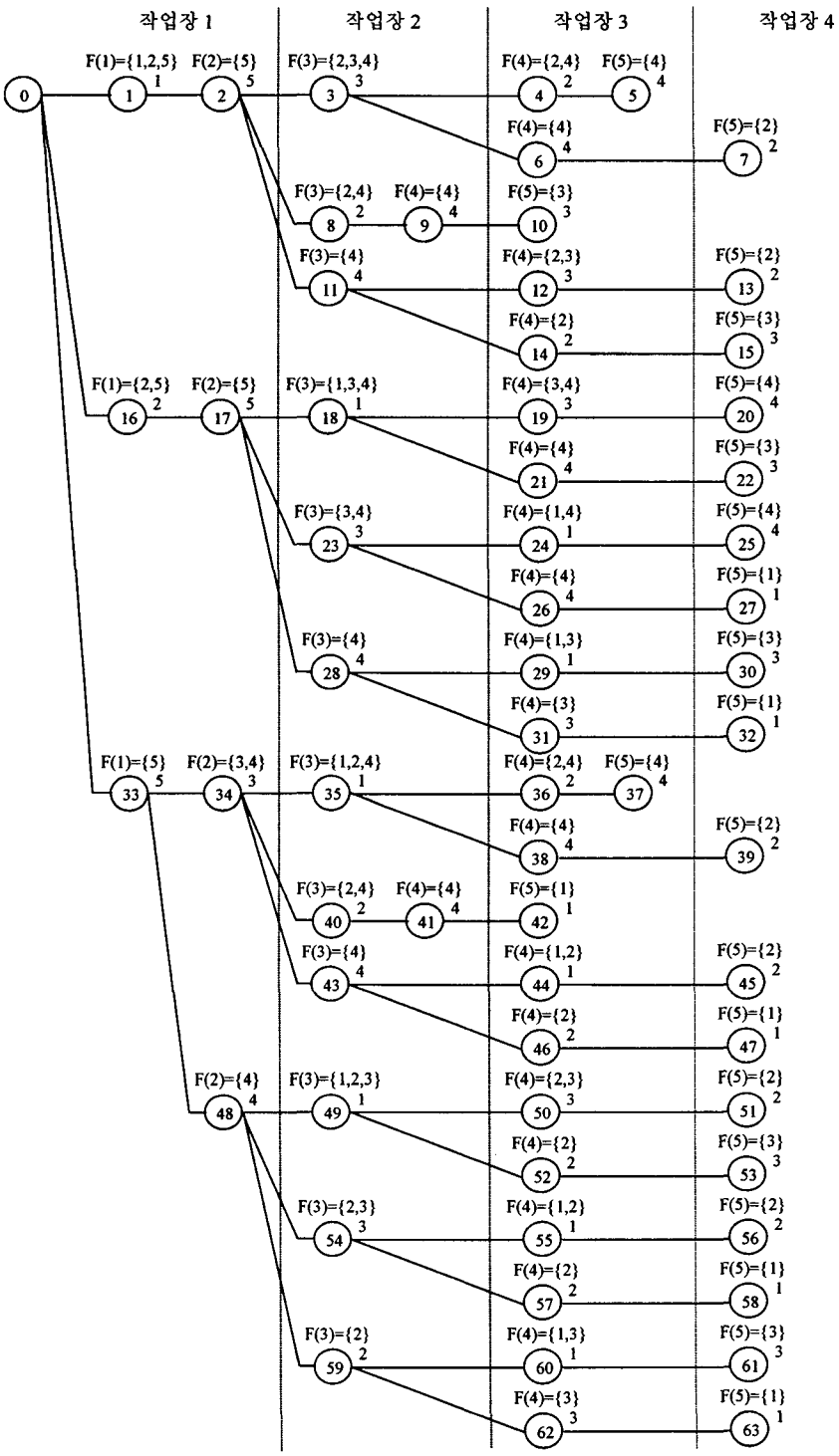
[그림 3] ULB를 위한 선행공정도

[그림 4]는 사이클타임이 4일때, 열거된 모든 가능해를 보여준다. 할당규칙으로는 할당가능 작업집합내에서 작업시간이 최대인 작업을 우선적으로 할당하였고, 동일한 우선순위를 갖는 경우는 작업번호가 낮은 것을 먼저 할당하였다. [그림 4]에서 한 노드에 대하여 원 위의 집합은 해당노드에 대한 할당가능한 작업집합, 원 안의 숫자는 할당순서, 그리고 원 밖의 숫자는 할당된 작업번호를 각각 나타낸다.

3.1 깊이우선탐색

깊이우선탐색에서 할당가능 작업집합과 작업할당은 다음과 같이 진행된다. 깊이 k_0인 노드의 할당가능 작업집합 $F(k_0)$가 있다고 하자. 앞에서 언급하였듯이 U라인에서 할당가능 작업집합이란 모든 선행 또는 후행작업이 할당되고, 사이클타임제약을 만족하는 미할당 작업들의 집합이다.

깊이우선탐색에서 작업장 시작노드의 $F(k_0)$는 모든 선행 또는 후행작업이 할당된 미할당 작업들로 구성된다. 여기에서 미할당작업은 사



[그림 4] 모든 가능해가 열거된 트리

이클타임제약을 항상 만족하므로 선후행관계만을 고려한다.

할당가능 작업집합을 구하면 집합내에서 할당할 작업 $a(k_0)$를 결정한다. 할당할 작업은 우선순위에 따라 선택한다. 할당규칙은 SLB문제에 적용되었던 single-pass 규칙들을 U라인의 특성에 맞게 변형하여 사용할 수 있다[12]. 본 연구에서는 최대작업시간규칙을 사용하기로 한다.

현재의 노드 $n(k_0)$에서, $F(k_0)$로부터 할당규칙에 의해 $a(k_0)$가 할당작업으로 결정되었다고 하자. 그리고 다음 깊이의 노드 $n(k_0+1)$에 작업을 할당한다고 하자. 이때 노드 $n(k_0+1)$의 할당가능 작업집합 $F(k_0+1)$은 다음 두 단계에 의해 생성한다. 첫단계로 (1)$F(k_0+1):=F(k_0)-\{a(k_0)\}$로 두고, $F(k_0+1)$에서 사이클타임제약을 위반하는 작업을 제거한다. 그리고 다음 단계로 (2) 작업 $a(k_0)$의 직선행작업이나 직후행작업중 사이클타임제약과 선후행관계제약을 만족하는 작업을 $F(k_0+1)$에 추가하여 이를 구한다.

새롭게 구한 할당가능 작업집합이 공집합이고 미할당작업이 존재하면, 가능해가 완성되지 않았으나 현 작업장에 할당할 작업이 더 이상 없는 것이다. 그러면 새로운 작업장을 생성하고 이때 미할당 작업중에서 할당가능한 모든 작업을 할당가능 작업집합으로 두고 깊이우선탐색을 계속한다. 그러나 할당가능 작업집합이 공집합이고 미할당작업도 없으면 하나의 가능해가 열거된 것이다. 하나의 가능해가 완성되거나 분지완료되면 마지막 노드로부터 역추적탐색을 한다.

예로, [그림 4]에서 노드 1을 보자. 이 노드는 작업장 1의 시작노드이므로, [그림 3]으로부터 $F(1)=\{1,2,5\}$가 되고, 최대작업시간규칙에 의해 작업 1이 선택되어 $a(1)=1$이 된다. 노드 2의 할당가능 작업집합은 $F(1)$에서 $a(1)=1$을 제거하고, 남은 작업중 사이클타임제약을 위반하는 작업 2를 제거한다. 다음으로 작업 1의 직선행작업이나 직후행작업중 할당가능조건을 만족하는 작업은 없으므로, $F(2)=\{5\}$이다. 할당가능작업은 작업 5뿐이므로, 이를 작업장에 할당한다. 계속하여 노드 3의 할당가능 작업집합을 구하면 공집합이 된다. 따라서 노드 3은 작업장 2로 넘어가게 되며, 작업장 2의 시작노드가 되어 노드 1과 같은 방법으로 작업을 할당한다. 이와 같은 방법으로 하나의 가능해가 완성될 때까지 계속한다.

3.2 역추적탐색

역추적노드에 작업을 할당하기 위해서는 먼저 할당가능 작업집합을 다시 구하고, 이 집합내에서 할당규칙을 사용하여 작업을 할당해 나간다. 역추적노드와 이 노드가 있는 작업장에 할당되는 다른 노드의 할당가능 작업집합의 구성방법은 역추적노드가 있는 작업장에 이미 열거된 작업집합과 똑같은 열거가 일어나지 않도록 해야 한다.

역추적탐색은 다음과 같이 진행된다. 현재 노드 $n(k_0)$에서 역추적한다고 하자. $a(k_0)$를 '역추적된 작업'집합에 보관한다. 그리고 지금까지 깊이 (k_0+1)이후에 역추적된 작업들은 역추적된 작업집합에서 제거된다. 물론 초기 역추적된 작업집합은 공집합으로 둔다. $n(k_0-1)$에서 새로 분지되어 생성되는 노드를 역추적노드 $n(k_1)$이라 하자. 그러면 역추적노드 $n(k_1)$의 할당가능 작업집합 $F(k_1)$은 $F(k_1):=F(k_0)-\{a(k_0)\}$로 두고, $F(k_1)$에 있는 작업중에서 할당규칙에 의하

여 작업을 선택하여 할당한다. 만약 역추적노드의 $F(k_1)$이 공집합이면 $n(k_0\text{-}1)$로 이동하여 탐색이 진행된다.

역추적노드가 생성되고 작업이 할당되면, 역추적노드 다음 노드부터의 작업할당은 다음과 같은 세 단계에 의해 진행된다. 역추적노드가 $n(k_1)$이라면, $F(k_1+1)$을 구하기 위하여 우선, 깊이우선탐색에서 작업장 시작노드 이외의 노드들에 대한 할당가능 작업집합을 구하는 방법과 동일한 단계 (1)과 (2)를 수행한다. 그리고 세 번째 단계로 (3)할당이 진행되는 '현 작업장의 역추적된 작업'들을 제거하여 할당가능 작업집합을 완성한다. 여기서 현 작업장의 역추적된 작업들이란 역추적되어 제거된 작업중에서 깊이 (k_1+1)이전의 노드에서 역추적되어 제거된 작업들이면서 현재의 역추적노드가 있는 작업장에서 열거된 작업들을 말한다. [그림 4]에서 노드 8에서는 노드 3에 할당된 작업 3이 역추적된 작업이며, 노드 11에서 보면 노드 3과 8에 각각 할당된 작업 3과 2가 역추적된 작업들이다. 깊이우선탐색에서는 불필요한 단계 (3)은 역추적된 작업이 동일한 작업장에 다시 할당되어 할당순서는 다르나, 동일한 작업들로 구성되는 해가 열거되는 것을 방지하기 위한 것이다.

그리고 역추적노드로부터 시작하여 이 노드를 갖는 작업장의 할당이 완성되면 다음의 모든 작업장에서의 열거는 깊이우선탐색에서와 동일하게 수행하여 새로운 가능해를 생성한다.

제안한 열거법의 컴퓨터 기억저장측면에서는 할당규칙에 의한 작업할당을 위하여, FABLE처럼 하나의 경로에 있는 각 노드의 할당 작업을 저장할 뿐만 아니라 추가적으로 노드의 할당가능 작업집합을 보관해두어야 한다. 그러나 각 노드의 깊이에서 가장 최근 열거된 노드의 할당가능 작업집합만을 저장하면 된다. 또한 역추적된 작업은 역추적탐색에서 역추적노드가 생성되면 저장되고, 역추적되어 깊이가 감소하면 감소전 깊이 이후에 저장된 역추적된 작업을 삭제하므로 많은 저장용량을 필요로 하지 않는다.

예를 들어, [그림 4]에서 노드 1, 2, 3, 4, 5에 의해 하나의 가능해가 완성되면 노드 5로부터 역추적탐색이 시작된다. 노드 5의 역추적노드에 대한 할당가능 작업집합은 노드 5의 할당가능 작업집합에서 그 노드에 할당된 작업 4를 제거하여 구한다. 그러면 노드 5의 역추적노드의 할당가능 작업집합은 공집합이므로, 역추적탐색은 깊이를 하나 감소하여 앞으로 진행된다. 노드 4에서 역추적노드의 할당가능 작업집합을 구하면 $F(4)=\{4\}$가 되어, 역추적노드 6이 생성된다. 이 노드에 작업 4를 할당하며 역추적된 작업 2를 보관한다. 노드 6의 다음 노드에 대한 할당가능 작업집합을 깊이우선탐색에서 작업장 시작노드가 아닌 경우와 동일하게 단계 (1), (2)에 의하여 구하면 작업 2가 된다. 그러나 작업 2가 작업장 3에 할당되면, 이전에 구한 가능해의 작업장 3에 할당된 작업들이 2와 4이므로 열거에 의해 순서만 바뀐 결과가 되어 해의 중복탐색이 발생된다. 그러므로 단계 (3)에 의해 역추적된 작업 2를 제거하면 노드 6의 다음 노드에 대한 할당가능집합은 공집합이 된다. 깊이우선탐색에서와 마찬가지로 할당가능 작업집합이 공집합이고, 미할당작업이 있으면 다음 작업장에 대하여 할당을 계속한다. 이 예에서는 작업 2가 미할당 작업이므로 작업 2는 새로운 작업장 4에 할당된다.

3.3 열거절차

사이클타임이 주어진 작업장수 최소화를 위한 ULB문제의 가능해 열거절차는 다음과 같다. S_j는 작업장 j의 여유시간, $B_j(k)$는 작업장 j의 깊이 k에서 역추적되었던 작업들을 저장하는 집합이다. 그리고 s는 작업장 j의 시작노드 깊이를 나타낸다.

단계 1. (초기화)
$j:=1$, $k:=1$, $ST_j:=CT$,
$backtrack:=0$

단계 2. (할당가능 작업집합생성 : 작업장의 시작노드인 경우)
(a) $P_a(i)$ 또는 $S_a(i)$가 이미 할당된 작업집합 $F(k)$를 구한다.
(b) $B_j(k)=\phi$
(c) $F(k)=\phi$이면 단계 5로 간다.

단계 3. (작업할당과 여유시간 갱신)
(a) 할당규칙을 만족하는 작업 i를 j에 할당하고, $a(k):=i$, $ST_j:=ST_j-t_j$로 갱신한다.
(b) $k:=k+1$으로 두고 $k>M$이면 단계 6으로 간다.

단계 4. (할당가능 작업집합생성 : 역추적노드 또는 작업장의 시작노드가 아닌 경우)
(a) $F(k):=F(k-1)-\{a(k-1)\}$ 로 하고, $F(k)$에서 사이클타임제약을 만족하지 못하는 작업을 제거한다.
(b) $a(k-1)$의 직선행작업이나 직후행작업중 선행제약과 사이클타임제약을 마족하는 작업을 $F(k)$에 추가한다.
(c) 만약, $backtrack=1$이면,
$F(k):=F(k)-\{B_j(s)\cup B_j(s+1)\cup\ldots\cup B_j(k-1)\}$
(d) $F(k)\neq\phi$이면 단계 3으로 간다.

단계 5. (새로운 작업장 생성)
$j:=j+1$, $ST_j:=CT$, $backtrack:=0$으로 두고 단계 2로 간다.

단계 6. (새로운 해 완성)
새로운 해가 완성된다. 만일 새로운 해가 현재해보다 좋으면, 새로운 해를 현재해로 갱신한다.

단계 7. (역추적)
(a) 만약, $ST_j=CT$이면 $j:=j-1$로 둔다.
(b) $B_j(k):=\phi$, $k:=k-1$로 둔다.
(c) 작업장 j에서 $a(k)$를 제거하고 $ST_j:=ST_j+t_{a(k)}$으로 갱신한다.

단계 8. (할당가능 작업집합생성 : 역추적노드인 경우)
(a) $F(k):=F(k)-\{a(k)\}$
(b) $F(k)=\phi$이고 $k=1$이면 알고리듬을 종료한다.
(c) $F(k)=\phi$이고 $k>1$이면 단계 7로 간다.
(d) $F(k)\neq\phi$이면 $B_j(k)=B_j(k)\cup\{a(k)\}$, $backtrack:=1$로 두고 단계 3으로 간다.

할당가능 작업집합은 해의 탐색방향 또는 작업의 할당시점에 따라 각각 단계 2, 4, 8에서 구하며, 작업할당은 단계 3에서 이루어진다. 또한 할당가능 작업집합 생성시 현 작업장에서 역추적된 작업의 제거는 역추적이 이루어진 경우만 확인하면 되므로 *backtrack*의 값을 이용하여 깊이우선탐색과 역추적탐색을 구분하였다.

ULB문제에서 작업번호순으로 작업을 할당하

면 앞의 예에서는 [그림 3]의 노드 33과 34, 33과 50과 같이 작업번호의 역순으로 구성되는 작업장이 형성될 수 없다. 그리고 라인의 입구와 출구가 동일한 작업장에 할당되어야 한다는 조건을 추가하고자 한다면, 제안한 알고리듬에서 단순히 첫번째 작업장에 이들 두 작업을 우선 할당한 후 첫번째 작업장의 사이클타임은 초기 사이클타임에서 이들의 작업시간을 뺀 값으로 두고 알고리듬을 진행하면 된다.

4. U라인밸런싱을 위한 분지한계법

분지한계법은 적절한 분지전략과 한계전략, 그리고 분지완료규칙을 사용하여 가능해를 가능한 적게 열거하면서 최적해를 구하는 부분열거방법이다. 본 연구에서 분지전략은 깊이우선탐색을 사용하며, 본 절에서는 한계전략과 분지완료규칙들을 제시한다.

4.1 한계전략

한계전략은 문제에 따라 상한(upper bound: *UB*) 또는 하한(lower bound: *LB*)을 이용하여 현재 탐색중인 해의 분지완료여부를 결정하고, 해의 수렴을 촉진시킨다. 다루는 문제에서는 상한과 하한을 동시에 사용한다.

*UB*는 현재까지 구한 가장 좋은 해(최선해)의 작업장 수로 둔다. 이 값은 가능해의 탐색과정에서 현재까지의 최선해보다 더 좋은 해를 유도할 수 없는 열거는 하지 않도록 하는데 사용된다. 그리고 열거나무 생성절차를 진행하는 동안 현재까지의 최선해가 갱신되면, 상한 역시 갱신된다. *LB*는 하나의 작업이 할당될 때마다 할당이 결정된 작업장수와 미할당된 작업들의 최소 작업장수를 더하여 구한 값보다 크거나 같은 최소의 정수로 둔다. 여기서 할당이 결정된 작업장수는 정수가 아닐 수 있다. 예로, 사이클타임이 10일 때 3개의 작업장이 구성된 후 그 다음 작업장에 작업시간이 4인 작업이 할당되었다면 할당이 결정된 작업장수를 3.4로 둔다. 그리고 미할당된 작업들의 최소 작업장수는 다음 세 가지 방법에 의해 구한 값의 최대값으로 둔다.

첫번째 방법은 단순히 대상 작업들의 총작업시간을 사이클타임으로 나누어 구한 값으로 둔다. 나머지 두 방법은 Johnson[8]의 연구에서 사용되었다. 두번째 방법은 작업시간이 사이클타임의 1/2보다 큰 작업들은 반드시 한 작업장에 하나씩 할당되고, 작업시간이 사이클타임의 1/2과 같은 작업들은 선행제약조건이 무시될 때, 2개의 작업이 한 작업장에 할당될 수 있음을 이용하여 요구되는 미할당된 작업들의 최소 작업장수를 결정한다. 이 방법에서 작업시간이 사이클타임의 1/2보다 적은 작업들은 하한의 계산에 반영되지 않는다. 세번째 방법은 작업시간이 사이클타임의 1/3보다 크면 단지 두 작업이 한 작업장에 할당될 수 있으며 이 작업들은 작업시간이 사이클타임의 2/3보다 큰 작업과 같은 작업장에 할당될 수 없음을 이용하여 하한을 계산한다. 또한 작업시간이 정확히 사이클타임의 1/3과 2/3인 작업에 의해 요구되는 작업장 수는 증가될 수 있다.

한계전략은 하나의 가능해가 완성되거나 탐색중인 해에서 하나의 작업이 할당될 때마다 사

용한다. 즉, 가능해 열거과정에서 한 작업이 할당될 때마다 LB를 구하여 $LB \geq UB$이면 분지완료한다. 그리고 모든 작업을 미할당작업으로 두고 구한 LB를 LB^*로 둘 때, 완성된 현재해의 작업장수가 LB^*와 같으면 최적해이므로 열거절차를 끝낸다. 또한 한 작업장이 완성된 경우도 작업할당이 이루어진 경우를 포함하므로 한계전략을 적용한다.

4.2 분지완료규칙

ULB문제는 SLB문제와 비교할때 단지 할당가능 작업집합에 차이가 있으므로, SLB문제의 여러 분지완료규칙들은 ULB문제에도 대부분 적용가능하다. 본 연구에서 사용하는 분지완료규칙들은 다음과 같다.

1) Jackson 지배규칙1

Jackson지배규칙1은 최근 생성된 작업장이 이미 생성된 임의의 작업장에 대한 부분집합이면 그 작업장 이후는 분지완료하는 규칙이다. 제안한 열거법에서 역추적된 작업들은 역추적노드가 있는 작업장의 작업할당에서 제외된다. 따라서 제안한 열거법에서는 하나의 작업장을 완성한 후 역추적된 작업에 의해 할당가능 조건을 만족하지만 할당되지 않은 작업이 존재할 수 있다. 이 경우 최근 완성된 작업장은 과거 생성된 임의의 작업장에 대한 부분집합으로 구성되어 이 지배규칙이 적용될 수 있다. 깊이우선탐색에서 이 지배규칙은 다음과 같은 방법으로 확인한다. 작업장이 완성된 후, 미할당작업중 사이클타임제약과 선후행제약조건을 만족하면서 최근 완성된 작업장의 할당량이 증가될 수 있는 작업장은 분지완료한다. [그림 4]의 예에서는 노드 6에 적용된다. 노드 6에서 작업 4를 할당한 후 할당가능조건을 만족하는 작업2는 작업장 3에 할당되지 않았다. 따라서 노드 7은 분지완료하며, 노드 6에 의해 생성된 작업장은 노드 4와 5에 의해 생성된 작업장의 부분집합임을 알 수 있다.

2) Jackson 지배규칙2

Jackson 지배규칙2는 두 작업 i와 k에 대하여 $t_i \leq t_k$이고, $S_a(i) \subset S_a(k)$이면 작업 k가 작업 i를 지배한다는 것이다. 이 지배규칙을 후방향 작업할당에 적용하면 $t_i \leq t_k$이고, $P_a(i) \subset P_a(k)$이면 작업 k가 작업 i를 지배하는 것으로 변형된다. ULB에서 할당가능 작업집합은 모든 선행 또는 후행작업이 할당된 작업이므로 전방향 또는 후방향 작업할당에 사용되는 지배규칙이 모두 적용된다는 특징을 갖는다. 따라서 ULB를 위한 Jackson 지배규칙2는 $t_i \leq t_k$이고, $S_a(i) \subset S_a(k)$이거나 $P_a(i) \subset P_a(k)$이면, 작업 k가 작업 i를 지배하는 것이 된다. 깊이우선탐색에 이를 적용하기 위하여 지배관계를 갖는 모든 작업을 열거절차를 시작하기 전에 확인해둔다. 그리고 각 작업장의 할당이 완성되면 이 지배규칙의 적용여부를 판단한다. [그림 3]의 선행공정도에서 작업 1과 3은 각각 작업 2와 4를 지배한다. [그림 4]의 노드 17에 의해 완성된 작업장 1에서 작업 2대신 작업 1의 할당이 가능하고, 노드 48에서 작업 4와 작업 3의 교체가 가능하므로 각각 노드 17과 48이후의 노드는 분지완료한다.

3) 첫번째 작업장 지배규칙

첫번째 작업장 지배규칙은 한번 첫번째 작업장에 할당된 작업집합이 열거를 진행하는 동안

첫번째 작업장이 아닌 열거나무의 다른 위치에서 또다시 열거되거나 그 부분집합이 생성되면 두번째 출현한 노드는 분지완료시키는 규칙이다. [그림 4]에서 노드 18은 하나의 작업장을 구성한다. 노드 18에서 할당된 작업집합, 즉 작업 1은 노드 1과 2에 의해 생성된 첫번째 작업장의 부분집합이므로 분지완료된다. 이 지배규칙 역시 작업장의 할당이 완성된 후 판단하며 첫번째 작업장에 할당된 작업을 메모리에 저장해야 하는 단점이 있으나, 이 규칙을 적용함으로써 작업장 번호가 낮은 곳에서 분지완료될 가능성이 높아 분지완료의 효과가 매우 크다.

4.3 분지한계법 절차

ULB를 위한 분지한계법은 아래와 같은 절차에 의하여 진행된다.

(1) 입력자료를 읽는다.

(2) Jackson 지배규칙 2의 가능성이 있는 작업쌍들을 확인한다.

(3) 모든 작업을 미할당작업으로 두고 하한 LB^*를 구한다.

(4) 다음 규칙들과 함께 열거나무 생성절차에 의하여 가능해를 열거한다.

(4-1) 단계 3에서 작업을 할당한 후, 한계전략이 적용된다. $LB \geq UB$이면 분지완료하고 단계 7로 이동한다.

(4-2) 단계 5에서 새로운 작업장을 생성하기 전에 한계전략과 분지완료규칙들을 적용한다. 만약 이들이 성공적으로 수행되어 분지완료되면 단계 7로 이동한다.

(4-3) 단계 6에서 완성된 새로운 해의 작업장수가 LB^*와 같으면 새로 구한 해가 최적해이므로 알고리듬을 종료한다. 그렇지 않은 경우, 모든 가능해가 열거된 후 저장된 현재까지의 최선해가 최적이다.

4.4 발견적 분지완료규칙

일반적으로 분지한계법을 이용하여 최적해를 구하는데 소요되는 계산시간은 아주 짧거나 아주 긴 시간이 소요되는 이분법적인 경향이 있다[6]. 따라서 일정시간이 경과한 후, 최적해를 구하지 못하면 계산시간의 단축을 위해 아래와 같이 좀더 강화된 분지완료규칙을 사용한다. 물론 이와 같은 발견적 분지완료규칙은 해의 질을 떨어뜨릴 수 있다.

발견적 분지완료규칙 1: 역추적노드의 최대 분지수를 제한하는 방법이다. 제안한 열거법은 할당규칙을 사용하여 가능한 초기에 좋은 해를 찾도록 유도하고 있다. 따라서 역추적되어 늦게 생성되는 해일수록 좋은 해를 유도하지 못할 가능성이 높다. 이 규칙은 해의 질에 크게 손상을 주지 않을 수 있다.

발견적 분지완료규칙 2: 작업장 번호가 증가함에 따라 탐색공간을 줄이는 방법이다. 현재 할당되는 작업장을 p라 할때, UB와 작업장번호에 따라 $[\{(UB-p+1)/UB\}\times|F(k)|]^+$개의 작업만 분지하기로 한다. 이 규칙은 뒷 작업장은 앞 작업장에 비해 다양한 해로의 유도 가능성이 낮다는 성질을 이용한 것이며, 제안한 알고리듬은 좋은 초기해를 유도하므로 작업장 번호가 증가함에 따라 분지 노드수를 줄이더라도 비교적 좋은 해를 탐색할 수 있다.

발견적 분지완료규칙 3: 각 작업장의 작업할당이 완성된 후 현재까지 소모된 여유시간에 근거하여 현재해보다 더 좋은 해를 유도할 가능성이 낮은 경우 분지완료하는 규칙이다. 현재의 상한에서 1을 뺀 작업장수를 '개선해'라 하자. 그리고 현재 탐색이 작업장 p까지 완성되었다고 하자. 또한 작업장의 여유시간을 S_j, $j=1,2,...,p$로 두자. 개선해에서 허용되는 총 여유시간은 $TST=\{(UB-1)\times CT\}-\sum_{i=1}^{M} t_i$ 이고, 작업장당 평균 여유시간은 $AST=TST/(UB-1)$이다. 그러면 개선해를 기준으로 남아있는 작업장당 평균 여유시간은 $RAST=(TST-\sum_{j=1}^{p} S_j)/(UB-1-p)$가 되고 남아있는 평균여유율은 $sr=RAST/AST$이 된다. 이때 sr이 낮을수록 여유시간을 많이 소모한 것이 되므로 미리 정한 값 a(단, $a<1$)보다 작으면 좋은 해로의 유도 가능성이 낮다고 볼 수 있다. 그러나 작업할당이 완성된 작업장이 많으면 많을수록 여유시간의 소모량이 많더라도 좋은 해로의 유도가능성이 있다고 본다. 따라서 작업할당이 완성된 작업장 번호가 증가할수록 분지완료될 가능성을 낮게 하고자 한다. 이를 위하여 $\beta=a\{(UB-p)/(UB-1)\}$을 사용하고, $sr<\beta$이면 분지완료한다.

이 밖에 분지한계법을 진행하는데 소요되는 누적 수행시간이 일정시간을 초과하면 현재해를 근사최적해로 하여 알고리듬을 종료할 수 있으며, 제안한 발견적 분지완료규칙을 모두 혼합하여 사용할 수도 있다.

또한, U라인은 라인의 첫번째 작업과 마지막 작업을 첫 작업장에서 한 작업자가 수행함으로써 많은 장점을 갖는다. 따라서 첫 작업장에 할당되는 작업을 고정하면 상당히 많은 노드가 생성되지 않아 분지되는 노드수를 줄일 수 있다. 이때 첫번째 작업과 마지막 작업은 선행작업이 없는 작업과 후행작업이 없는 작업을 나타내며, 사이클타임제약을 만족하는 이러한 작업이 각각 하나 이상 존재할 경우 할당규칙에 사용되는 우선순위에 따라 할당작업을 결정할 수 있다.

5. 실험 및 분석

본 연구에서 제시한 알고리듬의 성능 분석을 위하여 라인밸런싱문제에서 벤치마크문제로 알려진 문제를 대상으로 해의 효율과 계산시간을 실험하였다. 작업장수가 많은 경우를 실험하기 위하여 Tonge문제[15]에서 작업 18의 작업시간을 319에서 143, Bartholdi문제[2]는 작업 79의 작업시간은 281에서 111, 작업 108의 작업시간은 383에서 43으로 작업시간이 너무 큰 작업의 작업시간을 조정하였다.

실험에 사용한 할당규칙은 작업시간이 최대인 작업을 선택하였으며, 최대 작업시간을 갖는 작업이 하나 이상 존재하면 작업번호가 낮은 것을 우선적으로 할당하였다. 제안한 알고리듬은 C++언어를 사용하여 작성하였고, Pentium 166MHz를 가진 IBM-PC에서 실험을 수행하였다. 〈표 1〉의 실험결과를 얻었다. 〈표 1〉에서 세번째 열은 총 작업시간을 주어진 사이클타임으로 나눈 이론적인 최소작업장수이다. 네번째 열은 본 연구에서 제안한 분지한계법에 의해 구한 작업장 수이며, 다섯번째 열은 계산소요시간으로 CPU시간이다.

실험결과, 거의 모든 문제에서 이론적인 작업

〈표 1〉 해의 효율과 계산시간 비교

실험 문제		사이클타임	이론적인 최소작업장수	최적작업장수	계산시간(초)
Bowman[3]	9	20	3.75	4	0.01
Jackson [12]	11	7	6.6	7	0.00
		9	5.1	6	0.00
		10	4.6	5	0.01
		13	3.5	4	0.00
		14	3.3	4	0.00
		21	2.2	3	0.00
Dar-El [11]	11	48	3.8	4	0.01
		62	3.0	3	0.01
		94	2.0	2	0.00
Mitchell [16]	21	14	7.5	8	0.01
		15	7.0	8	0.02
		21	5.0	5+	0.02
Sawyer [4]	30	25	12.96	14	0.17
		27	12.00	13	3.08
		30	10.80	11+	0.01
		36	9.00	9+	0.05
		41	7.90	8	0.06
		54	6.00	6+	0.08
		75	4.32	5	0.01
Kilbridge & Wester [9]	45	57	9.68	10	0.01
		79	6.99	7	2.19
		92	6.00	6	2.78
		110	5.02	6	0.01
		138	4.00	4	10.03
		184	3.00	3	16.23
Tonge [15]	70	176	19.94	20+	> 30.00
		364	9.64	10	0.01
		410	8.56	9	0.03
		468	7.50	8	0.02
		527	6.66	7	0.02
Arcus [1]	83	5048	14.99	16	0.94
		5853	12.93	13+	6.33
		6842	11.06	12	0.03
		7571	9.99	11	5.88
		8412	8.99	10	5.08
		8898	8.51	9	0.02
		10816	6.99	7+	28.23
Arcus [1]	111	5755	26.15	27+	0.08
		7969	18.89	19	0.14
		8847	17.01	18	0.06
		9018	16.69	17	0.06
		10027	15.01	16	0.06
		10743	14.01	15	0.06
		11378	13.23	14	0.06
		17067	8.82	9	0.09
Bartholdi [2]	148	193	26.54	27	0.13
		238	21.52	22	0.11
		272	18.83	19	0.11
		289	17.72	18	0.09
		306	16.74	17	0.09
		323	15.86	16	0.17
		525	9.76	10	0.09

장수가 최적임을 알 수 있으며, 특히 '+' 로 표시된 해는 SLB문제에서 알려진 최적해보다 작업장 수가 적은 경우를 나타낸다. 이것은 앞에서 언급하였듯이 U라인이 작업할당에 대한 유연성이 높음으로 인하여 가능한 것으로 보인다. 계산시간면에서 볼 때, Tonge의 사이클타임이 176인 문제는 다른 문제에 비하여 할당가능 작업집합의 작업수가 많고 분지완료되는 경우가 적어 계산시간이 많이 소요되었다. 그러나 나머지 문제는 비교적 적은 시간내에 최적해를 찾았으며, 특히 계산시간이 1초미만인 경우는 초기해가 최적해인 경우로 본 연구에서 제안한 가능해 열거법이 좋은 초기해를 구할 수 있다는 특성을 보여준다.

다음으로 발견적 분지완료규칙의 성능을 실험하였다. 실험은 분지한계법 적용에서 계산시간이 3초이상 소요된 문제만을 대상으로 제안한 방법 1, 2, 3에 대하여 실시하였다. 발견적 분지완료규칙의 파라메터로 방법 1의 m은 2, 방법 3의 a는 0.5로 두었고, 실험결과는 〈표 2〉와 같으며, '작업장수(계산시간)'로 나타내었다. 제안한 발견적 분지완료규칙들은 많은 문제에서 최적해를 찾았으며, 특히 분지되는 가지수를 제한하는 규칙 1이 좋은 근사최적해를 구하면서도 계산시간 단축에 크게 기여하는 것으로 나타났다. 그러나 몇몇 결과에서 알 수 있듯이 발견적 분지완료규칙을 적용하면 해의 탐색방향을 변화시켜, 좋은 해를 탐색하지 못하면서 더 많은 계산시간이 소요되는 경우도 발생될 수 있다. 따라서 반드시 발견적 분지완료규칙이 계산시간을 단축하는 효과를 갖는다고 할 수는 없다.

6. 결 론

본 연구에서는 U자형 조립라인의 라인밸런싱 문제에서 작업장수 최소화를 목적으로 하는 0-1 정수계획모형과 분지한계법을 제안하였다. 정수계획모형에서는 작업할당변수와 제약식수를 감

〈표 2〉 발견적 분지완료규칙의 성능

실험문제		사이클타임	최적작업장수	규칙1		규칙2		규칙3	
Sawyer	30	27	13	13	(0.05)	13	(3.30)	13	(0.22)
Kilbridge & Wester	45	138	4	4	(0.77)	4	(10.33)	4	(9.11)
		184	3	3	(1.05)	3	(17.03)	3	(16.48)
Tonge	70	176	20	20	(>30.0)	20	(>30.0)	21	(>30.0)
Arcus	83	5853	13	13	(5.54)	13	(7.30)	13	(6.59)
		7571	11	11	(0.44)	11	(6.31)	11	(5.99)
		8412	10	10	(0.50)	10	(5.06)	10	(5.27)
		10816	7	8	(3.29)	7	(28.73)	8	(>30.0)

소시키기 위한 방법을 제시하였고, 분지한계법에서는 우선순위에 의한 할당규칙으로 작업을 할당하는 가능해 열거법과 적절한 한계전략과 분지완료규칙을 제안하였다. 또한 계산소요시간을 많이 요구하는 문제의 해를 적은 시간내에 구하기 위한 발견적 분지완료규칙을 개발하였다.

제안한 열거법은 할당규칙을 사용하므로 좋은 초기해를 구할 수 있고, 하나의 할당가능 작업집합으로 전방향 작업할당과 후방향 작업할당을 동시에 고려할 수 있다는 장점을 갖는다. 그러나 할당된 작업만을 저장하는 FABLE보다 더 많은 메모리를 차지한다는 단점을 갖는다. 즉, 할당규칙을 사용하기 위하여 각 노드에 할당되는 작업들과 함께 그 노드에 할당가능 작업집합과 역추적되어 제거된 작업을 저장해야 한다. 따라서 약간의 메모리를 더 요구하게 된다

제안한 알고리듬의 성능평가를 위하여 해의 효율과 계산시간면에서 실험하였다. 실험결과, 제안한 알고리듬은 대부분의 문제에서 적은 시간내에 최적해를 찾았다. 또한 제안한 발견적 분지완료규칙 역시 해의 성능과 계산시간면에서 우수함을 보였다. 본 연구결과는 효율적이고 합리적인 작업편성으로 조립라인의 균형효율을 높여 생산율을 향상시킬 수 있을 것으로 기대된다. 그리고 본 연구를 확장한 혼합모델 조립라인 또는 복수 U라인에서의 라인밸런싱에 관한 연구가 요청된다.

참 고 문 헌

[1] Arcus, A.L., "COMSOAL: A Computer Method of Sequencing Operations for Assembly Lines," *International Journal of Production Research*, Vol.4(1966), pp.259-277.

[2] Bartholdi, J.J., "Balancing Two-sided Assembly Lines: A Case Study," *International Journal of Production Research*, Vol.31 (1993), pp.2447-2461.

[3] Bowman, E.M., "Assembly Line Balancing by Linear Programming, *Operations Research*, Vol.8(1960), pp.385-389.

[4] Buxey, G.M., "Assembly Line Balancing with Multiple Stations," *Management Science*, Vol.20(1974), pp.1010-1021.

[5] Hackman, S.T., M.J. Magazine, and T.S. Wee, "Fast, Effective Algorithms for Simple Assembly Line Balancing Problems," *Operations Research*, Vol.37(1989), pp.916-924.

[6] Hoffmann, T.R., "Eureka: A Hybrid System for Assembly Line Balancing," *Management Science*, Vol.38(1992), pp.39-47.

[7] Hwang, H., J.U. Sun, and T.H. Yoon, "U-line Line Balancing with Simulated Annealing," *Proceedings of the First ASIA-PACIFIC Decision Sciences Institute Conference*, Hong Kong, (1996), pp.101-108.

[8] Johnson, R.V., "Optimally Balancing Large Assembly Lines with 'FABLE'," *Management Science*, Vol.34(1988), pp.240-253.

[9] Kilbridge, M.D. and L. Wester, "A Heuristic Method of Assembly Line Balancing Problem," *The Journal of Industrial Eng-*

ineering, Vol.12(1961), pp.292-298.

[10] Klein, R. and A. Scholl, "Maximizing The Production Rate in Simple Assembly Line Balancing A Branch and Bound Procedure," *European Journal of Operational Research*, Vol.91(1996), pp.367- 385.

[11] Mansoor, E.M., "Assembly Line Balancing -An Improvement on the Ranked Positional Weight Technique," *Journal of Industrial Engineering*, Vol.15(1964), pp.73-77.

[12] Miltenburg, G.J. and J. Wijngaard, "The U-line Line Balancing Problem," *Management Science*, Vol.40(1994), pp.1378-1388.

[13] Monden, Y., *Toyota Production System* (2nd Ed.), Industrial Engineering and Management Press, Institute of Industrial Engineers, Norcross, GA, 1993.

[14] Patterson, J.H. and J.J. Albracht, "Assembly Line Balancing: Zero-one Programming with Fibonacci Search," *Operations Research*, Vol.23(1975), pp.166-172.

[15] Tonge, F.M., *A Heuristic Program for Assembly Line Balancing*, Prentice-Hall Englewood Cliffs, NJ, 1961.

[16] _____, "Assembly Line Balancing Using Probabilistic Combinations of Heuristics," *Management Science*, Vol.11(1965), pp.727-735.