Journal of the Korean Institute
of Industrial Engineers
Vol.24, N0.3, September, 1998

분지한계법을 이용한 양면조립라인 밸런싱*

Two-sided assembly line balancing using a branch-and-bound method*

김여근** · 이태옥** · 신태호***

Yeo Keun Kim** · Tae Ok Lee** · Tae Ho Shin***

Abstract

This paper considers two-sided (left and right side) assembly lines which are often used, especially in assembling large-sized products such as trucks and buses. A large number of exact algorithms and heuristics have been proposed to balance one-sided lines. However, little attention has been paid to balancing two-sided assembly lines. We present an efficient algorithm based on a branch and bound for balancing two-sided assembly lines. The algorithm involves a procedure for generating an enumeration tree. To efficiently search for the near optimal solutions to the problem, assignment rules are used in the method. New and existing bound strategies and dominance rules are also employed. The proposed algorithm can find a near optimal solution by enumerating feasible solutions partially. Extensive computational experiments are carried out to make the performance comparison between the proposed algorithm and existing ones. The computational results show that our algorithm is promising and robust in solution quality.

1. 서론

조립라인밸런싱은 특정 목적에 맞도록 작업장에 작업을 할당하는 문제와 관련된다. 조립라인은 작업장의 배치에 따라 단면조립라인(one-sided assembly line)과 양면조립라인(two-sided assembly line)으로 분류할 수 있다. 단면조립라인은 작업장이 라인의 왼쪽 또는 오른쪽의 한면에 일렬로 배치되는 라인이고, 양면조립라인은 작업장이 그림 1과 같이 라인의 왼쪽과 오른쪽의 양면에 병렬로 배치되는 라인이다. 작업장 1과 2, 작업장 3과 4와 같이 마주보는 두 작업장을 '작업장쌍'이라 부르기로 하고, 작업장쌍에서 맞은 편에 있는 작업장을 '상대작업장'이라 부르기로 한다. 본 연구는 양면조립라인 밸런싱문제를 다룬다.

양면조립라인은 트럭이나 버스처럼 조립 대상제품이 크고 작업 대칭성(왼쪽 또는 오른쪽)이 높은 제품의 생산라인에서 흔히 사용된다. 즉 양면조립라인은 많은 대칭작업이 존재하고 작업자간 작업간섭이 발생하지 않는 경우에

* 이 논문은 1997년도 전남대학교 학술연구비 지원에 의하여 연구되었음

** 전남대학교 산업공학과

*** 순천제일대학 품질경영과

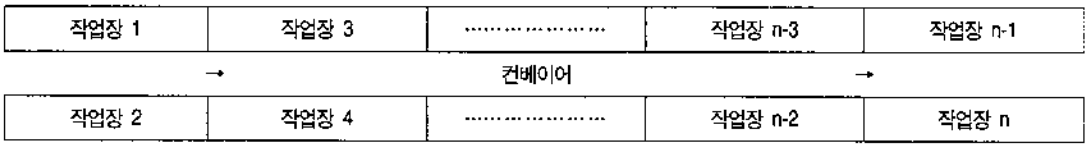


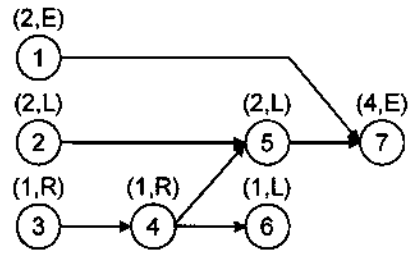
그림 1. 양면조립라인의 구조

설치된다. 예로 A 자동차회사의 트럭라인에서 연료탱크, 공기정화기, 공구상자 장착 작업은 왼쪽에서, 배터리, 공기탱크, 소음기 장착은 오른쪽에서 작업을 해야 한다. 그리고 액슬(axle), 프로펠러 샤프트(propeller shaft), 라디에이터 장착 등은 왼쪽 또는 오른쪽 어느 작업자가 하여도 작업 편의상 차이가 없는 작업들이다. 양면조립라인의 효율을 높이기 위해서는 작업할당이나, 설비 및 치공구의 배치에 있어서 작업의 방향이 고려되어야 한다.

작업 대칭성이 높고 작업자간 작업간섭이 일어나지 않는 공간이 허용되는 경우, 양면조립라인은 여러 잇점을 갖는다. 양면조립라인은 단면조립라인보다 라인길이를 줄일 수 있고 제품의 완성 소요시간(throughput time)을 단축할 수 있다. 그리고 설비나 치공구를 양면에서 공유할 수 있어 비용을 절감할 수 있으며, 조립위치(왼쪽 또는 오른쪽)에 따른 작업자의 이동과 작업 준비시간을 줄일 수 있다.

양면조립라인의 작업할당은 단면조립라인에서와는 달리 상대작업장의 작업에 의해 한 작업장에 할당된 작업간에도 유휴시간이 발생할 수 있다. 이로 인해 양면조립라인에서 사이클타임제약은 각 작업장의 마지막 작업의 완료시점이 사이클타임보다 크지 않다가 된다. 예로 그림 2의 작업들을 단면조립라인 밸런싱기법을 적용하여 작업장 1, 2, 3, 4에 각각 작업집합 {1, 2}, {3, 4}, {5, 6}, {7}이 할당되었다고 하자. 작업 7은 상대작업장에 할당된 작업 5에 의해 2단위시간의 유휴시간이 발생하게 되고, 이로 인해 작업 7의 완료시점이 6이 되어 사이클타임제약을 어기게 된다. 이로부터 기존의 단면조립라인 밸런싱기법을 양면조립라인 밸런싱문제에 적용하는 데는 한계가 있음을 알 수 있다.

Baybars[2]는 조립라인에서 단일모델이 생산되고 작업시간이 확정적인 조립라인밸런싱의 최적해(exact solution) 모형에 관한 연구현황을 조사하였다. 그러나 조립라인밸



사이클타임(CT) = 5
(p,q): p는 작업시간, q는 작업방향
L: 왼쪽 방향, R: 오른쪽 방향, E: 제약없음

그림 2. 작업의 선행공정도

런싱은 NP-hard 문제에 속하므로[2], 주로 빠른 시간에 근사최적해를 구하는 발견적기법에 관하여 연구가 이루어졌다. Talbot *et al.*[10]은 기존의 여러 발견적기법들을 평가하고 분석하였다. Ghosh와 Gagnon[3]은 조립라인의 설계, 균형화, 일정계획 등에 관한 기존의 연구들을 종합적으로 분석·검토 하였다. 분지한계법에 관한 연구로는 Van Assche와 Herroelen[11], Johnson[6,7], Klein과 Scholl[8]의 연구가 있다. Johnson[6]의 연구에서는 여러 제약하에서 작업장 수의 최소화를 위한 분지한계법을 제시하였다. 그 후 Johnson[7]은 분지한계법에 여러 분지완료규칙을 사용한 FABLE(Fast Algorithm for Balancing Lines Effectively)을 제시하였다. 이러한 연구들은 사이클타임이 주어진 경우 작업장 수를 최소화 하는 연구들이다. 최근에 Klein과 Scholl[8]은 사이클타임을 최소화 하는 조립라인밸런싱문제를 풀기 위하여 새로운 열거방법인 local lower bound method를 제시하였다. 이들 모든 연구는 단면조립라인을 다루고 있다. 양면조립라인밸런싱에 관한 연구는 조사한 바에 의하면 현재까지 Bartholdi[1], 이태옥과 김여근[13]의 연구가 있다. Bartholdi는 양면조립라인 밸런싱문제를 위한 대화형 프로그램(interactive pro-

gram)의 설계와 사용에 관하여 연구하였다. 이 연구에서는 간단한 first fit rule 할당규칙을 적용하였다. 이태욱과 김여근의 연구는 작업관련성과 작업여유성을 고려한 작업그룹단위의 할당규칙을 제안하였다.

본 연구에서는 양면조립라인의 라인길이(작업장쌍 수) 최소화를 목적으로 하는 효율적인 알고리듬을 개발한다. 이 알고리듬은 분지한계법에 기초한 일종의 발견적 기법이다. 효율적인 해공간의 탐색을 위하여 할당규칙을 사용하는 새로운 열거나무 생성절차를 제안한다. 이 절차에서는 할당작업들의 순서를 국부적으로 조사하여 작업간 유휴시간을 줄일 수 있는 방법을 제시한다. 또한 적절한 분지전략과 한계전략, 그리고 분지완료규칙들을 다루며, 합리적인 시간내에 비교적 좋은 근사최적해를 구하기 위한 발견적 분지완료규칙들을 제안한다.

2. 제한적 열거법

일반적으로 분지한계법의 주요 요소는 분지전략, 탐색전략, 한계전략, 그리고 분지완료(fathoming)이다. 분지한계법에서 해를 탐색해 가는 방법은 분지전략과 탐색전략에 의해 결정된다. 이 절에서는 이들 전략에 의해 만들어지는 열거나무(enumeration tree)를 생성하는 방법에 관하여 다룬다. 그리고 이들 열거에서 분지를 완료하는 데 사용되는 한계전략과 분지완료규칙은 제3절에서 다룬다.

탐색전략에는 크게 깊이우선탐색(depth-first search)방법과 너비우선탐색(breadth-first search)방법이 있다. 깊이우선탐색방법은 노드에서 단지 하나의 노드만을 분지하여 하나의 해를 찾고, 이 해로부터 역추적(backtracking)하여 해를 탐색해 가는 방법이다. 너비우선탐색방법은 현 노드에서 가능한 모든 경우를 분지하면서 해를 탐색해 가는 방법이다. 깊이우선탐색방법은 하나의 가능해를 빨리 구할 수 있고, 이 해를 이용하여 역추적탐색을 함으로써 적은 메모리를 사용할 수 있는 알고리듬이다[7]. 본 연구에서도 깊이우선탐색방법을 사용한다.

앞에서 언급했듯이 양면조립라인에서는 각 작업장의 작업할당뿐만 아니라 작업장쌍내에 할당된 작업의 순서도 고려되어야 한다. 또한 방향제약이 없는 작업들은 할당가능한 방향이 모두 고려되어야 한다. 이를 고려한 모든 가능해의 열거는 많은 노력과 계산시간을 요구한다. 따라서 본 연구에서는 모든 가능해를 열거하지 않으면서 효율적으로 근사최적해를 구할 수 있는 제한적 열거법을 제안하고자 한다.

제한적 열거법은 할당규칙에 의하여 작업을 선택하고, 이 작업을 열거나무의 각 노드에 할당한다. 만약 선택된 작업이 유휴시간이 발생하면, 선택된 작업과 이미 할당된 일부 작업들의 작업순서를 국부적으로 조사한다. 이와 같은 방법에 의하여 어떤 노드에서 하나의 가능해가 완성되거나 분지완료하면 다른 가능해를 탐색하기 위하여 역추적을 시작한다. 이때 더 이상 역추적할 노드가 존재하지 않으면 가능해의 열거가 끝난다.

먼저 열거법에서 사용된 기호를 정의한다.

k = 노드의 깊이, k = 1, 2, ...

o_k = 깊이 k인 노드에 할당된 작업,

$F(k)$ = 깊이 k인 노드에 할당가능 작업의 집합.

여기에서 할당가능 작업이란 미할당작업으로 작업의 선후행제약, 사이클타임제약, 그리고 방향제약을 만족하는 작업을 나타낸다.

그림 3은 제시한 열거법에 의하여 그림 2의 선행공정도로부터 모든 가능해를 열거한 열거나무이다. 이때 사이클타임(CT)은 5로 두었다. 모든 작업이 열거나무의 노드에 할당되면 한 경로가 완성되고 이 경로는 가능해를 나타낸다. 그림 3의 노드 0에서 7까지의 경로가 가장 먼저 구해지는 초기해이다.

2.1 깊이우선탐색

조립라인밸런싱문제에서 깊이우선탐색방법을 사용할 때 흔히 작업번호에 기초하여 열거하고 있다[4,5,6,7,8]. 본 연구에서는 효율적으로 해를 탐색하기 위하여 가능한 좋은 해가 먼저 탐색될 수 있도록 작업 할당규칙을 제안한다. 여기서 좋은 해란 작업간 유휴시간을 줄여 소요 작업장 수가 적은 작업할당을 의미한다. 앞에서 언급했듯이 작업장에 할당될 작업과 함께 그들 작업의 작업순서가 고려되어야 한다. 본 연구에서는 할당작업의 모든 순서를 고려하는 데는 많은 노력이 요구되므로, 먼저 작업간 유휴시간이 가능한 발생하지 않도록 할당규칙을 사용하여 작업을 선택하여 할당한다. 선택된 작업을 할당하는 과정

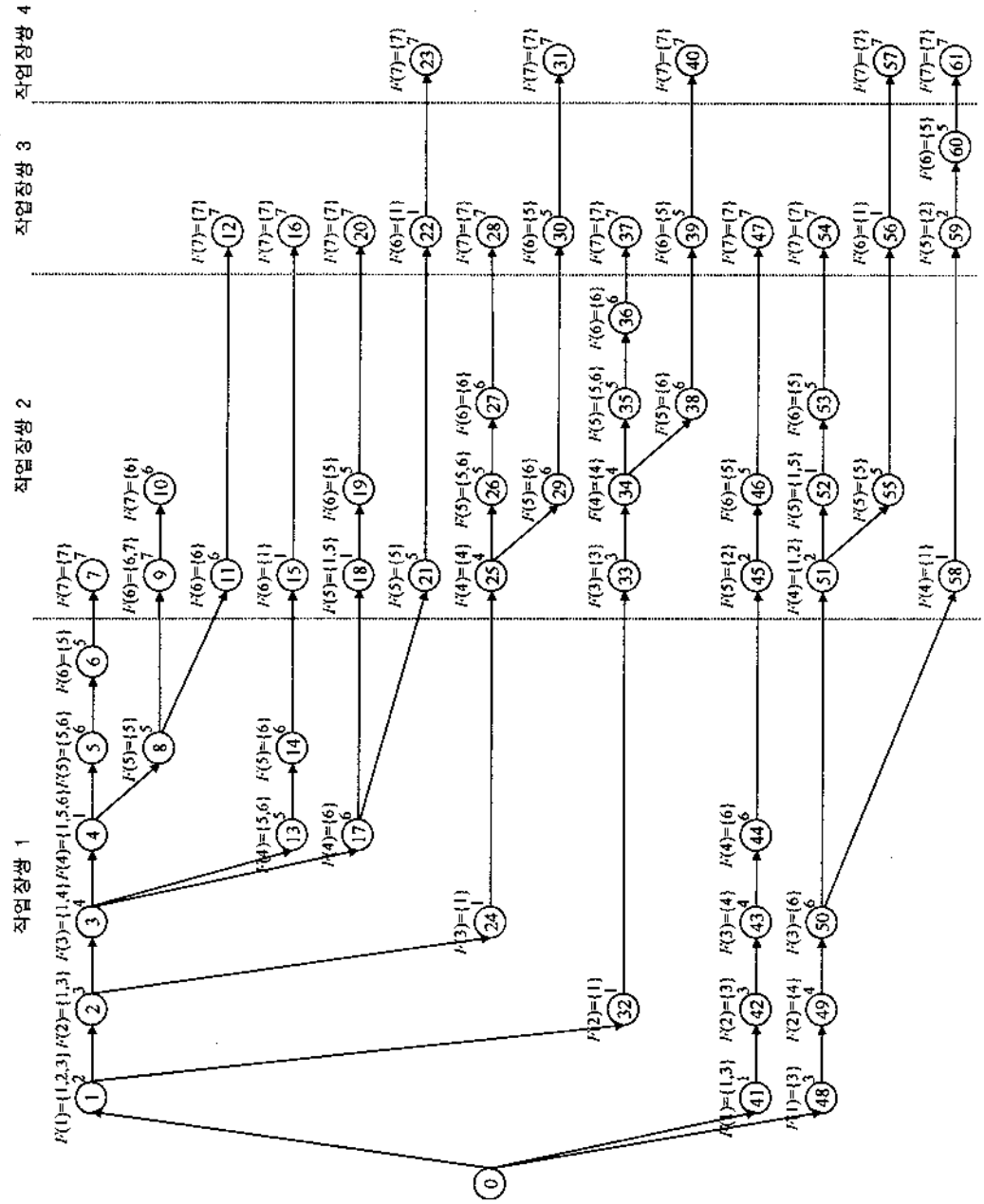


그림 3. 제한적 열거법에 의한 열거나무

에서 작업간 유휴시간이 발생하지 않으면 이미 할당된 작업순서를 고정하고 할당을 계속해 나간다. 작업간에 유휴시간이 발생하게 되면, 할당작업들의 작업순서를 국부적으로 조사하여 작업간 유휴시간이 최소화되는 할당순서를 찾는다. 이렇게 함으로써 각 작업장에 가능한 최대작업부하집합(maximal work load set)이 할당되도록 하고자 한다. 이러한 작업할당은 모든 작업순서를 고려하지 않으므로 최적해를 보장하지 못하나, 짧은 시간에 좋은 해를 탐색할 수 있다는 장점을 갖는다.

본 연구에서 사용하는 할당규칙은 다음과 같다.

우선순위 1: 시작가능시점이 가장 빠른 작업을 선택한다.

우선순위 2: 직후행작업의 작업방향이 다른 작업을 선택한다.

우선순위 3: 기존 할당규칙[10]으로 최대작업시간규칙, 최대총후행작업수규칙, 최대직후행작업수규칙, 그리고 최대순위위치가중치규칙을 차례로 사용하여 작업을 선택한다.

선택된 작업이 방향제약이 없으면서 양면에서 시작가능시점이 같으면, 이 작업을 할당할 작업장을 선택해야 한다. 이 경우, 작업간 유휴시간이 없으면 임의로 하고, 그렇지 않으면 작업간 유휴시간이 발생하지 않는 작업장을 선택한다.

할당규칙에 의하여 선택된 작업이 선행제약에 의하여 유휴시간이 발생하면, 유휴시간을 유발한 작업을 가능한 빨리 할당할 수 있는 작업순서를 찾는다. 이를 위하여 현 작업장쌍에 할당된 작업중에서 유휴시간이 발생한 작업의 시작시점에 직접적으로 영향을 주는 선행작업들의 사슬(chain)을 만든다. 이 사슬을 '선행사슬(precedence chain)'이라 부르기로 한다. 선행사슬은 유휴시간을 유발한 작업부터 차례로 직선행작업들을 연결한 것인데, 여기서 연속 할당된 두 작업이 직선후행관계에 있으면서 시작시점과 완료시점이 같은 작업들의 사슬이다. 작업간 유휴시간은 이 선행사슬에 속한 작업들의 이동에 의해서 줄일 가능성이 있다.

예로 그림 4(a)의 작업 6에서 유휴시간이 발생하였다. 이때 작업 6과 4는 직선후행관계에 있고 선행작업 4의 완료시점과 후행작업 6의 시작시점이 같다. 또한 작업 4와 3은 직선후행관계에 있고 작업 3의 완료시점과 작업 4의 시작시점이 같다. 그러나 작업 1과 2의 완료시점은 작업 3의 시작시점과 같으나 이들 작업은 직선후행관계에 있지 않으므로 사슬을 이루지 않는다. 따라서 현 작업장쌍에서 선행사슬은 그림 4(a)에서 보인 작업 {3,4}로 이루어 진다.

작업순서조정은 현 작업장쌍내에서 (역추적기준점이 있는 경우 역추적노드 이후에서) 선행사슬에 속한 작업과 교환가능한 작업을 구하고, 이를 이용하여 교환이동을 행한다. 먼저 선행사슬에 속한 작업들을 구하여 집합 PCS로 두자. 그리고 PCS에 있는 각 작업과 교환가능작업은 PCS에 속하지 않는 작업중 아래의 조건을 만족하는 작업으로 둔다. 이때 역추적기준점이 있는 경우, 교환가능작업은 역추적노드 이후에 할당된 작업들로 구성된다.

교환가능작업의 설명을 위하여, 현 작업장쌍내에 있는 작업 i의 직선행작업들을 집합 $P'(i)$, 직후행작업들을 집합 $S'(i)$로 두자. 그리고 작업 i의 시작시점을 t_i^s, 완료시점을 t_i^f로 두자.

교환가능작업: 작업 $i_p \in$PCS와 교환이 가능한 작업 $i_q \notin PCS$는 재 할당된 작업방향이 i_p와 같고, $t_{i_q}^s \geq \max_{h \in P'(i_p)} \{t_h^f\}$이고, $t_{i_p}^f \leq \min_{i \in S'(i_q)} \{t_i^s\}$을 만족하는 작업이다. 여기서 $P(i_p) = \phi$이면 $t_{i_q}^s \geq 0$이고, $S(i_q) = \phi$이면 $t_{i_p}^f \leq CT$이다. 그리고 이러한 조건을 만족하는 작업집합을 EWS(i_p)로 둔다.

교환이동은 PCS에 있는 작업중 시작시점이 가장 빠른 작업부터 차례로 행하여 진다. 작업 $i_p \in$PCS를 교환한다고 하자. 작업 i_p가 EWS(i_p)= ϕ이면 교환이동하지 못하고, 그렇지 않으면 교환가능 작업중 현 할당에서 시작시점이 가장 빠른 작업 $i_t \in$EWS(i_p)를 선택하여 두 작업을 교환이동 한다.

선행사슬을 이용한 교환이동절차는 아래와 같다.

〈교환이동절차〉

단계 1. 선행사슬에 속한 작업들을 구하여 집합 PCS로 둔다.

단계 2. 만일 PCS = ϕ이면 단계 5로 간다. 그렇지 않으면 PCS에 있는 작업중 현 할당에서 시작시점이 가장 빠른 작업 i_p를 선택하고, PCS ← PCS - $\{i_p\}$로 둔다. i_p와 교환가능한 작업을 구하여 EWS(i_p)로 둔다.

단계 3. 만일 EWS(i_p) = ϕ이면 단계 5로 간다. 그렇지 않으면 EWS(i_p)에 있는 작업중 현 할당에서 시작시점이 가장 빠른 작업을 선택하여 교환작업 i_r로 둔다.

단계 4. 작업 i_p와 i_r를 교환이동한다. 그리고 현 할당작업들의 완료시점을 조정하고 단계 2로 간다.

단계 5. 해가 개선되었으면(유휴시간이 감소하였으면) 현 작업장쌍에 할당된 노드들의 할당가능 작업집합을 조정하고 끝낸다. 그렇지 않으면 원상태로 복구하고 끝낸다.

예로 그림 4(a)는 그림 2의 예제를 사용하여 제안한 할당규칙에 의해 일부 할당된 작업의 간트챠트이다. 이때 사이클타임은 5로 두었다. 작업 6의 유휴시간 발생으로 교환이동절차를 행한다. 선행사슬에 속한 작업들을 구하여 PCS = {3,4}으로 둔다. 만약 PCS에 있는 작업중 먼저 작업 3을 교환이동 한다면 EWS(3) = {1}이 되고, PCS = {4}가 된다. 그림 4(b)는 작업 3과 1를 교환이동한 후의 간트챠트이다. 다음은 PCS = {4}이므로 EWS(4) = {1}이 된다. 이때 작업 3은 PCS에 속하지 않으나 교환가능작업 조건을 만족하지 않는다. 작업 4와 1이 교환이동되고, PCS = ϕ가 된다. 이 결과는 그림 4(c)의 간트챠트이다. PCS = ϕ이므로 교환이동은 끝난다. 작업순서 조정에 의해서 유휴시간이 감소하였으므로 현 할당순서에 따라 할당가능 작업집합을 조정하고 교환이동절차를 끝낸다. 그리고 할당규칙을 사용하여 작업할당을 계속한다. 그림 4(d)는 이와 같은 작업순서 조정에 의해 새로이 작업 5가 할당된 간트챠트이다.

이 방법에 의한 작업할당은 현재해(trial solution)의 각 노드에 할당된 작업번호뿐만 아니라 각 노드의 할당가능 작업집합의 저장이 요구된다. 열거과정에서 작업순서조정에 의해 작업순서가 변경되면, 각 노드의 할당가능 작업집합을 조정하여야 한다. 이는 다음절에서 설명하는 역추적을 용이하게 하기 위함이다. 따라서 약간의 저장용량이

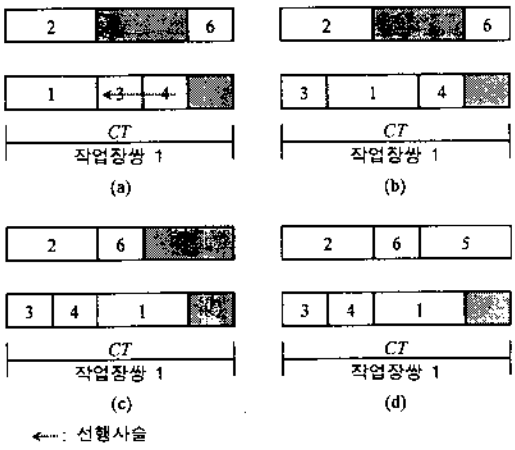


그림 4. 선행사슬과 작업순서의 국부탐색

필요하게 된다. 그리고 깊이우선탐색에서 이 할당규칙에 의한 초기해를 열거하는 방법은 다음과 같다.

깊이 k의 노드에 한 작업 o_i이 할당되었다고 하자. 그러면 깊이 $(k+1)$ 노드의 $F(k+1)$은 다음과 같이 구해진다. 먼저 (a) $F(k+1) \leftarrow F(k) - \{o_i\}$로 두고, (b) $F(k+1)$에서 할당가능조건을 만족하지 않는 작업들을 삭제한다. 그리고 (c) o_i의 직후행작업중 할당가능조건을 만족하는 작업들을 $F(k+1)$에 추가한다. $F(k+1) = \phi$이면 새로운 작업장쌍을 생성하여 미할당작업중 선행제약을 만족하는 모든 작업을 $F(k+1)$로 둔다. $F(k+1)$에서 할당규칙에 의해 할당작업을 선택한다. 이와 같은 방법으로 모든 작업이 할당될 때까지 열거하여 할당한다. 이렇게 하여 현재해가 만들어지면 마지막 열거된 노드부터 역추적하게 된다.

2.2 역추적

깊이우선탐색방법은 하나의 가능해를 완성하거나 분지완료하면 가장 최근에 생성된 노드에서 역추적하여 새로운 가능해를 탐색하게 된다. 앞에서 언급했듯이 본 연구에서는 현재해의 각 노드에 할당된 작업과 그 노드에서의 할당가능 작업집합을 저장하고 있다. 역추적방법을 설명하기 위하여 깊이 k인 노드에서 역추적한다고 하고, 이 노드를 '역추적기준점'이라고 하자. 그리고 그 노드에 할당된 작업을 o_i, 그 노드에 할당가능 작업집합을 $F(k)$로 두자. 여기서 $o_i \in F(k)$이다. 만약 $F(k) - \{o_i\} = \phi$이면 역추

적기준점이 깊이 $(k-1)$의 노드로 옮겨간다.

$F(k) - \{o_k\} \neq \phi$이라 하자. 현재해의 깊이 $(k-1)$에서 새로 분지되어 생성되는 노드를 '역추적노드'라 부르기로 하자. 깊이 k인 역추적노드의 작업할당은 먼저 $F(k) \leftarrow F(k) - \{o_k\}$로 두고, $F(k)$에 있는 작업중에서 할당규칙을 적용하여 선택한다. $F(k)$에서 o_k의 제거는 역추적기준점에 할당된 작업이 현 작업장쌍에 다시 할당되지 않게 하기 위함이다. 이와 같이 깊이 $(k-1)$에서 분지되는 역추적노드의 작업할당은 $F(k) = \phi$일때 까지 계속된다. 따라서 현재해의 깊이 $(k-1)$에서 역추적되는 분지의 수는 $F(k)$에 있는 원소의 수, $|F(k)|$만큼 된다.

역추적노드에 작업이 할당되면 깊이 $(k+1)$ 이후부터의 작업할당은 앞 절에서 언급한 작업할당규칙과 국부적 작업순서 탐색에 의하여 현재해를 탐색한다. 이때 역추적노드가 있는 작업장의 국부적 작업순서 탐색은 역추적노드 이후에 할당된 작업들의 순서에 국한한다. 그리고 새로운 작업장의 탐색에서 첫번째 노드의 $F(\bullet)$는 초기해의 생성에서와 같이 모든 할당가능 작업들로 구성된다. 더 이상 역추적할 노드가 존재하지 않으면 이 절차를 끝낸다.

2.3 열거절차

이 절에서는 앞에서 설명한 할당방법과 국부탐색에 의한 작업순서 재배열방법을 이용한 열거절차를 자세히 보인다. 편의상 왼쪽과 오른쪽 작업장을 각각 j와 j'로 둔다. S_j와 $S_{j'}$는 작업장 j와 j의 여유시간(CT - 마지막 할당작업의 완료시점), t_i는 작업 i의 작업시간, d_i는 작업 i의 유휴시간을 나타낸다.

단계 1. (초기화)

(a) 작업장 번호 $j := 1$, $j' := 2$, 노드의 깊이 $k := 1$로 둔다.

(b) 작업장 j와 j'의 여유시간을 각각 $S_j := CT$, $S_{j'} := CT$로 둔다.

(c) 선행작업이 없는 미할당작업들을 구하여 작업집합 $F(k)$로 둔다.

단계 2. (작업할당)

(a) $F(k)$에 있는 작업중 할당규칙에 의하여 작업 i를 선택한다.

(b) 작업 i를 해당 작업장에 할당하고 $o_k := i$로 둔다

단계 3. (여유시간 갱신)

(a) 작업간 유휴시간 d_i를 계산한다.

(b) 만일 작업 i가 작업장 j에 할당되면 $S_j := S_j - (t_i + d_i)$, 작업장 j'에 할당되면 $S_{j'} := S_{j'} - (t_i + d_i)$로 둔다.

(c) 만일 $d_i > 0$이면 2.1절의 교환이동절차를 수행한다.

(d) $k := k + 1$로 둔다.

단계 4. (할당가능 작업집합: 작업장쌍에서 첫번째 노드가 아닌 경우)

(a) $F(k) \leftarrow F(k-1) - \{o_k\}$로 두고, $F(k)$에서 사이클타임 제약을 만족하지 않는 작업을 삭제한다. o_k의 직후행작업중 선행제약, 사이클타임제약, 방향제약을 만족하는 작업을 $F(k)$에 추가한다.

(b) 만일 $F(k) = \phi$이면 단계 5로 간다. 그렇지 않으면 단계 2로 간다.

단계 5. (해의 완성검사와 새로운 작업장쌍 생성)

모든 작업이 할당되었으면 단계 7로 간다. 그렇지 않으면 $j := j + 2$, $j' := j' + 2$, $S_j := CT$, $S_{j'} := CT$로 둔다.

단계 6. (할당가능 작업집합: 작업장쌍에서 첫번째 노드인 경우)

미할당작업중에서 선행작업이 모두 할당된 작업을 $F(k)$로 두고 단계 2로 간다.

단계 7. (현재해의 완성)

현재해가 완성된다. 만일 완성된 현재해가 최선해보다 좋으면 현재해를 최선해로 갱신한다.

단계 8. (역추적)

(a) 만일 $S_j = CT$이고 $S_{j'} = CT$이면 $j := j - 2$, $j' := j' - 2$로 둔다.

(b) $k := k - 1$로 둔다.

(c) 만일 $k = 0$이면 모든 작업의 열거가 완성된다. 끝낸다.

(d) 작업 o_k를 해당 작업장에서 제거하고, $b := o_k$로 둔다.

(e) 만일 작업 b가 작업장 j에서 제거되면 $S_j := S_j + t_b + d_b$, 작업장 j'에서 제거되면 $S_{j'} := S_{j'} + t_b + d_b$로 갱신한다.

단계 9. (역추적노드 작업집합 갱신)

(a) $F(k) \leftarrow F(k) - \{b\}$로 둔다.

(b) 만일 $F(k) = \phi$이면 단계 8로 간다. 그렇지 않으면 단계 2로 간다.

3. 한계전략과 분지완료규칙

본 절에서는 해의 탐색공간을 가능한 줄이기 위하여, 양면조립라인의 특성들을 이용한 한계전략과 분지완료규칙들을 제시한다.

3.1 한계전략

한계전략은 문제에 따라 상한(upper bound: *UB*), 하한(lower bound: *LB*) 또는 혼합방법이 흔히 사용된다. 본 연구에서는 상한과 하한을 동시에 이용하는 전략을 사용한다.

현재까지 구한 가장 좋은 해를 현재의 최선해(incumbent solution)로 두자. 이 최선해를 상한으로 둔다. 현재해의 하한, *LB*은 각 작업장쌍의 할당이 완성되면, 할당되어 결정된 작업장쌍의 수, n_a와 아직 미할당된 작업들의 이론적 최소 작업장쌍의 수, n_t를 더하여 구한다. 이때 미할당된 이론적 최소 작업장쌍의 수, n_t는 다음과 같이 구한다.

$$n_t = \begin{bmatrix} [max(LT,RT)/CT]^+ & ET \le DT \\ [max(LT,RT)+(ET-DT)/2/CT]^+ & ET \ge DT \end{bmatrix} \quad (1)$$

여기서 $LT = \sum_{i \in A_L'} t_i$, $RT = \sum_{i \in A_R'} t_i$, $ET = \sum_{i \in A_E'} t_i$, $DT = |LT - RT|$이고, A_L', A_R', A_E'는 각각 미할당된 왼쪽 작업, 오른쪽 작업, 방향제약 없는 작업들의 집합이다. 모든 작업을 미할당작업으로 두면, 식 (1)로 부터 구한 n_t는 이론적인 최소 라인길이(작업장쌍 수), *LB**가 된다.

현재해의 탐색과정에서 작업장쌍이 완성되면 *LB*를 구하여 $LB \ge UB$이면 분지완료한다. 이 과정은 2.3절 열거절차 단계 5에서 새로운 작업장쌍을 생성하기 전에 적용된다. 그리고 완성된 현재해의 라인길이(작업장쌍 수)가 *LB**와 같으면 최적해이므로 모든 열거절차를 끝낸다. 이 과정은 2.3절 열거절차 단계 7의 해의 완성에서 적용된다.

3.2 분지완료규칙

분지완료규칙은 현재의 최선해보다 더 좋은 해를 유도할 수 없는 열거를 제거하는 규칙이다. 깊이우선탐색방법을 이용한 단면조립라인 밸런싱문제의 여러 분지완료규칙이 제안되었다[7,8]. 본 절에서는 양면조립라인의 특성을 고려한 새로운 분지완료규칙과 기존 단면조립라인 밸런싱문제에서 적용된 분지완료규칙들의 변형에 관하여 다룬다.

3.2.1 유휴시간 지배규칙

유휴시간 지배규칙은 작업간 유휴시간을 이용한 분지완료규칙이다. 깊이 *k*인 노드에 할당가능한 작업집합 $F(k)$를 고려하자. 작업 *i*와 *p*가 $F(k)$의 원소이고 동일 작업장에 할당가능할 때, $t_i^s \ge t_p^f$이면 작업 *i*의 할당은 작업 *p*의 할당보다 더 좋은 해를 유도하지 못한다. 여기에서 t_i^s는 작업 *i*가 할당될 때의 가장빠른 시작시점, t_p^f는 작업 *p*가 할당될 때의 가장빠른 완료시점이다. 이러한 경우는 그림 5에서 처럼 작업 *i*가 작업 *k*의 직후행작업일 때, 작업 *i*의 유휴시간, d_i가 작업 *p*의 작업시간, t_p보다 크거나 같은 경우이다.

작업 *i*가 작업 *p*보다 먼저 할당되는 경우는 할당규칙에 의하여 일어나지 않는다. 그러나 깊이 *k*에서 $F(k)$로부터 역추적노드를 생성할 때 작업 *i*가 역추적노드에 할당될 수 있다. 따라서 역추적노드를 생성할 때, 이와 같이 유휴시간 지배규칙에 의해 지배되는 작업은 역추적노드에 할당하지 않는다. 이 지배규칙은 열거절차의 단계 2에서 할당작업을 선택할 때 적용된다.

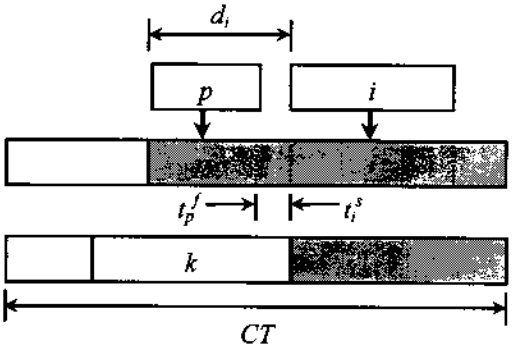


그림 5. 유휴시간 지배규칙

3.2.2 Jackson Rule-1 지배규칙

Jackson Rule-1 지배규칙은 최근 생성된 작업장쌍이 이미 생성된 임의의 작업장쌍에 대한 부분집합이면 그 작업장쌍 이후는 분지완료하는 규칙이다. 깊이우선탐색방법은 현재까지 생성한 모든 노드를 저장하고 있지 않으므

로 이 지배규칙은 다음과 같은 방법으로 확인한다. 제시한 열거법은 역추적기준점에 해당하는 작업들은 역추적노드가 있는 작업장쌍의 작업할당에서 제외된다. 따라서 제안한 열거법에서는 하나의 작업장쌍을 완성한 후 역추적기준점에 해당하는 작업에 의해 할당가능 조건을 만족하는 작업이 존재할 수 있다. 이 경우에 최근 완성된 작업장쌍은 과거에 생성된 임의의 작업장쌍에 대한 부분집합으로 구성되어 Jackson Rule-1 지배규칙이 적용된다.

예로 그림 3에서 작업장쌍 1을 완성하는 노드 6과 8을 비교해보자. 노드 6에서 완성되는 작업장쌍은 작업 {2,3,4,1,6,5}로 이루어지고, 노드 8에서 완성되는 작업장쌍은 작업 {2,3,4,1,5}로 이루어진다. 작업 6이 노드 8에서 완성되는 작업장쌍 1의 할당에서 제외되었다. 그러므로 노드 8은 분지완료한다. 이와 같이 그림 3에서 Jackson Rule-1 지배규칙이 성공적으로 적용되는 노드는 노드 8, 14, 17, 24, 32, 44, 50이다.

3.2.3 Jackson Rule-2 지배규칙

단면조립라인에서 Jackson Rule-2 지배규칙은 완성된 작업장에서 작업 h가 i로 교체될 때, $S_a(i) \subseteq S_a(h)$ 이고 $t_i \leq t_h$이면 작업 h는 작업 i를 지배한다. 여기에서 $S_a(i)$는 작업 i의 선행작업집합이다. 양면조립라인에서는 작업방향이 존재하므로 교체되는 작업이 동일 작업장에서 행해질 때, 이 지배규칙이 성립한다.

3.2.4 첫번째 작업장쌍 지배규칙

첫번째 작업장쌍 지배규칙은 선행작업이 없는 몇몇 작업들에 의해 발생한다[7]. 역추적탐색에서 첫번째 작업장쌍에 할당된 작업집합이나 그 부분집합이 첫번째 작업장쌍 이후의 작업장쌍에 다시 할당되면 분지완료하는 방법이다. 깊이우선탐색방법에서 이 지배규칙을 적용하기 위하여 첫번째 작업장쌍에 할당된 작업집합들을 저장하여야 한다. 그러나 이 지배규칙에 의한 분지완료는 그 효과가 매우 크므로 이에 대한 메모리는 약간 희생될 수 있을 것이다.

3.3 분지한계절차

본 절에서는 제한적 열거법에 분지완료규칙의 적용을 설명한다. 본 연구에서 제시한 알고리듬의 전체적인 절차는 아래와 같다.

1) 입력자료를 읽는다.
2) Jackson Rule-2 지배규칙의 가능성이 있는 모든 작업을 확인한다.
3) 3.1절에서 설명한 라인길이의 하한, LB^*를 구한다.
4) 아래의 규칙들과 함께 제한적 열거법에 의하여 가능해를 열거한다.
   (1) 단계 2에서 작업을 선택할 때 3.2.1절의 유휴시간 지배규칙이 적용된다. 여기에서 유휴시간 지배규칙은 역추적노드를 생성할 때 적용된다.
   (2) 단계 5에서 새로운 작업장쌍을 생성하기 전에 3.1절의 한계전략과 3.2.2절에서 3.2.4절까지 설명한 지배규칙들이 적용된다. 만약 이러한 지배규칙들이 성공적으로 적용되면 새로운 작업장쌍을 생성하지 않고 단계 8로 이동한다.
   (3) 단계 7의 해의 완성에서 3.1절에서 설명한 것처럼 새로 생성된 해의 라인길이와 하한, LB^*를 비교한다. 만약 새로 구한 해의 라인길이가 LB^*와 같다면 새로 구한 해가 최적해이므로 열거절차를 끝낸다.

4. 발견적 분지완료규칙

제안한 알고리듬의 성능을 분석하기 위하여 예비 실험한 결과, 계산소요시간은 흔히 이분법으로 아주 짧거나 아주 길었다. 일정시간이 경과한 후, 열거가 완성되지 않으면 계산시간의 단축을 위하여 아래와 같이 좀 더 강화된 분지완료규칙을 사용한다. 이때 현재까지의 최선해를 상한으로 사용한다.

4.1 가지수 제한규칙

가지수 제한규칙은 역추적노드의 최대분지수를 제한하는 방법이다. 할당가능 작업집합에 속한 작업수가 많으면 열거나무의 크기가 크게 된다. 제안한 열거법은 할당규칙을 사용하여 가능한 초기에 좋은 해를 찾도록 유도하고 있다. 역추적되어 늦게 할당되는 작업일수록 좋은 해를 유도하지 못할 가능성이 높다. 따라서 이 규칙은 해의 질

에 크게 손상을 주지 않을 수 있다. 이 지배규칙은 역추적노드에 할당작업을 선택하기 전에 열거절차의 단계 9에서 적용된다.

4.2 할당 여유시간을 이용하는 방법

작업장쌍의 할당이 완료되면 현재까지 소모된 여유시간(slack time)에 근거하여 현재해보다 더 좋은 해를 유도할 가능성이 낮은 경우 분지완료하는 방법이다. 현재의 상한에서 1을 뺀 라인길이, UB-1을 '개선해'의 라인길이로 두자. 그리고 탐색중인 해에서 작업할당이 이미 완성된 작업장쌍을 n_d, 작업장쌍 j의 여유시간을 SL_j로 두자. 개선해에서 허용되는 총 여유시간은 $TSL = \{2 \times CT \times (UB\text{-}1)\} - \sum_{i=1}^{m} t_i$이 된다. 개선해의 라인길이를 기준으로 작업장쌍당 평균여유시간은 $ASL = TSL / (UB - 1)$이고, 남아있는 작업장쌍당 평균여유시간은 $RASL = (TSL - \sum_{j=1}^{n_d} SL_j) /(UB - 1 - n_d)$이 된다. 이때 남아 있는 평균여유율은 $\gamma = RASL/ASL$이 된다.

γ가 낮을수록 좋은 해로의 유도가능성은 낮다고 본다. 좋은 해로의 유도가능성은 γ와 매개변수로 주어지는 α ($0 < \alpha < 1$)를 비교하여 판단할 수 있다. 즉 $\gamma < \alpha$이면 분지완료하는 방법을 사용할 수 있다. 그러나 γ는 n_d가 커감에 따라 낮은 값을 갖는 경향이 있다. 이를 고려하여 $\beta = \alpha \{(UB - n_d)/(UB\text{-}1)\}$을 도입하고, $\gamma < \beta$이면 분지완료하는 정책을 사용한다. 이 지배규칙은 열거절차의 단계 5에서 새로운 작업장쌍을 생성하기 전에 적용된다.

5. 실험 및 비교분석

본 연구에서 제시한 알고리듬의 성능을 비교분석하기 위하여 기존의 할당규칙과 비교하였다. 할당규칙으로는 할당가능작업중에서 최대작업시간을 갖는 작업을 선택하는 최대작업시간(MAX-DUR)규칙, 후행작업의 수가 가장 많은 작업을 할당하는 최대총후행작업수(MAX-TFOL)규칙, 직후행작업수가 가장 많은 작업을 할당하는 최대직후행작업수(MAX-IFOL)규칙, 자신과 모든 후행작업의 작업시간의 합이 가장 큰 것 부터 차례로 할당하는 최대순위위치가중치(MAX-RPW)규칙과 비교하였다. 이들 할당규칙은 Talbot *et al.*[10]을 참조할 수 있다. 이들은 모두 단면조립라인 밸런싱문제를 위해 개발한 것이므로, 이들 할당규칙을 양면조립라인 밸런싱문제에 적용 가능하도록 수정하였다. 또한 양면조립라인 밸런싱문제에 적용한 Bartholdi[1]의 first fit rule(FFR)과 이태욱과 김여근[13]의 작업그룹단위의 할당규칙(GR)과도 비교하였다. 양면조립라인에서 사이클타임이 주어진 라인길이 최소화문제에 대하여 해의 효율과 계산시간측면에서 기법의 성능을 비교하였다. 사용된 모든 할당규칙과 제안한 알고리듬은 C++ 언어를 사용하여 작성하였고, Pentium CPU 166MHz를 가진 IBM-PC를 사용 하였다.

실험에 사용된 문제는 A자동차회사의 트럭조립라인[12]에서 발췌한 65개의 작업을 갖는 문제(A65), 100개의 작업을 갖는 문제(A100), 205개의 작업을 갖는 문제(A205) 그리고 148개의 작업을 갖는 Bartholdi[1]의 문제를 사용하였다. Bartholdi의 문제는 작업장쌍의 수가 많은 경우를 실험하기 위하여 작업 79의 작업시간은 281에서 111로, 작업 108의 작업시간은 383에서 43으로 조정하였다.

5.1 기존 할당규칙의 수정

비교실험을 위하여 기존의 할당규칙을 아래와 같이 수정하여 양면조립라인 밸런싱문제에 적용하였다.

단계 1. 작업장 번호 $j := 1$, $j' := 2$로 둔다.

단계 2. 작업장 j와 j'중 적어도 어느 한 작업장에 할당가능한 작업들의 집합 H를 생성한다. 만일 $H = \phi$이면 단계 5로 간다.

단계 3. H에 있는 작업중에서 기존의 특정 할당규칙에 의하여 작업을 선택한다.

단계 4. 선택된 작업이 작업방향을 갖으면 해당작업장에 할당하고, 작업방향 제약이 없고 양쪽 작업장 모두에 할당가능하면, (1) 시작가능시점이 빠른 작업장이 할당작업장이 되고, 두 작업장에서 시작가능시점이 같으면 (2) 미할당작업들의 모든 왼쪽 작업의 작업시간합과 모든 오른쪽 작업의 작업시간합을 비교하여 적은 값을 갖는 작업장을 할당작업장으로 한다. 그리고 단계 2로 간다.

단계 5. 모든 작업이 할당되었으면 끝낸다. 그렇지 않으면 $j := j+2$, $j' := j'+2$로 두고 단계 2로 간다.

단계 2에서 할당가능한 작업집합 H는 작업의 선후행 제약, 사이클타임제약 그리고 방향제약을 만족하는 작업으로 이루어진다. 단계 3에서 선택된 작업이 복수개인 경우 임의로 선택하였다. 단계 4에서 방향제약이 없으면서 작업장 여유시간 제약에 의해 한쪽 작업장에만 할당되는 작업이 있을 수 있다. 이러한 작업은 현 할당에서 작업방향을 갖는 작업이 된다.

5.2 비교분석

라인길이(작업장쌍수) 최소화문제는 사이클타임을 제약으로 두고, 각 작업장의 완료시점이 사이클타임을 초과하지 않게 작업을 할당하는 문제이다. 기존의 할당규칙과

표 1. 해의 효율 비교

실험 문제	순서 강도	사이클 타임	발견적 기법	제안한 알고리듬	개선율(%)	계산시간(초)
A65	0.48	272	10+(146,131)[4]	9+(237,259)	5.5	0.01
		299	9+(89,139)[4]	8+(292,286)	5.2	4.90
		326	8+(275,182)[6]	7+(321,326)	9.5	26.77
		354	8+(65,0)[6]	7+(138,340)	2.7	0.01
		381	7+(0,95)[6]	6+(351,374)	3.7	11.53
		408	6+(342,272)[2]	6+(114,288)	1.9	0.01
		435	6+(171,106)[6]	5+(425,414)	6.5	0.86
A100	0.50	452	11+(311,423)[6]	10+(450,327)	7.9	0.04
		497	10+(223,389)[2]	9+(453,457)	8.0	11.97
		542	9+(161,331)[4]	8+(526,489)	6.7	1.15
		588	8+(253,548)[6]	7+(584,569)	10.5	15.44●
		633	8+(304,416)[5]	7+(484,619)	7.8	0.02
		678	8+(338,450)[6]	6+(667,645)	19.4	52.00●
		723	7+(0,416)[6]	6+(720,696)	7.7	0.12
		768	6+(605,750)[5]	6+(304,657)	1.7	0.01
B148	0.26	170	15+(113,108)[4]	15+(167,150)	-2.0	0.29
		187	14+(66,76)[4]	13+(184,184)	2.9	2.11
		204	13+(0,130)[6]	12+(196,172)	5.0	1.19
		221	11+(216,211)[2]	11+(221,220)	-0.2	2.39
		238	11+(18,155)[6]	10+(226,236)	5.7	4.73
		255	10+(223,0)[6]	10+(131,191)	1.2	0.02
		272	9+(263,241)[4]	9+(247,197)	0.6	0.02
		289	9+(111,47)[6]	8+(270,279)	4.5	12.02
		306	8+(257,221)[6]	8+(210,159)	1.7	0.02
		323	8+(307,0)[6]	7+(322,323)	10.6	3.29
A205	0.81	944	13+(823,896)[6]	12+(913,896)	7.0	28.69●
		1038	12+(752,858)[6]	11+(998,793)	6.7	6.19●
		1133	11+(0,751)[6]	10+(1112,1103)	5.8	1.21●
		1227	10+(462,435)[6]	9+(1134,1145)	4.3	0.38
		1322	10+(90,726)[5]	9+(575,938)*	8.0	1.21●
		1416	8+(1129,1309)[6]	8+(1290,1325)	-0.1	0.02
		1510	9+(244,332)[5]	7+(1481,1472)	13.4	1289.07●
		1605	8+(423,561)[5]	7+(1044,1407)	5.7	0.03
		1699	7+(1377,1521)[6]	7+(712,883)*	4.8	1.71●
		1794	7+(1467,1611)[6]	6+(1713,1738)	11.8	0.02
		1888	7+(0,303)[5]	6+(850,1516)	5.0	0.02
		1982	7+(90,85)[5]	6+(850,1213)*	6.2	1.70●

[1]: MAX-DUR, [2]: MAX-TFOL, [3]: MAX-IFOL, [4]: MAX-RPW, [5]: FFR, [6]: GR

*: 이론적인 최소 라인길이를 구하지 못한 해

●: 발견적 분지완료규칙을 적용하여 구한 해

제안한 알고리듬을 비교한 실험결과가 표 1에 제시되었다.

표 1은 각 문제마다 여러 사이클타임에 대하여 해의 효율과 계산시간 측면에서 비교한 실험결과이다. 각 문제의 사이클타임은 최대작업시간에 상수를 곱하여 결정하였다. 즉, A65문제에서는 최대작업시간에 1.0, 1.1, ..., 1.5, 1.6, A100문제에서는 1.0, 1.1, ..., 1.6, 1.7, B148문제에서는 1.0, 1.1, ..., 1.8, 1.9, 그리고 A205문제에서는 1.0, 1.1, ..., 2.0, 2.1의 값을 곱하였다. 제안한 알고리듬은 가능해의 열거과정에서 CPU 시간이 30초를 초과하면 발견적 분지완료규칙을 적용하였다. 이때 가지수 제한규칙에서 분지되는 노드수는 8, 할당여유시간을 이용하는 방법에서 α=0.5로 두었다.

표 1에서 2열은 선행공정도의 순서강도(order strength), 3열은 주어진 사이클타임이다. 4열은 기존의 할당규칙에 의하여 구한 해중 가장 좋은 해의 라인길이, 5열은 본 연구에서 제안한 알고리듬에 의해 구한 라인길이이다. 이때 라인길이는 $(n_d\text{-}1)+(L^f_{n_d}, R^f_{n_d})$로 나타내었다. 여기서 n_d는 기존의 할당규칙 또는 제안한 알고리듬으로 구한 해의 마지막 작업장쌍 번호이고, $L^f_{n_d}$와 $R^f_{n_d}$는 각각 작업장쌍 n_d에서 왼쪽 작업장의 완료시점과 오른쪽 작업장의 완료시점이다. 6열은 라인길이에 대한 개선율(improved rate) = $\frac{HN - PN}{HN} \times 100$이다. 여기서 HN과 PN은 각각 4열의 기존 할당규칙의 결과와 5열의 제안한 알고리듬의 결과를 나타낸다. 이때 $HN = (n_d - 1)+\frac{\max(L^f_{n_d}, R^f_{n_d})}{CT}$과 같이 계산되었다. PN은 HN을 구하는 방법과 같은 방법으로 계산되었다. 그리고 7열은 제안한 알고리듬의 계산소요시간으로 CPU시간이다.

실험결과, 해의 효율측면에서는 대부분의 문제에서 제안한 알고리듬이 기존의 할당규칙보다 우수함을 보여주고 있다. 또한 제안한 알고리듬은 거의 모든 문제에서, 특히 발견적 분지완료규칙이 적용된 문제를 포함하여, 여러 사이클타임에 대하여 대부분 이론적인 최소 작업장쌍 수를 구하였다. 개선율에서 발견적기법보다 더 좋지 않게 나온 세 경우는 모두 이론적 최소 작업장쌍수를 구한 경우로써 제안한 알고리듬이 이론적 최소 작업장쌍수를 구하면 종료되기 때문인 것으로 본다.

계산시간(CPU 시간)측면에서는 모든 기존의 할당규칙들은 모든 실험대상문제에서 1초 이하의 계산시간이 소요되었다. 제안한 알고리듬은 기존의 할당규칙보다 긴 CPU시간이 소요되었다. A65문제와 B148문제는 비교적 짧은 시간에 최적해를 모두 찾았다. 그러나 A100문제와 A205문제에서는 특정 사이클타임에 대하여 일정시간내에 해를 구하지 못하였다. A205문제에서 사이클타임이 1510인 경우, 발견적 분지완료규칙을 적용하였어도 많은 계산시간이 소요되었다. 발견적 분지완료규칙은 해의 탐색방향을 변경하여 때로는 이 규칙을 사용하지 않는 경우보다 더 많은 계산시간이 소요될 수 있다.

6. 결론

본 연구에서는 양면조립라인 밸런싱문제를 다루었다. 이 문제의 해결을 위하여 분지한계법에 기초한 효율적인 알고리듬을 개발하였다. 분지한계법에서는 할당규칙을 사용한 새로운 열거나무 생성절차를 제시하였다. 그리고 가능해의 열거과정에서 작업간 유휴시간이 발생하는 경우에 한하여 국부적으로 작업순서를 탐색하는 방법을 제시하였다. 또한 양면조립라인 밸런싱문제에 적절한 한계전략과 분지완료규칙을 제안하였다.

제안된 열거법은 모든 가능 작업순서를 고려하지 않는 제한된 열거법을 사용하고 있으나 할당규칙의 사용으로 비교적 좋은 해를 초기에 구할 수 있고, 국부적으로 작업순서를 고려함으로써 합리적인 시간내에 효율적으로 좋은 해를 탐색할 수 있다는 장점을 갖는다. 소요저장용량은 할당규칙에 의한 작업할당을 위하여, FABLE[7]에서보다 추가적으로 각 노드의 할당가능 작업집합을 보관해 두어야 한다. 그러나 각 깊이에서 가장 최근에 열거된 노드의 할당가능 작업집합만을 저장하기 때문에 많은 기억용량을 필요로 하지는 않는다.

제안한 알고리듬의 성능평가를 위하여 해의 효율과 계산시간면에서 실험하였다. 실험결과, 해의 효율측면에서 제안한 알고리듬은 기존의 할당규칙들보다 대부분 우수하였으며, 대부분의 문제에서 적은 시간내에 최적해를 찾았다. 또한 제안한 발견적 분지완료규칙 역시 해의 성능과 계산시간면에서 대부분 좋은 결과를 보였다.

참고문헌

[1] Bartholdi, J. J., "Balancing two-sided assembly lines: a case study", *International Journal of Production Research*, Vol. 31, No. 10, pp. 2447-2461, 1993.

[2] Baybars, I., "A survey of exact algorithms for the simple assembly line balancing problem", *Management Science*, Vol. 32, No. 8, pp. 909-932, 1986.

[3] Ghosh, S. and Gagnon, R. J., "A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems", *International Journal of Production Research*, Vol. 27, No. 4, pp. 637-670, 1989.

[4] Hackman, S. T., Magazine, M. J. and Wee, T. S., "Fast, effective algorithms for simple assembly line balancing problems", *Operations Research*, Vol. 37, No. 6, pp. 916-924, 1989.

[5] Hoffmann, T. R., "EUREKA: A hybrid system for assembly line balancing", *Management Science*, Vol. 38, No. 1, pp. 39-47, 1992.

[6] Johnson, R. V., "A branch and bound algorithm for assembly line balancing problems with formulation irregularities", *Management Science*, Vol. 29, No. 11, pp. 1309-1324, 1983.

[7] Johnson, R. V., "Optimally balancing large assembly lines with FABLE", *Management Science*, Vol. 34, No. 2, pp. 240-253, 1988.

[8] Klein, R. and Scholl, A., "Maximizing the production rate in simple assembly line balancing - a branch and bound procedure", *European Journal of Operational Research*, Vol. 91, pp. 367-385, 1996.

[9] Salveson, M. E., "The assembly line balancing problem", *Journal of Industrial Engineering*, Vol. 6, pp. 18-25, 1955.

[10] Talbot, F. B., Patterson, J. H. and Gehrlein, W. V., "A comparative evaluation of heuristic line balancing techniques", *Management Science*, Vol. 32, No. 4, pp. 430-454, 1986.

[11] Van Assche, F. V. and Herroelen, W. S., "An optimal procedure for the single-model deterministic assembly line balancing problem", *European Journal of Operational Research*, Vol. 3, pp. 142-149, 1978.

[12] 김여근 외 5인, "대형트럭라인의 생산성향상에 관한 연구", *아시아 자동차 공업(주) 보고서*, 1990.

[13] 이태욱, 김여근, "양면조립라인 밸런싱을 위한 할당규칙", *대한산업공학회*, 10권, 2호, pp. 29-40, 1997.

98년 4월 최초 접수, 98년 8월 최종 수정