Journal of the Society of Korea Industrial and Systems Engineering
Vol, 31, No. 2, pp.80－93, June 2008.

부품외주를 고려한 조립형 Flowshop 일정계획 해법 개선

윤상흠* · 전재호**†

*영남대학교 경영학부
**충주대학교 행정학부 행정정보시스템학전공

An Improvement of Algorithms for Assembly-type Flowshop Scheduling Problem with Outsourcing

Sang Hum Yoon* · Jaeho Juhn**†

*School of Management, Yeungnam University
**Division of Public Management Information System, Chungju National University

This paper improves algorithms for an assembly-type flowshop scheduling problem in which each job is to assemble two types of components and makespan is the objective measure. For the assembly, one type of the components is outsourced with job-dependent lead time but the other type is fabricated in-house. When both components for a job are prepared, the assembly operation for the job can be started. This problem had been proved to be NP-Complete, so branch-and-bound (B&B) and heuristic algorithms have already been developed. In this paper, we suggest other dominance rules, lowerbound and heutistic algorithms. Also, we develop a new B&B algorithm using these improved bound and dominance rules. The suggested heuristics and B&B algorithm are compared with existing algorithms on randomly-generated test problems.

Keywords : Assembly-Type Flowshop, Scheduling, Heuristic, Branch-and-Bound

1. 서 론

일반적으로 조립라인은 작업의 흐름에 따라서 두 가지로 구분 가능하다. 첫째는 흐름생산라인(flowshop)으로, 반제품이 흐름라인을 따라 이동해 가면서 직렬라인상의 각 설비를 통해 해당 부품이 조립됨으로써, 제품이 점차 완성되는 형태이다. 두 번째는 최종조립에 필요한 각 부품이 서로 다른 설비와 경로를 통해 독립적으로 생산되고, 완성된 부품들은 조립단계로 이동한 후, 모든 필요부품들이 준비된 시점에서 최종 조립공정을 통해 제품으로 완성되는 형태이다. 이러한 생산체제에서는 조립전단계(pre-assembly stage)에 위치한 각 부품 생산설비와 조립단계(assembly stage)간에 물리적으로 2단계 직렬구조를 형성하는 반면, 조립전단계의 부품생산설비들 간에는 독립적인 병렬구조를 형성하는데, 이를 일반적으로 AFS(assembly-type flowshop)라고 부르고 있다[10, 12].

AFS 일정계획 문제는 Lee et al.[9]과 Potts et al.[11]에 의해 제시되었으며 이후 다양한 관련연구가 이루어져왔다. Lee et al.은 두 개의 부품(차체와 차대)을 자체적으로 생산하여 소방차로 조립하는 문제에서 최대작업완료시간을 최소화하는 문제를 연구하였다. 이후 Sun et al.[12]은 Lee et al.의 문제를 보다 효과적으로 해결할 수 있는

논문접수일 : 2008년 02월 26일 논문수정일 : 2008년 04월 15일 게재확정일 2008년 04월 23일
† 교신저자 jhjuhn@cjnu.ac.kr
※ 이 논문은 2006년도 충주대학교 교내 학술연구비의 지원을 받아 수행한 연구임.

다양한 발견적 해법을 제시하였고, Lin et al.[10]은 Lee et al.의 문제에서 일괄조립기계가 존재하는 경우를 연구하였다. Potts et al.[11]은 Lee et al.의 문제를 보다 일반화하여 다수의 부품을 내부에서 생산한 후 조립하는 문제를 다루었다. Potts et al.의 일반화된 문제에 대해 Hariri and Potts[4]는 분지한계(branch-and-bound) 알고리즘을 제시하였고, Koulamas and Kyparisis[7]는 동 문제를 3단계 문제로 확장한 연구를 수행하였다. 또 Kovalyov et al.[8]은 동 문제에서 배칭의사결정을 추가한 문제를 연구하였다. Tozkapan et al.[13]은 총가중흐름시간을 최소화하는 분지한계 알고리즘을 제시한 바 있다. 이상의 연구들에서는 조립전단계에서 조립에 소요되는 모든 부품을 자체생산하는 상황을 가정하고 있다. 그러나, 현실에서 조립공정에 필요한 부품의 일부 혹은 전부를 외주로 조달하는 것이 보편적이므로[3] Juhn et al.[6]과 전재호[1]는 제 1단계 부품중 하나는 자체생산하고, 또 다른 하나의 부품은 외주로 조달한 후, 제 2단계에서 조립하는 문제를 연구하였다.

일반적으로 부품 외주를 고려하는 AFS는 다시 두 가지 경우 즉, 조립업체가 원하는 시점에 외주부품 조달이 가능한 경우와 그렇지 못한 경우로 대별될 수 있다. 전자의 경우 외주부품 조달시점이 문제해결에 제약으로 작용하지 않으므로, Juhn et al.[6]과 전재호[1]에서 대상으로 한 AFS는 결과적으로 기존의 2-machine flowshop 문제와 동일하게 된다. 그러나, 외주부품 조달시점이 외주부품 조달업체의 상황에 의해 결정되어 조립업체의 입장에서 이를 제약으로 수용해야 하는 경우에는 외주부품 조달시점과 내부생산 및 조립시간간의 최적 조화(coordination) 여부가 중요하게 된다. 현실에서 주문설계생산(ETO; Engineer-to-Order)에 속하는 제품생산의 경우 이러한 현상이 빈번히 발생한다[2]. Juhn et al.[6]에서 외주조달시간이 제약으로 작동하는 AFS 문제가 NP-Complete 문제임이 증명되었으므로, 전재호[1]는 발견적 해법과 분지한계알고리듬을 제안하고 이들의 성능평가를 수행한 바 있다.

본 연구는 외주조달이 있는 AFS 일정계획 문제에 대한 전재호[1]의 연구결과에 새로운 발견적 해법, 하한값 그리고 우월성질을 추가하고, 이들을 활용한 분지한계 해법을 제시함으로써 해법들의 성능을 전반적으로 향상시키고, 여러 가지 문제조합에 대해 계산실험을 수행하여 이를 확인하였다.

본 연구의 현실 적용 가능성은 다음과 같은 두 가지 측면에서 찾을 수 있다. 일차적으로, 현실의 전체공정 중에서 두 개의 부품을 조립하여 하나의 중간부품(반제품; subassembly)을 생산하는 특정 공정만을 집중적으로 고려하는 경우를 생각할 수 있다. 이 경우 본 문제에서 언급한 "작업(혹은 제품)"의 완결은 하나의 중간부품을 생산하는 것을 의미한다. 일반적으로, 이러한 사례는 전체공정에서 애로(bottleneck)공정만을 집중적으로 해결하려고 하는 경우에 발생한다. 잘 알려진 연구주제인 2-machine flowshop 일정계획 문제가 일반적인 m-machine flowshop (전체 공정)에서 애로공정이 되는 특정 2-machine flowshop 부분만을 집중적으로 고려한 경우라고 하겠다. 두 번째 적용가능성은 이미 세부부품들의 조립을 통해 준비되어진, 두 가지 유형의 공통 중간부품(반제품)별로 고객주문에 적합화를 위한 추가적인 가공처리(혹은 추가부품 조립)를 한 후 이 반제품들을 조립하여 최종제품을 생산하는 상황에서 찾을 수 있다. 이러한 상황에 대한 사례로는 전술한 Lee et al.[9]의 소방차 조립생산을 들 수 있다. Lee et al.의 연구에서 "부품(component)"으로 언급된 차체와 차대가 실제로는 다수의 세부부품들로 구성되어진 반제품(subassembly)에 해당된다. Lee et al.은 이 두 개의 반제품(차체와 차대) 생산시간 속에 각 반제품에 소요되는 다양한 부품의 생산/조달 및 조립시간을 포함시킴으로써, 다양한 부품으로 이루어진 소방차 생산문제를 두 개의 반제품을 생산/조립하는 문제로 모형화 하였다. 본 연구문제는 Lee et al.의 연구문제에서 두 부품(혹은 반제품) 중 하나를 외주로 조달하면서 외주조달 시간제약이 있는 경우를 다루고 있으므로, Lee et al.의 연구와 유사한 적용가능성을 갖는다.

본 연구의 구성은 다음과 같다. 제 2장에서는 연구대상 문제를 명확히 정의하고, 두개의 문제성질(solution property)들을 추가로 제시하였다. 제 3장에서는 한 개의 새로운 하한값(lower bound)을 추가하였고, 제 4장에서는 분지한계 알고리즘 및 세 가지의 수정된 발견적 해법을 제시하였다. 제 5장에서는 본 연구에서 제시한 발견적 해법들과 분지한계 알고리즘의 성능을 전재호[1]의 알고리즘들과 비교평가하기 위해 수치실험을 수행하고, 그 결과로부터 성능향상을 확인하였으며, 제 6장에서는 결론 및 추후 연구방향에 대한 논의를 하였다.

2. 문제정의와 분석

문제 자체에 대한 정의는 전재호[1]에 나타나 있으나, 사용된 기호의 이해도 제고와 논의의 완결성을 위해 간단히 재정리한다. 처리할 n개의 작업(제품)이 있다. 각 작업은 모두 2단계의 처리를 통해 완결된다. 제 1단계에서 각 작업에 소요되는 두 개의 부품중 하나는 내부기계 M_p를 사용해 자체적으로 생산하고, 또 다른 하나의 부품은 외주조달로 각각 준비한다. 이때, 외주조달 시점은 외

주업체의 사정에 따라 조립업체의 일정계획자에게 제약조건으로 통보된다. 1단계에서 두 개의 부품이 모두 준비되면 제 2단계에서는 기계 M_q를 사용해 두 부품들을 조립하여 하나의 작업을 완성한다. 각 작업의 완료시간은 조립기계 M_q에서의 조립완결 시점으로 측정된다. 본 문제에서는 각 작업의 완료시간 중에서 최대인 완료시간을 최소화하는 것을 목적으로 한다. 본 문제에서 사용된 가정은 전재호[1]와 동일하다. 문제의 표기와 분석에 사용되는 기호는 다음과 같다.

- p_i : 작업 i에 소요되는 내부생산부품의 생산시간.
- A_i : 작업 i에 소요되는 외주부품의 도착시간.
- q_i : 작업 i에 소요되는 두 부품의 조립시간.
- $\pi=(\pi(1), \cdots, \pi(n))$: 임의의 일정계획해, 여기서 $\pi(i)$는 일정계획해 π의 i번째 작업을 의미.
- π^* : 최적일정계획해
- $C^{\pi}_{\max}$: 임의의 일정계획해 π의 최대작업완료시간.

본 문제의 최적해는 M_p와 M_q에서 작업순서가 동일한 순열일정계획해(permutation schedule)와, 작업이 처리대기중인 경우 해당 기계가 유휴상태로 있을 수 없는 비지연일정계획해(nondelay schedule)들 중에서 존재함이 증명되어 있다[1]. 따라서, 위에서 제시한 기호를 사용하여 본 문제의 목적식을 표기하면 다음과 같다.

$$C^{\pi}_{\max}=\max_{u\in\{1,\cdots,n\}}\left[\max\left\{\sum_{i=1}^{u}p_{\pi(i)},A_{\pi(u)}\right\}+\sum_{i=u}^{n}q_{\pi(i)}\right] \quad (1)$$

본 문제의 난이도가 NP-Complete임을 서론에서 언급한 바 있다. 이는 본 문제에서 작업수 n이 커지면 최적해를 구하는 어떠한 해법도 복잡도가 지수함수적으로 증가해서, 제한된 시간 안에 최적해를 구하는 것이 매우 어렵다는 것을 의미한다. 일반적으로, NP-Complete문제의 경우 특정조건이 성립하는 특수상황에서 쉽게 문제를 해결할 수 있음(우월성질; dominance property)을 규명함으로써, 해당 조건을 성립하는 문제의 해결은 물론 일반적인 문제의 최적해를 구하는 분지한계 알고리즘의 성능향상에 활용하게 된다. 다음의 (정리 1)과 (정리 3)은 분지한계 알고리즘에서 특정 분지노드를 제거함으로써, 탐색공간을 효과적으로 줄여주는 역할을 수행한다.

본 연구문제는 Johnson[5]이 제시한 "2-기계 흐름라인" 문제의 일반형으로 볼 수 있다. "2-기계 흐름라인"의 "최대작업완료시간 최소화"문제는 이른 바 "Johnson의 법칙(첫 번째 기계(기계 1)의 처리시간이 두 번째 기계(기계 2)의 처리시간보다 짧은 작업은 기계 1 처리시간이 짧을수록 먼저 처리하고, 반대로 기계 1 처리시간이 기계 2 처리시간보다 긴 작업들은 기계 2의 처리시간이 짧을 수록 뒤에 처리)"에 따라 최적일정계획을 도출할 수 있다[5]. 아래의(정리 1)은 "Johnson의 법칙"을 본 문제에 적합화한 것으로 임의의 두 작업이 아래의 세가지 조건(첫 번째 조건과 세 번째 조건은 Johnson의 법칙과 동일하고, 두 번째 조건은 외주조달시점이 빠를 수록 앞에 처리하는 것이 동일하거나 우월함)을 동시에 만족한다면 두 작업 처리의 선후관계를 확정적으로 알 수 있게 된다.

정리 1 : 임의의 두 작업 j와 k에 대해 다음 조건들이 만족한다면 작업 j를 k보다 선행해서 처리하는 경우가 그렇지 않은 경우보다 작거나 같은 최대작업완료시간을 갖는다.
(조건 1) $p_j \le p_k$,
(조건 2) $A_j \le A_k$,
(조건 3) $q_j \ge q_k$.

증명 : 식 (1)을 다음과 같이 나타낼 수 있다. 즉,

$$\begin{aligned}C^{\pi}_{\max}&=\max_{u\in\{1,\cdots,n\}}\left[\max\left\{\sum_{i=1}^{u}p_{\pi(i)},A_{\pi(u)}\right\}+\sum_{i=u}^{n}q_{\pi(i)}\right]\\&=\max_{u\in\{1,\cdots,n\}}\left[\max\left\{\sum_{i=1}^{u}p_{\pi(i)}+\sum_{i=u}^{n}q_{\pi(i)},\ A_{\pi(u)}+\sum_{i=u}^{n}q_{\pi(i)}\right\}\right]\\&=\max\left[\max_{u\in\{1,\cdots,n\}}\left\{\sum_{i=1}^{u}p_{\pi(i)}+\sum_{i=u}^{n}q_{\pi(i)}\right\},\ \max_{u\in\{1,\cdots,n\}}\left\{A_{\pi(u)}+\sum_{i=u}^{n}q_{\pi(i)}\right\}\right].\end{aligned} \quad (2)$$

여기서, 작업 j가 k보다 선행하는 작업순열을 π^1, 그리고 π^1에서 작업 j와 k를 서로 맞교환한 작업순열을 π^2라고 하자. 그러면, (조건 1)과 (조건 3) 그리고 Johnson의 법칙[5]에 의해 다음 부등식이 성립한다. 즉,

$$\begin{aligned}&\max_{u\in\{1,\cdots,n\}}\left\{\sum_{i=1}^{u}p_{\pi^1(i)}+\sum_{i=u}^{n}q_{\pi^1(i)}\right\}\\&\le\max_{u\in\{1,\cdots,n\}}\left\{\sum_{i=1}^{u}p_{\pi^2(i)}+\sum_{i=u}^{n}q_{\pi^2(i)}\right\}.\end{aligned} \quad (3)$$

한편, (조건 2)와 (조건 3)으로부터 다음 부등식이 성

립함도 알 수 있다. 즉,

$$\max_{u\in\{1,\cdots,n\}}\left\{A_{\pi^1(u)}+\sum_{i=u}^{n}q_{\pi^1(i)}\right\}$$
$$\le \max_{u\in\{1,\cdots,n\}}\left\{A_{\pi^2(u)}+\sum_{i=u}^{n}q_{\pi^2(i)}\right\}. \quad (4)$$

식 (2), 식 (3), 식 (4)을 함께 고려함으로써 본 정리가 증명됨을 알 수 있다.

다음 정리는 특정 조건이 만족하면 항상 성립하는 부등식을 나타내고 있다. 본 정리는 이후에 기술되는 (정리 3)의 증명에 활용하기 위해 제시하였다.

정리 2 : 두 양의 정수 p와 q에 대해 $p\le q$이면 임의 정수 α, β에 대해 다음 부등식이 성립한다. 즉,

$$\max(\alpha-p,\ \beta)+q\ge\max(\alpha,\ \beta). \quad (5)$$

증명 : 본 정리는 다음과 같이 두가지 경우로 나누어 증명한다. 즉, (a)$\alpha-p\ge\beta$, (b)$\alpha-p<\beta$.

(a) $\alpha-p\ge\beta$인 경우
이 경우의 조건으로부터 식 (5)의 좌변은 $\alpha-p+q$가 된다. 또, p가 양의정수이므로, 이 경우의 조건으로부터 $\alpha>\beta$가 성립하여 식 (5)의 우변은 α가 된다. 여기서, $p\le q$임을 고려하면 식 (5)가 성립함을 알 수 있다.

(b) $\alpha-p<\beta$인 경우
이 경우의 조건에 의해 식 (5)의 좌변은 $\beta+q$가 된다. 여기서, 다시 $\alpha\ge\beta$인 경우와 그렇지 않은 경우로 세분화하여 증명한다.

(b-1) $\alpha\ge\beta$인 경우
세부조건 $\alpha\ge\beta$에 의해 식 (5)의 우변은 α이고, 조건 $\alpha-p<\beta$에 의해 $\alpha<\beta+p$가 성립한다. 다시 조건 $p\le q$에 의해 $\alpha<\beta+q$가 성립하므로 이 경우 식 (5)가 성립함을 알 수 있다.

(b-2) $\alpha<\beta$인 경우
이 세부경우의 조건에 의해 식 (5)의 우변이 β이고 q가 양의정수이므로 $\beta<\beta+q$가 당연히 성립하여 역시 식 (5)가 성립한다.

다음에 논의하는 우월성질은 작업들 중 일부의 순서가 결정된 부분작업계획(partial sequence)이 있는 경우에 적용된다. 이후에서는 순서가 이미 결정된 작업들에 대해 내부 부품생산기계 M_p와 조립기계 M_q가 작업을 마치는 시간을 각각 T_p와 T_q로 나타내기로 한다(이후 부터 이를 기계 M_p와 M_q의 이용가능시점이라고 지칭한다). 또, 논의의 편의를 위해 순서가 확정된 작업을 제외한 나머지 작업들(순서 미확정 작업들)의 개수를 n으로 표시하고, 현재 분지한계트리에서 고려 중인 노드의 부모노드에서 계산한 하한값을 L로 표기하기로 한다(현재 노드가 분지한계트리의 루트노드 직하위 노드라면 L값은 원문제의 하한값). 부분작업계획이 있는 경우, 본 문제의 최대작업완료시간을 나타내는 식 (1)은 다음의 식 (6)과 같이 수정된다.

본 문제의 특성상 내부 부품기계 M_p에서는 유휴시간이 발생하지 않고, 작업순서에 따라 조립기계 M_q에서 유휴시간이 발생할 수 있다. 따라서, M_q에서의 유휴시간을 최소화하는 해가 본 문제의 최적해가 된다. 아래 (정리 3)의 (조건 2)를 만족하는 순서미결정 작업이 존재할 경우, 이 작업을 현재 부분작업계획 직후에 처리한다면 조립기계 M_q에 유휴시간을 발생시키지 않으면서 조립처리가 가능하게 된다. 이에 더하여, 동 작업이 (정리 3)의 (조건 1)까지 만족한다면 이 작업은 현재 순서미결정 작업들 중에서 가장 앞에 처리하는 것이 그렇지 않은 경우에 비해 동일하거나 우월한 결과를 가져오게 된다.

$$C^{\pi}_{\max}=\max\left[\begin{array}{l}\max_{u\in\{1,\cdots,n\}}\left\{\begin{array}{l}\max\left(T_p+\sum_{i=1}^{u}p_{\pi(i)},A_{\pi(u)}\right)\\ +\sum_{i=u}^{n}q_{\pi(i)}\end{array}\right\},\\ T_q+\sum_{i=1}^{n}q_i\end{array}\right]. \quad (6)$$

정리 3 : 기계 M_p와 M_q의 이용가능 시점이 각각 T_p와 T_q로 주어지고, 부모노드의 하한값이 L이라고 할 때, 다음 조건들을 만족하는 작업 j가 존재한다면, 작업 j가 (순서미결정 작업들 중) 최초에 위치하는 일정계획이 그렇지 않은 일정계획보다 작거나 같은 최대작업완료시간을 갖는다.
(조건 1) $p_j\le q_j$,
(조건 2)
$$\max(T_p+p_j,\ A_j)\le\max\left(T_q,\ L-\sum_{i=1}^{n}q_i\right).$$

증명 : 순서미결정 작업들 중 작업 j가 $\tilde{j}(\tilde{j}>1)$번째 위치에 있는 일정계획 π^*가 최적 일정계획이라고 가

정하자. 그리고, 일정계획 π^*에서 작업 j만을 원래 위치에서 최초 위치로 이동시킨 새로운 일정계획을 π라고 하면 $\pi=(j, \pi^*(1), \cdots, \pi^*(\tilde{j}-1), \cdots, \pi^*(n))$이 된다. 이후에서, 새로운 일정계획 π가 최적일정계획임을 보임으로써 본 정리가 성립함을 증명한다.

식 (6)의 값을 결정하는 특정 $u(u=1, \cdots, n)$에 대해 일정계획 π의 최대 작업완결시간은 다음과 같다. 즉,

$$C_{\max}^{\pi}=\max\left[\max\left\{T_p+\sum_{i=1}^{u}p_{\pi(i)}, A_{\pi(u)}\right\}+\sum_{i=u}^{n}q_{\pi(i)}, \; T_q+\sum_{i=1}^{n}q_i\right].$$

여기서, $C_{\max}^{\pi}=T_q+\sum_{i=1}^{n}q_i$라면 식 (6)으로부터 $C_{\max}^{\pi^*} \geq T_q+\sum_{i=1}^{n}q_i$가 성립하므로 명백히 일정계획 π는 최적일정계획이다. 따라서, 이후의 증명에서는 다음 식이 성립하는 경우로 논의를 한정한다. 즉,

$$C_{\max}^{\pi}=\max\left\{T_p+\sum_{i=1}^{u}p_{\pi(i)}, A_{\pi(u)}\right\}+\sum_{i=u}^{n}q_{\pi(i)}. \quad (7)$$

이하에서는 u값을 기준으로 세 가지 경우 즉, (a) $u=1$, (b) $1<u\leq\tilde{j}$, 그리고 (c) $\tilde{j}<u\leq n$으로 구분하여 증명을 진행한다.

(a) $u=1$인 경우

식 (7)과 본 정리의 (조건 2)로부터 다음식이 성립함으로써, 작업계획 π가 최적일정계획임을 알 수 있다.

$$C_{\max}^{\pi}=\max\{T_p+p_j, A_j\}+\sum_{i=1}^{n}q_i \leq \max\left(T_q+\sum_{i=1}^{n}q_i, L\right).$$

(b) $1<u\leq\tilde{j}$인 경우

식 (7)로부터 $C_{\max}^{\pi^*}$의 하한값을 다음과 같이 나타낼 수 있다. 즉,

$$C_{\max}^{\pi^*} \geq \max\left\{T_p+\sum_{i=1}^{u-1}p_{\pi^*(i)}, \; A_{\pi^*(u-1)}\right\}+\sum_{i=u-1}^{n}q_{\pi^*(i)}. \quad (8)$$

또, u값이 $1<u\leq j'$인 경우에 일정계획 π^*와 π의 관계로부터, 다음과 같은 작업집합 간 관계와 작업간 관계를 파악할 수 있다.

$$\{\pi^*(1), \cdots, \pi^*(u-1)\}=\{\pi(1), \cdots, \pi(u)\}-\{j\},$$
$$\{\pi^*(u-1), \cdots, \pi^*(n)\}=\{\pi(u), \cdots, \pi(n)\}\cup\{j\},$$
$$\pi^*(u-1)=\pi(u).$$

이 관계를 식 (8)에 적용함으로써 다음 부등식이 도출된다. 즉,

$$C_{\max}^{\pi^*} \geq \max\left\{T_p+\sum_{i=1}^{u}p_{\pi(i)}-p_j, \; A_{\pi(u)}\right\}+\sum_{i=u}^{n}q_{\pi(i)}+q_j. \quad (9)$$

여기서, (조건 1), 식 (7), 식 (9)을 함께 고려하면서 $T_p+\sum_{i=1}^{u}p_{\pi(i)}$, $A_{\pi(u)}$, p_j, q_j를 (정리 2)의 α, β, p, q로 각각 대체하면 $C_{\max}^{\pi} \leq C_{\max}^{\pi^*}$가 성립한다.

(c) $\tilde{j}<u\leq n$인 경우

식 (7)로부터 다음 부등식이 성립함을 알 수 있다. 즉

$$C_{\max}^{\pi^*} \geq \max\left\{T_p+\sum_{i=1}^{u}p_{\pi^*(i)}, A_{\pi^*(u)}\right\}+\sum_{i=u}^{n}q_{\pi^*(i)}. \quad (10)$$

또, u값이 $j'<u\leq n$인 경우에 대해서, 순열 π^*와 π의 관계로부터 다음과 같은 작업집합 간 관계와 작업간 관계를 파악할 수 있다.

$$\{\pi^*(1), \cdots, \pi^*(u)\}=\{\pi(1), \cdots, \pi(u)\},$$
$$\{\pi^*(u), \cdots, \pi^*(n)\}=\{\pi(u), \cdots, \pi(n)\},$$
$$\pi^*(u)=\pi(u).$$

따라서, 위 식 (10)의 우변값은 $C_{\max}^{\pi}$ 값과 일치하므로, $C_{\max}^{\pi} \leq C_{\max}^{\pi^*}$가 증명된다.

3. 하한값(Lower bounds)

본 절에서는 분지한계 알고리즘에서 특정 노드제거와 발견적 해법의 성능분석에 공통적으로 활용될 하한값을 제시하였다. 다음 정리의 부등식은 임의 작업 j의 조립작업은 해당 작업이 필요로 하는 모든 외주부품들의 조달이 완결된 시점 이후에 가능하다는 사실에 기반하여 도출되었다.

정리 4 : 본 연구문제의 최적일정계획을 π^*라고 하고, 각 작업 $j(j=1, \cdots, n)$의 외주조달시점 A_j를 기준으로 작업들을 오름차순으로 정렬하여 도출한 일정계획을 π^A라고 하면 다음 부등식이 항상 성립한다. 즉,

$$C_{\max}^{\pi^*} \geq \max_{u\in\{1,\cdots,n\}}\left(A_{\pi^A(u)}+\sum_{j=u}^{n} q_{\pi^A(j)}\right).$$

증명 : 본 연구문제에서 내부 부품생산시간들(p_j)을 제외하고, 외주 부품조달시간들(A_j)과 조립시간들(q_j)만을 고려한(축소된) 문제 $\tilde{P}$를 고려하면 일정계획 π^A가 $\tilde{P}$의 최적일정계획임을 알 수 있으므로, 다음 부등식이 성립한다. 즉,

$$C_{\max}^{\pi^A} \leq C_{\max}^{\pi^*}. \tag{11}$$

여기서, $C_{\max}^{\pi^A}$를 다음과 같이 정리할 수 있다. 편의를 위해, 축소된 문제 $\tilde{P}$의 최적작업계획 π^A의 j번째 작업 $\pi^A(j)$의 작업완료시점을 $C_{\pi^A(j)}$로 나타내기로 하자. 그러면, 다음의 식들이 만족함을 알 수 있다. 즉,

$$C_{\pi^A(1)} = A_{\pi^A(1)} + q_{\pi^A(1)},$$
$$C_{\pi^A(j)} = \max\left(C_{\pi^A(j-1)}, q_{\pi^A(j)}\right),\ j=2, 3, \cdots, n.$$

이 n개의 등식들로부터 다음의 식이 유도된다. 즉,

$$C_{\pi^A(n)} = \max_{u\in\{1,\cdots,n\}}\left(A_{\pi^A(u)}+\sum_{j=u}^{n} q_{\pi^A(j)}\right). \tag{12}$$

여기서, $C_{\pi^A(n)} = C_{\max}^{\pi^A}$ 이므로 식 (11)와 식 (12)로부터 본 정리의 부등식이 성립함을 알 수 있다.

(정리 4)로부터 첫 번째 하한값을 다음과 같이 표기할 수 있다. 즉,

$$LB_1 = \max_{u\in\{1,\cdots,n\}}\left(A_{\pi^A(u)}+\sum_{j=u}^{n} q_{\pi^A(j)}\right).$$

본 연구에서는 LB_1과 전재호[1]의 (정리 4), (정리 5)에서 다음과 같이 제시한 하한값을 함께 사용한다. 논의의 완결성을 위해 전재호[1]의 (정리 4)와 (정리 5)에서 제시한 하한값을 각각 LB_2와 LB_3로 나타내면 다음과 같다.

$$LB_2 = \max_{u\in\{1,\cdots,n\}}\left(\sum_{i=1}^{u} p_{\pi^J(i)}+\sum_{i=u}^{n} q_{\pi^J(i)}\right),$$

여기서 π^J는 원문제에서 외주조달시점 A_j를 배제하고, 내부생산 및 조립시간 p_j와 q_j만 존재하는 축소된 문제(2-machine flowshop문제)에 Johnson의 법칙[5]을 적용하여 구한 최적일정계획을 의미한다.

$$LB_3 = \min_{i\in\{1,\cdots,n\}}\{\max(p_i,\ A_i)\}+\sum_{j=1}^{n} q_j.$$

참고로, 전재호[1]의 (정리 6)에 의해 생성된 하한값 $(\max_{i\in\{1,\cdots,n\}}(A_i)+q_{\arg\max_{i\in\{1,\cdots,n\}}(A_i)})$은 본 연구의 LB_1보다 항상 작거나 같음을 쉽게 알 수 있으므로, 전체하한값 계산에서 배제하였다.

이후에 논의되는 분지한계 알고리즘에서 특정노드 제거여부 결정과 작업수가 큰 문제의 발견적해 성능평가에서는 이상에서 논의한 하한값들 중 가장 큰 값을 다음과 같이 계산하여 활용한다. 즉,

$$LB = \max(LB_1,\ LB_2,\ LB_3). \tag{13}$$

특히, 분지한계 알고리즘에서는 특정 노드에서 이미 순서가 확정된 부분일정계획이 존재하므로, 하한값 계산시에 이러한 사실을 적절히 반영하여야 한다.

4. 분지한계 알고리즘

4.1 분지한계 알고리즘 개요

본 절에서는 작업 수가 적은 경우 최적해를 제공해주는 분지한계 알고리즘에 대해 소개한다. 본 분지한계 알고리즘에서는 전재호[1]와 동일하게 순방향 작업순서결정 분지규칙(forward sequencing branching rule)을 사용한다. 즉, 분지한계 탐색트리의 l번째 계위에 있는 노드들에서는 작업일정계획 순열의 앞에서부터 l번째 위치의 작업을 결정하게 된다. 탐색트리의 각 노드들 중에서 일차적으로 가장 큰 부분순열(작업순서가 가장 많이 확정된 부분일정계획)을 가진 노드를 분지대상 노드로 결정하고(depth-first), 동일한 크기의 부분순열을 갖는 노드가 다수개일 경우는 가장 작은 하한값을 갖는 노드를 선택한다(best-first). 분지노드를 선택한 후(l번째 위치의 작업을 결정한 후), 해당 노드가 제거가능한지 여부는 제 2장에서 논의한 우월성질(정리1, 3)들과 전재호[1]의 (정리 3)

을 적용하여 확인한다.

우월성질을 적용하여도 제거되지 않은 노드들에 대해서는 식 (13)의 하한값을 계산하여 상한값보다 큰 경우 역시 제거한다. 특정 노드의 하한값은 해당노드의 부모노드에서 순서가 기확정된 부분작업계획 이후에 순서미결정 작업들을 최적으로 나열했을 때, 가질 수 있는 최대작업완료시간보다 작거나 같은 값을 의미한다. 따라서, 특정 노드의 하한값이 상한값(이미 발견된 실행가능일정계획의 최대작업완료시간)보다 크다면 해당 노드를 포함하여 그 하부노드 모두는 어떠한 경우에도 분지한계 알고리즘이 구하려고 하는 최적해가 될 수 없음이 명백하므로, 이를 제거하는 것은 최적해 산출에 영향을 주지 않으면서 동시에 불필요한 탐색시간을 줄여주는 역할을 한다. 분지한계 알고리즘에서 하한값과 비교되는 상한값은 일차적으로 다음 제 4.2절에서 논의하는 발견적 해법(heuristic algorithm)을 사용하여 계산된다. 이후, 분지한계 알고리즘이 n번째 계위까지 진행되어 실행가능해(feasible solution)가 생성될 때마다 현재의 상한값과 크기를 비교하여, 새 실행가능해가 기존 상한값보다 작은 경우 상한값을 개선한다.

4.2 발견적 해법

본 연구문제가 NP-Complete이므로 작업 수가 큰 문제의 근사해를 신속히 제공하는 발견적해법의 개발이 필수적이다. 뿐만 아니라, 본장의 분지한계 알고리즘에 최초 실행가능해(상한값)를 제공해야 한다는 측면에서도 발견적해법이 필요하다. 특히, 최초 상한값의 최적해 근사정도가 분지한계 알고리즘의 성능향상에 일차적으로 중요한 역할을 수행하므로 발견적해법의 성능은 근사해 제공이라는 본연의 목적은 물론, 분지한계 알고리즘의 성능향상 측면에서도 중요하다.

본 연구에서는 전재호[1]에서 제시된 "Johnson법칙 기반 휴리스틱"을 수정하여 세 가지 발견적 해법들을 제시하였다. 전재호[1]의 발견적 해법은 "모든 대상작업들에 대해 외주조달시점이 제약으로 작동하는 경우에는 외주조달시점이 가장 빠른 작업을 우선 처리하고, 그렇지 않은 경우에는 Johnson법칙을 적용하여 우선 처리할 작업을 선택"하고 있다. 본 연구에서 제시하고 있는 세 가지 발견적 해법도 기본적으로 이 개념을 사용하면서 기존 해법의 약점을 가능한 한 개선한 것이다. 즉, 전재호[1]의 해법은 모든 대상작업들에 대해 외주조달 시점이 일정계획의 제약으로 작동하는 경우 가장 빠른 외주조달시간을 갖는 작업을(외주조달시간만을 고려하여) 우선적으로 선택하는 편향된 방법으로써, 불필요한 유휴시간을 발생시킬 가능성이 있다. 아래의 수정해법 MH_1은 "단계 1"이 추가된 것을 제외한다면 전재호[1]의 해법과 동일하다. 반면, 두 번째 수정해법 MH_2는 모든 대상작업들에 대해 외주 조달시간이 제약으로 작동하는 상황 하에서 외주조달시간과 더불어 내부부품작업시간 및 조립시간을 함께 고려하여 처리작업을 선택("단계 3"에서)함으로써, MH_1의 편향성을 보완하는 기능을 수행한다. 마지막 수정해법인 MH_3는 MH_1과 MH_2에서 선택한 작업이 상이할 경우, "단계 3"에서 이 두 작업을 순서확정된 부분일정계획 직후에 순서를 맞교환해 가면서 완료시간(부분일정계획과 두 작업을 완결하는 시간)을 계산한 후, 완료시간이 짧은 쪽으로 순서를 결정하고 있다. 세 가지 수정해법에 공통적으로 사용되고 있는 "단계 1"은 순서미결정 작업 중 조립기계 M_q에서 유휴시간을 가장 적게 발생시키는 작업을 선별하는 기능을 수행한다. 세 가지 해법중 MH_3의 성능이 평균적으로는 가장 우수하지만, 모든 문제상황에 대해 어느 해법도 항상 우월한 해를 제공하지는 못함이 실험을 통해 확인되어, 이후의 논의에서 발견적 해(상한값)는 $MIN_{i\in\{1,2,3\}}(MH_i)$ 값을 사용한다.

<수정 휴리스틱1 : MH_1>

단계 **0** : (초기화)

- $J^u=\{1,\cdots,n\},\ u=0,\ T_p=0,\ T_q=0.$

단계 **1** : (할당작업찾기 1)

- 집합 J^u의 모든 원소들에 대해, 다음 두 조건이 만족되는 작업x가 있으면 (단계 4)로 이동하고, 그렇지 않으면 (단계 2)로 이동

  (조건 1) $p_x \le q_x$,

  (조건 2) 모든 작업 $j\in J^u$에 대해

  $$\max(T_p+p_x,\ A_x) \le \max(T_p+p_j,\ A_j).$$

단계 **2** : (할당작업찾기 2)

- 기준시점 R 계산, 여기서

  $R=\max\{T_q,\min_{j\in J^u}(T_p+p_j)\}$.

- 할당작업후보집합 Ω와 Ω' 찾기, 여기서

  $\Omega=\{j|A_j \le R, p_j \le q_j, j\in J^u\}$,

  $\Omega'=\{j|A_j \le R, p_j > q_j, j\in J^u\}$.

- $\Omega=\Omega'=\varnothing$이면 (단계 3)으로 이동하고, 그렇지 않으면 다음과 같이 작업 x를 설정한 후 (단계 4)로 이동. 즉, $\Omega\neq\varnothing$이면 $x=\arg\min_{j\in\Omega}(p_j)$, $\Omega=\varnothing$이고 $\Omega'\neq\varnothing$이면 $x=\arg\max_{j\in\Omega'}(q_j)$.

단계 **3** : (할당작업찾기 3)

- $x=\arg\min_{j\in J^u}(A_j)$.

단계 **4** : (선택작업 할당 및 중요변수 수정)

- $u=u+1$로 갱신한 후, u번째 위치에 작업 x할당.
- $T_p=T_p+p_x,\ T_q=\max(T_q,\ A_x,\ T_p)+q_x,$

$J^u = J^u - \{x\}$.

단계 5 : (종료조건 검사)

- $J^u = \varnothing$ 이면 종료하고, 아니면 (단계 1)로 이동.

<수정 휴리스틱2 : MH_2>

단계 0~**단계 2** 그리고 **단계 4**~**단계 5.**

- MH_1의 대응단계와 동일

단계 3 : (할당작업찾기 3)

- $x = \arg\min_{j \in J^u}\{\max(p_j,\ A_j - T_p)/q_j\}$.

<수정 휴리스틱 3 : MH_3>

단계 0~**단계 2** 그리고 **단계 4**~**단계 5.**

- MH_1의 대응단계와 동일

단계 3 : (할당작업찾기 3)

- 할당 후보 작업 y와 z찾기, 여기서
  $y = \arg\min_{j \in J^u}(A_j)$,
  $z = \arg\min_{j \in J^u}\{\max(p_j,\ A_j - T_p)/q_j\}$.
- 현재 확정된 부분일정계획 바로 뒤에 작업 y, z순으로 추가했을 때의 완료시점 T_{qyz}와 반대로 z, y 순으로 추가했을 때의 완료시점 T_{qzy} 계산, 즉

$T_{qyz} = \max\{T_{qy},\ A_z,\ T_p + p_y + p_z\} + q_z$,
여기서 $T_{qy} = \max\{T_q, A_y, T_p + p_y\} + q_y$,
$T_{qzy} = \max\{T_{qz}, A_y, T_p + p_z + p_y\} + q_y$,

여기서 $T_{qz} = \max\{T_q,\ A_z, T_p + p_z\} + q_z$.

- 할당작업 결정
  $T_{qyz} \le T_{qzy}$ 이면 $x = y$,
  $T_{qyz} > T_{qzy}$이면 $x = z$로 설정.

발견적 해법들의 진행과정에 대한 이해를 위해 <표 1>에 세 개의 작업으로 구성된 간단한 예제를 제시하였다. 각 해법별 진행과정은 다음과 같다. 편의상 작업 i를 J_i로 표기한다.

<표 1> 발견적 해법 진행과정 이해를 위한 예시

작업번호(i)	p_i	A_i	q_i
1	14	13	2
2	3	14	5
3	2	3	7

<휴리스틱 MH_1의 진행과정>

(1) 최초 처리작업 결정과정

- (단계 1)에서 J_3가 (단계 1의 두조건)을 만족시키므로 J_3가 최초처리작업으로 결정됨.
- $T_p = 2$, $T_q = 10$, $J^u = \{J_1, J_2\}$로 갱신.

(2) 두 번째 처리작업 결정과정

- (단계 1)에서 현재 순서미결정 작업인 J_1과 J_2는 조건을 만족하지 못함.
- (단계 2)에서 $\Omega = \Omega' = \varnothing$ 이므로 순서결정 안됨.
- (단계 3)에서 $A_1 < A_2$이므로 J_1 선택.
- $T_p = 16$, $T_q = 18$, $J^u = \{J_2\}$로 갱신.

(3) 세 번째 처리작업 결정과정

- (단계 1)에서 조건을 만족하여 J_2를 세 번째 작업으로 결정.
- 결과적으로 MH_1이 결정한 작업순서는 (J_3, J_1, J_2).

<휴리스틱 MH_2의 진행과정>

(1) 최초 처리작업 결정과정-MH_1과 동일함.

(2) 두 번째 처리작업 결정과정

- (단계 1)에서 현재 순서미결정 작업인 J_1과 J_2는 조건을 만족하지 못함.
- (단계 2)에서 $\Omega = \Omega' = \varnothing$ 이므로 순서결정 않됨.
- (단계 3)에서 $x = 2$가 되므로 J_2선택.

(3) 세 번째 처리작업 결정과정

- (단계 1)에서 조건을 만족하지 못함.
- (단계 2)에서 J_1 선택 .
- 따라서, MH_2가 결정한 작업순서는 (J_3, J_2, J_1).

<휴리스틱 MH_3의 진행과정>

(1) 최초 처리작업 결정과정-MH_1과 동일함

(2) 두번째 처리작업 결정과정

- (단계 1)에서 현재 순서미결정 작업인 J_1과 J_2는 조건을 만족하지 못함
- (단계 2)에서 $\Omega = \Omega' = \varnothing$ 이므로 순서결정 안됨
- (단계 3)에서 $y = 1$이고 $z = 2$로 상이하므로 현재 순서확정된 작업 3 뒤에 (J_1, J_2)순으로 작업을 나열했을때의 J_2완료시간 T_{q12}과 작업 3 뒤에 (J_2, J_1)순으로 나열했을때의 J_1완료시간 T_{q21}를 각각 계산하여 $T_{q21} < T_{q12}$이므로 작업 2 선택

(3) 세 번째 처리작업 결정과정-MH_2의 세 번째 처리작업 결정과정과 동일. 따라서, MH_3가 결정한 작업순서는 MH_2가 결정한 작업순서와 동일함

5. 계산실험 및 성능평가

본 절에서는 개선된 분지한계 알고리즘(하한값 포함)

과 발견적 해법의 성능을 수치실험을 통해 전재호[1]의 알고리즘들과 비교분석한다. 알고리즘들은 C언어로 프로그래밍 되었고, Pentium IV 3GHz PC에서 실행되었다. 실험에 사용된 수치의 생성방법도 다음과 같이 전재호[1]와 동일하게 설정하였다.

(1) 각 작업의 부품가공시간(p_i)과 조립시간(q_i)

- p_i와 q_i의 변이가 상대적으로 큰 문제유형 (문제유형 1) : $U[1, 50]$으로부터 생성, 여기서 $U[a, b]$는 a와 b를 모수로 갖는 이산형 균등분포(discrete uniform distribution)을 의미.
- p_i와 q_i의 변이가 상대적으로 작은 문제유형 (문제유형 2) : $U[\rho+1, \rho+10]$으로부터 생성, 여기서 ρ 값은 $U[1, 50]$으로부터 생성.

(2) 각 작업의 외주부품 조달시간 (A_i) : $U[1, \alpha P]$로부터 생성, 여기서 $P=\sum_{i=1}^{n} p_i$, $\alpha \in \{0.4, 0.6, 0.8, 1.0\}$.

- α는 외주부품 조달시점의 산포를 조절하기 위한 계수, α값이 클수록 외주부품의 조달시간 편차가 커짐

위에서 설정한 "문제유형 1과 2"는 내부 작업시간과 외주부품조달시간의 분산이 다르게 나타나도록 설정한 것이다. 그 차이를 보기 위한 간단한 예제가 <표 2>에 나타나있다. 표에서 $A_i^{\alpha}(\alpha \in \{0.4, 0.6, 0.8, 1.0\})$는 산포계수별 외주부품조달시점을 나타낸다. 또, 표에 나타난 "문제유형 2"의 값들은 ρ값을 25(1과 50의 중간정도값)로 가정하고 산출된 값이다. 두 문제유형에 대해 p_i와 q_i의 분산값을 비교해보면 "문제유형 1"이 "문제유형 2"보다 각각 27배와 35배로 매우 크다는 것을 알 수 있다. 반면 A_i^{α}값은 "문제유형 2"의 분산값이 약간 크게 (1.1에서 1.4배정도) 나타난다.

<표 2> 문제유형별 내부작업시간/외주조달시점 사례

	작업 i	p_i	q_i	$A_i^{0.4}$	$A_i^{0.6}$	$A_i^{0.8}$	$A_i^{1.0}$
문제유형 1	1	28	45	13	22	17	19
	2	2	11	30	45	45	77
	3	49	5	9	7	61	49
	분산	**554**	**465**	**124**	**366**	**496**	**841**
문제유형 2	1	34	26	7	8	25	26
	2	25	31	33	49	37	85
	3	29	33	19	21	69	41
	분산	**20**	**13**	**169**	**439**	**517**	**940**

이하에서는 알고리즘들의 성능평가를 위한 수치실험이 작업수가 적은 경우와 큰 경우로 나뉘어 각각 진행된다.

<표 3> 분지한계 알고리즘 성능비교

	n	α	전재호[1] B&B사용			본 연구 B&B 사용		
			NO	ACT	ANN	NO	ACT	ANN
문제유형 1	30	0.4	30	≪0.01*	47.90	30	≪0.01	11.67
		0.6	30	≪0.01	70.00	30	≪0.01	23.37
		0.8	30	0.96	4,051.33	30	≪0.01	43.23
		1.0	25	N/A**	N/A	29	3.46	46,985.20
	40	0.4	30	≪0.01	66.67	30	≪0.01	18.17
		0.6	30	≪0.01	74.03	30	≪0.01	14.20
		0.8	29	49.60	98,763.37	30	0.01	79.00
		1.0	18	N/A	N/A	28	N/A	N/A
	50	0.4	30	0.01	92.73	30	≪0.01	16.07
		0.6	30	0.01	159.97	30	≪0.01	46.83
		0.8	29	N/A	N/A	30	0.01	69.43
		1.0	16	N/A	N/A	26	N/A	N/A
문제유형 2	30	0.4	30	≪0.01	19.07	30	≪0.01	6.77
		0.6	30	≪0.01	20.50	30	≪0.01	7.67
		0.8	30	0.73	2,937.30	30	≪0.01	9.47
		1.0	28	N/A	N/A	30	≪0.01	44.27
	40	0.4	30	≪0.01	27.80	30	≪0.01	4.90
		0.6	30	≪0.01	34.90	30	≪0.01	8.43
		0.8	30	1.74	3,655.60	30	≪0.01	13.20
		1.0	21	N/A	N/A	30	0.01	37.63
	50	0.4	30	≪0.01	72.30	30	≪0.01	20.57
		0.6	30	0.01	74.43	30	≪0.01	22.77
		0.8	29	15.73	17,337.93	30	≪0.01	6.37
		1.0	21	N/A	N/A	29	104.69	532,447.00

주) n : 작업수.
α : 외주부품 조달시점 산포조절 계수.
NO : 30개의 실험문제중에서 분지한계 알고리즘이 60초 이내에 최적해를 찾은 문제수.
ACT : 평균계산시간.
ANN : 평균탐색노드수.
* 평균계산시간이 0.01초보다 매우 작음을 의미.
** 30개의 실험문제중에서 7,200초를 초과하여도 최적해를 찾지 못하는 문제가 일부 존재하여 통계치를 제공하지 못함.

일차적으로, 상대적으로 적은 수의 작업을 갖는 문제들에 대해 본 연구와 전재호[1]에서 제시한 분지한계 알고리즘을 이용한 성능실험을 수행한다. 또, 두 연구에서 제안한 발견적해법의 성능비교를 위해 분지한계 알고리즘의 결과로 도출된 최적해 대비 상대오차를 각각 계산하여 비교한다. 이를 위해, 작업수(n)를 3종류(30, 40, 50), 문제유형은 위에서 논의한 "문제유형 1"과 "문제유형 2"의 두 가지, 그리고 각각에 대해 "외주부품 조달시점 산포계수(α)"를 4종류(0.4, 0.6, 0.8, 1.0)로 전재호[1]와 동일

하게 설정하였다. 따라서, 문제의 총 조합수는 24가지(3 × 2 × 4)이고, 이들 각 조합에 대해 30개씩의 실험문제를 임의로 생성하여 사용함으로써, 총 720개의 문제가 사용되었다(작업수가 10개, 20개인 경우는 전재호[1]의 알고리즘도 비교적 빠르게 최적해를 생성하므로 고려대상에서 배제하였다).

두 연구에서 제안된 분지한계 알고리즘의 성능비교는 <표 3>에 정리되어 있다. 표에서 NO, ACT, ANN은 각각 "문제조합별로 생성된 30개의 문제 중 분지한계 알고리즘이 60초 이내에 최적해를 도출한 문제수", "분지한계 알고리즘이 최적해 계산에 사용한 CPU 시간의 평균값(초)", "분지한계 알고리즘이 최적해를 도출할 때까지 탐색한 노드의 평균갯수"를 각각 의미한다. 단, 임의 생성된 30개의 문제 중 계산시간이 2시간(7,200초)을 초과하여도 최적해를 찾지 못하는 경우는 계산을 강제종료하도록 설정하였다. 따라서, 특정 문제조합에 대해 최적해를 찾지 못하고 종료된 문제가 하나라도 있는 경우 ACT와 ANN값은 계산되지 못하므로 N/A로 표시되어져 있다.

〈표 4〉 평균탐색노드수(ANN)의 감소정도

n	α	$\left(\frac{100\times\text{본 연구 분지한계법의 }ANN}{\text{전재호[1] 분지한계법의 }ANN}\right)$ (단위 : %)	
		문제유형 1	문제유형 2
30	0.4	24.36	35.49
	0.6	33.38	37.40
	0.8	1.07	0.32
	1.0	N/A*	N/A
40	0.4	27.25	17.63
	0.6	19.18	24.16
	0.8	0.08	0.36
	1.0	N/A	N/A
50	0.4	17.33	28.45
	0.6	29.28	30.59
	0.8	N/A	0.04
	1.0	N/A	N/A
평균		18.99**	19.38**

주) n : 작업수.
α : 외주부품 조달시점 산포조절 계수.
* 30개의 실험문제중에서 7,200초를 초과하여도 최적해를 찾지 못하는 문제가 일부 존재하여 통계치를 제공하지 못함.
** 통계치가 제시되지 못한 문제조합(N/A로 표시된 문제조합)은 배제하고 평균값을 계산함.

<표 3>의 비교실험결과에서 나타나는 전반적인 경향은 다음과 같다. 첫째, 전재호[1]의 분지한계법을 사용한 경우 "문제유형 1"과 "문제유형 2"에 대해 공통적으로 모든 작업수에 대해 α값이 1.0인 경우 2시간 이내에 최적해를 찾지 못하는 문제가 존재하는 반면, 본 연구에서 제시한 분지한계 해법은 "문제유형 1"의 경우 작업수 30개까지 α값에 무관하게 최적해를 도출하고, "문제유형 2"에 대해서는 50개의 작업을 갖는 문제까지 모두 최적해를 일정시간 내에 계산해 줌으로써 분지한계 알고리즘의 성능향상이 확연히 나타나고 있다. 평균계산시간(ACT)이나 평균탐색노드수(ANN)도 본 연구의 분지한계법을 사용했을 때 확연히 감소하고 있음을 알 수 있다. ACT의 경우 그 값이 0.01초보다 매우 작은 경우가 많이 나타나(<표 3>에서 《0.01로 표시) 감소정도를 나타내는 비율 계산에 부적합하다. 그런데, ACT는 일반적으로 ANN에 비례하므로 ACT의 감소정도를 대신해서 ANN의 감소정도를 계산해보면 <표 4>와 같다. 표의 값은 문제유형별로 (본 연구 B&B의 평균탐색노드수/전재호[1] B&B의 평균탐색노드수)값을 백분율로 계산한 것이다. 표에 나타난 바와 같이, 본 연구의 분지한계법에 의해 제거되지 않는 평균탐색노드수는 전재호[1]의 그것에 비해 평균적으로 19% 내외로 대략 81% 정도의 노드가 감소되었음을 알 수 있다. 이와 같은 분지한계 알고리즘의 성능향상은 초기상한값을 제공하는 발견적해법의 성능개선, 최적해가 되지 못하는 하부문제(부분일정계획) 제거능력(우월성질) 그리고 하한값의 성능 등에 모두 영향을 받으므로 이들의 성능이 전재호[1]의 그것들보다 향상되었음을 의미한다. 둘째, 전재호[1] 및 본 연구의 분지한계 알고리즘 모두 "문제유형 1"보다는 "문제유형 2"에 대해서 최적해를 상대적으로 잘 찾고 있음을 <표 3>으로부터 알 수 있다(본 연구 B&B가 "문제유형 2"에 대해서는 작업수 50까지 최적해를 모두 찾아주고 있으나, "문제유형 1"에 대해서는 그렇지 못하다). 이는 내부작업시간(p, q)의 분산정도가 작을수록 문제해결이 용이함을 의미한다. 셋째, 작업수 및 내부작업시간의 분산도가 동일한 문제들을 살펴보면 "외주부품 조달시간 산포계수(α)"가 커짐에 따라, 전반적으로 분지한계 알고리즘들이 문제해결을 위해 더 많은 노드를 탐색하고 있다(예를 들어, "문제유형 1"에서 작업수가 30개인 경우에 대해 본연구의 분지한계알고리즘을 사용한 경우 ANN값이 α가 0.4, 0.6, 0.8, 1.0으로 증가함에 따라 11.67, 23.37, 43.23, 46985.2로 각각 증가한다). α가 커질수록 외주부품 조달시간의 분산정도도 커지고, 이에 따라 문제해결의 난이도가 높아진 결과로 풀이된다.

다음 단계에서는, 상대적으로 작업수가 적은 경우에 대해 전재호[1]와 본 연구에서 각각 제시된 발견적 해와 하한값의 성능비교를 수행하였다. 비교실험에 사용된 문제

<표 5> 발견적 해 및 하한값 성능비교(작업수 50 이하)

	n	α	전재호[1] 발견적해와 하한값 사용†				본 연구 발견적해와 하한값 사용‡			
			NO	ARE	MRE	n(B = L)	NO	ARE	MRE	n(B = L)
문제 유형 1	30	0.4	30	0.2512	2.8000	25	30	0.0721	1.2429	25
		0.6	30	0.4085	3.6671	19	30	0.1645	1.7196	24
		0.8	30	0.3523	2.9851	13	30	0.2632	2.9851	21
		1.0	25	0.6585*	4.2007*	5	29	0.1216	1.8293	21
	40	0.4	30	0.0808	1.5326	23	30	0.0296	0.6014	23
		0.6	30	0.1427	1.4952	15	30	0.0299	0.5535	19
		0.8	29	0.5018	4.3358	15	30	0.2696	1.6175	24
		1.0	18	N/A**	N/A	6	28	N/A	N/A	24
	50	0.4	30	0.0897	1.5544	23	30	0.0330	0.6319	23
		0.6	30	0.1507	2.5681	23	30	0.1272	2.5681	26
		0.8	29	0.4129*	4.1478*	11	30	0.1804	2.1277	24
		1.0	16	N/A	N/A	5	26	N/A	N/A	19
문제유형 1 평균			27.25	0.3049***	2.9287***	15.25	29.42	0.1291***	1.5877***	22.75
문제 유형 2	30	0.4	30	0.0697	1.7699	26	30	0.0697	1.7699	26
		0.6	30	0.0976	2.0134	22	30	0.0976	2.0134	24
		0.8	30	0.0381	0.5068	17	30	0.0070	0.2096	25
		1.0	28	0.0185*	0.4124*	3	30	0.0185	0.4124	27
	40	0.4	30	0.0227	0.3752	27	30	0.0125	0.3752	27
		0.6	30	0.0119	0.1556	24	30	0.0083	0.1556	27
		0.8	30	0.0238	0.5710	17	30	0.0238	0.5710	25
		1.0	21	0.0199*	0.2914*	4	30	0.0101	0.1890	26
	50	0.4	30	0.0132	0.1988	27	30	0.0132	0.1988	27
		0.6	30	0.0251	0.4608	15	30	0.0251	0.4608	20
		0.8	29	0.0102	0.3046	12	30	0.0102	0.3046	26
		1.0	21	0.0000*	0.0000*	4	29	0.0000	0.0000	29
문제유형 2 평균			28.25	0.0292	0.5883	16.50	29.92	0.0247	0.5550	25.75
전체 평균			27.75	0.1671***	1.7585***	15.88	29.67	0.0769***	1.0714***	24.25

주) n : 작업수.
α : 외주부품 조달시점 산포조절 계수.
NO : 30개의 실험문제중에서 분지한계 알고리즘이 60초 이내에 최적해를 찾은 문제수.
ARE : 발견적 해의 최적해 대비 평균상대오차(%).
MRE : 발견적 해의 최적해 대비 최대상대오차(%).
n(B = L) : 30개 문제중 최적해(분지한계 알고리즘 결과값)와 Lowerbound값이 동일한 문제수.
† 전재호[1]에서 제안된 방식으로 발견적해와 하한값을 산출하고, 최적해는 본 연구의 분지한계 알고리즘 결과값 사용.
‡ 본 연구에서 제안한 방식으로 계산한 발견적해, 하한값 및 최적해 사용.
* 전재호[1]의 분지한계 알고리즘으로는 7,200초 이내에 최적해를 찾지 못하는 문제가 일부 존재하여, 이 문제들에 대해서는 본 연구에서 제안한 분지한계 알고리즘에서 생성한 최적해를 사용하여 상대오차 계산.
** N/A : 본 연구의 분지한계 알고리즘으로도 7,200초 이내에 최적해를 산출하지 못하는 문제가 일부 존재하여 통계치를 제시하지 못함.
*** 통계치가 제시되지 못한 문제조합(N/A로 표시된 문제조합)은 배제하고 평균값을 계산함.

집합은 직전의 분지한계알고리즘 성능실험에서 사용한 것들과 동일하다. 발견적 해의 성능 비교평가 척도로는 최적해 대비 상대오차(RE; Relative Error)를 사용하였다.

상대오차는 $RE(\%) = 100 \times (Heu - Opt)/Opt$를 사용해 백분율 값으로 계산되었다. 여기서, Heu와 Opt는 각각 발견적해와 최적해를 의미한다. 전재호[1]의 경우 하나의 Heu값만 존재하여 그 값을 그대로 사용하지만 본 연구에서는 제안된 세 가지 발견적 해들 중 가장 작은 해값을

의미한다. 또, Opt는 분지한계 알고리즘이 제공하는 최적해를 사용하였다. 하한값의 성능평가는 각 문제조합에 속한 30개의 문제 중 최적해와 하한값이 동일하게 계산된 문제수(표에서 $n(B=L)$로 표기)를 사용하여 이루어졌다. 최적해와 하한값이 동일하다면 이는 하한값이 최고의 성능을 발휘하였음을 의미하므로 $n(B=L)$값을 사용해 두 연구의 하한값 성능을 비교하였다. 이상의 척도를 사용하여 계산한 비교값이 <표 5>에 요약되어 있다. 표에서 각 문제조합별로 30개의 임의생성 문제들에 대한 상대오차의 평균값(ARE; 평균상대오차)과 최대값(MRE; 최대상대오차), 그리고 "최적해와 하한값이 동일하게 산출된 문제수($n(B=L)$)"가 공통적으로 전재호[1] 대비 본 연구의 발견적 해법과 하한값 성능향상을 보여주고 있다. 즉, 전재호[1] 발견적 해법의 최적해 대비 평균상대오차는 평균 0.1671%인 반면 본 연구에서 제안한 발견적해법의 경우 0.0769%로 감소하였다. 또, $n(B=L)$값도 전재호[1]의 하한값을 사용한 경우 평균 15.88개로 나타났으나, 본 연구의 하한값을 사용한 결과 평균 24.25개로 증가(개선)되었으며, 특히 α값이 증가함에도 $n(B=L)$값이 급속히 감소하지 않는 형태로 하한값의 성능이 개선되었음을 볼 수 있다. 표에 나타난 바와 같이, 전재호[1] 발견적 해의 상대오차값이 이미 상당히 낮은 수준(평균 0.1671%)에 도달해 있어서, 발견적 해법의 상대오차 개선보다는 하한값의 개선이 상대적으로 많이 이루어진 것을 알 수 있다. 참고로, 문제유형별로 상대오차와 $n(B=L)$값을 비교해보면 공통적으로 "문제유형 2"에 대한 상대오차 및 $n(B=L)$값이 "문제유형 1"의 해당 값들보다 양호하게 나타나고 있음이 관찰된다(전재호[1]의 하한값을 사용한 경우 "문제유형 1"의 $n(B=L)$값이 평균 15.25로 "문제유형 2"의 해당값 16.5보다 작고, 본 연구의 하한값을 사용한 경우 역시 "문제유형 1"이 22.75로 "문제유형 2"의 25.75보다 작다). 이는 두 연구에서 제안된 발견적 해법이나 하한값 계산식이 "내부부품 생산 및 조립시간의 분산이 작은 경우"에 대해 보다 정확한 발견적해와 하한값을 제공함을 의미한다.

이전의 <표 3>에 나타난 바와 같이, 작업수가 40개 이상이면 본 연구의 분지한계알고리즘도 2시간 이내에 최적해를 산출하지 못하는 경우가 발생한다. 따라서, 작업수가 100개 이상으로 큰 경우에 대해서 발견적 해법 및 하한값의 성능비교평가를 다음과 같이 수행하였다.

첫째로, 발견적해법의 성능비교평가는 발견적 해와 하한값과의 상대오차($RE(\%)=100\times(Heu-LB)/LB$)를 사용하였다. 여기서, LB는 하한값을 의미하는데, 발견적해들의 성능을 공정하게 비교하기 위해, 본 연구에서 제안한 방식으로 계산한 값을 사용하였다. 두 번째로, 하한값의 성능비교평가는 동일한 문제조합내 30개의 문제들

<표 6> 발견적 해 및 하한값 성능비교(작업수 100이상)

	n	α	전재호[1] Heu&LB† ARE	MRE	n(H = L)	본연구Heu&LB‡ ARE	MRE	n(H = L)
문제유형 1	100	0.4	0.13	1.16	22	0.07	0.56	23
		0.6	0.17	1.22	20	0.13	0.95	23
		0.8	0.22	1.56	19	0.18	1.29	21
		1.0	0.23	2.49	1	0.18	2.49	22
	300	0.4	0.06	0.27	20	0.03	0.26	22
		0.6	0.04	0.45	25	0.03	0.28	27
		0.8	0.08	0.83	23	0.04	0.43	25
		1.0	0.10	0.92	0	0.08	0.92	23
	500	0.4	0.03	0.15	20	0.02	0.15	22
		0.6	0.02	0.23	21	0.02	0.23	24
		0.8	0.03	0.29	16	0.02	0.26	23
		1.0	0.04	0.25	3	0.02	0.13	19
	700	0.4	0.01	0.11	24	0.01	0.11	25
		0.6	0.02	0.22	23	0.01	0.08	23
		0.8	0.02	0.20	22	0.02	0.16	24
		1.0	0.04	0.28	6	0.03	0.28	19
	900	0.4	0.02	0.27	20	0.02	0.18	22
		0.6	0.02	0.15	23	0.02	0.15	25
		0.8	0.02	0.19	18	0.01	0.16	24
		1.0	0.03	0.23	7	0.03	0.23	20
유형 1 평균			0.07	0.57	16.65	0.05	0.47	22.80
문제유형 2	100	0.4	0.01	0.13	27	0.01	0.13	27
		0.6	0.02	0.27	22	0.02	0.27	26
		0.8	0.02	0.25	21	0.02	0.25	25
		1.0	0.00	0.03	3	0.00	0.03	29
	300	0.4	0.03	0.58	22	0.03	0.36	22
		0.6	0.02	0.18	19	0.02	0.18	22
		0.8	0.02	0.43	17	0.02	0.30	25
		1.0	0.02	0.20	0	0.02	0.19	26
	500	0.4	0.01	0.15	26	0.01	0.12	26
		0.6	0.00	0.03	26	0.00	0.03	27
		0.8	0.01	0.11	17	0.01	0.09	23
		1.0	0.03	0.36	4	0.03	0.36	20
	700	0.4	0.00	0.02	28	0.00	0.02	28
		0.6	0.00	0.02	26	0.00	0.02	26
		0.8	0.00	0.03	22	0.00	0.03	26
		1.0	0.01	0.15	6	0.01	0.12	25
	900	0.4	0.00	0.04	23	0.00	0.04	23
		0.6	0.00	0.03	24	0.00	0.03	26
		0.8	0.01	0.07	20	0.01	0.06	24
		1.0	0.01	0.11	9	0.01	0.11	21
유형 2 평균			0.01	0.16	18.10	0.01	0.14	24.85
전체 평균			0.04	0.37	17.38	0.03	0.30	23.83

주) n : 작업수.
α : 외주부품 조달시점 산포조절 계수.
ARE : 발견적 해의 하한값 대비 평균상대오차(%).
MRE : 발견적 해의 하한값 대비 최대상대오차(%).
n(H = L) : 30개 문제중 Heuristic결과값과 Lowerbound값이 동일한 문제수.
† ARE와 MRE계산에는 전재호[1]에서 제안된 Heuristic 결과값과 본 연구의 Lowerbound값 사용, n(H = L)계산에는 전재호[1]의 Heuristic과 Lowerbound값 사용.
‡ 본 연구에서 제안한 Heuristic과 Lowerbound사용.

중 하한값과 발견적해가 동일한 문제수($n(H=L)$를 사용하였다. 또, $n(H=L)$값은 발견적해와 하한값의 성능에 모두 관련된 척도이지만, <표 5> 및 <표 6>에 나타난 바와 같이 전재호[1] 대비 본 연구의 발견적해법 성능개선은 미미한(ARE의 전체평균값이 0.04%에서 0.03%로 감소하여 0.01% 개선됨) 반면 하한값의 성능개선은 상당한 수준으로($n(H=L)$의 전체평균값이 17.38개에서 23.83개로 6.45개가 증가되는 것으로 개선됨) 이루어졌음을 감안하여 하한값의 상대적 평가척도로 사용하였다. 성능비교 평가에 사용된 문제조합은 문제유형 두가지("문제유형 1", "문제유형 2"), 작업수 n이 5가지(100, 300, 500, 700, 900)인 경우에 대해 외주부품조달시점 산포계수 α값 4가지(0.4, 0.6, 0.8, 1.0)로 총 40가지이며, 각 문제조합별로 각각 30개씩의 문제를 임의로 생성하여, 총 1,200개의 문제를 사용하였다. 계산실험을 수행한 결과는 <표 6>에 나타나 있다.

<표 6>의 결과로부터 다음과 같은 두 가지 사실을 알 수 있다. 첫째로, 작업수가 큰 문제의 경우에도 전재호[1]의 발견적해보다 본 연구 발견적해의 상대오차(ARE, MRE 모두)가 작은 크기이기는 하지만 개선되었음을 알 수 있다. 개선의 정도는 "문제유형 1"이 "문제유형 2"보다 크게 나타나고 있는데(문제유형 1의 경우 ARE 평균값이 0.07%에서 0.05%로 개선되었고, 문제유형 2에 대한 ARE평균값은 표시소수점 이하자리에서 약간(0.012%에서 0.0108%로) 개선되었다), 이는 작업수가 50개 이하인 경우에도 일관되게 나타나는 현상으로, "문제유형 2"의 경우 전재호[1]의 발견적해도 이미 상당히 정확한 해를 제공하고 있다는 것을 의미한다. 두 번째로 $n(H=L)$값을 통해서 하한값의 성능향상(전재호[1]의 발견적해법과 하한값을 사용한 경우 $n(H=L)$값의 평균이 17.38개인 반면 본 연구의 경우 평균 23.83개로 개선됨), 특히 "외주부품조달시점 산포계수(α)"가 1.0에 가까워질수록 급격히 악화되었던 전재호[1] 하한값의 단점이 상당히 개선되고 있음을 알 수 있다. 또, 본 연구 휴리스틱과 하한값을 사용한 평균 $n(H=L)$값이 "문제유형 2"의 경우 24.85로 "문제유형 1"의 22.80보다 크게 나타남으로써 100개 이상의 작업을 갖는 큰 문제에 대해서도 "문제유형 2"에 대해 최적해에 보다 근사한 발견적해 및 하한값을 제공해주고 있음을 알 수 있다.

6. 결 론

본 연구에서는 외주를 고려한 2단계 조립일정계획에 대한 기존연구의 해법 성능을 개선하기 위한 목적으로 우월성질, 하한값, 발견적해법, 분지한계알고리즘 등을 새로이 제안하고 계산실험으로 성능향상을 확인하였다. 발견적해의 경우 기존연구의 해법이 이미 상당히 정확한 근사해를 산출하고 있으므로 작은 규모로 향상된 반면, 하한값은 상당한 수준의 개선이 이루어졌다. 특히 외주부품의 조달시점 산포정도가 내부부품 작업시간의 합과 가까워질수록 하한값의 성능이 급격히 악화되었던 기존연구의 단점을 상당부분 개선하였다. 분지한계알고리즘의 경우 기존연구에서는 작업수 30개 이상인 문제조합의 경우 최적해를 제공하지 못하는 경우가 있었으나 본 연구에서는 작업수 30개까지 제공하는 것으로 향상되었다. 특히, "문제유형 2"에 대해서는 작업수 50개까지 모든 조합에 대해 최적해를 산출하는 성능향상이 이루어졌다.

본 연구에서 발견적해와 하한값의 성능이 상당한 수준에 도달하였으므로, 분지한계알고리즘의 최적해 도출 가능 작업수 확대를 위해 "외주부품 조달시점의 분산정도가 큰 경우에 효과적으로 작동하는 우월성질"의 개발에 대한 추가연구가 일차적으로 요구된다. 나아가서, 조립에 소요되는 자체생산 및 외주부품의 개수가 여러 개인 경우로의 확장과, 다양한 일정계획 평가척도를 적용한 문제에 대해서도 추후연구가 필요하다.

참고문헌

[1] 전재호; "부품외주를 고려한 조립형 Flowshop 일정계획 문제 연구", 산업경영시스템학회지, 29(4) : 34-42, 2006.

[2] Bertrand, J. W. M. and Muntslag, D. R.; "Production control in engineer-to-order firms," *International Journal of Production Economics*, 30-31 : 3-22, 1993.

[3] Chen, Z. -L., Hall, N. G.; "Supply Chain Scheduling : Conflict and Cooperation in Assembly Systems," *Operations Research*, 55(6) : 1072-1089, 2007.

[4] Hariri, A. M. A., and Potts, C. N.; "A branch and bound algorithm for the two-stage assembly scheduling problem," *European Journal of Operational Research*, 103 : 547-556, 1997.

[5] Johnson, S. M.; "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, 1 : 61-68, 1954.

[6] Juhn, J., Sung, C. S., and Yoon, S. H.; "Analysis of heuristics for a two-stage assembly scheduling problem with component available time constraint," Submitted to *Operations Research Letters* for publication, 2006.

[7] Koulamas, C., and Kyparisis, G. J.; "The three-stage assembly flowshop scheduling problem," *Computers & Operations Research*, 28 : 689-704, 2001.

[8] Kovalyov, M. Y., Potts, C. N., and Strusevich, V. A.; "Batching decisions for assembly production systems," *European Journal of Operational Research*, 157 : 620-642, 2004.

[9] Lee, C. -Y., Cheng, T. C. E., and Lin, B. M. T.; "Minimizing the makespan in the 3-machine assembly-type flowshop scheduling problem," *Management Science*, 39 : 616-625, 1993.

[10] Lin, B. M. T., Cheng, T. C. E., and Chou, A. S. C.; "Scheduling in an assembly-type production chain with batch transfer," *OMEGA The International Journal of Management Science*, 35(2) : 143-151, 2007.

[11] Potts, C. N., Sevast'janov, S. V., Strusevich, V. A., Wassenhove, L. N. V., and Zwaneveld, C. M.; "The two-stage assembly scheduling problem : complexity and approximation," *Operations Research*, 43 : 346-355, 1995.

[12] Sun, X., Morizawa, K., and Nagasawa, H.; "Powerful heuristics to minimize makespan in fixed, 3-machine, assembly-type flowshop scheduling," *European Journal of Operational Research*, 146 : 498-516, 2003.

[13] Tozkapan, A., Kirca, O., and Chung, C. -S.; "A branch and bound algorithm to minimize the total weighted flowtime for the two-stage assembly scheduling problem," *Computers & Operations Research*, 30 : 309-320, 2003.