Journal of the Korean Institute
of Industrial Engineers
Vol. 19, No.3, September., 1993.

단일 기계의 일정계획 문제에 대한 지식 베이스 빔 탐색 기법†

A Knowledge-based Beam Search Method for a Single Machine Scheduling †

김성인* · 김선욱** · 양허용* · 김승권*

Seong-in Kim*, Sun-Uk Kim**, Heo-Yong Yang* and Sheung-kown Kim*

Abstract

A basic problem of sequencing a set of independent tasks at a single facility with the objective of minimizing total tardiness is considered. A variation of beam search, called knowledge-based beam search, has been studied which uses domain knowledge to reduce the problem size with an evaluation function to generate nodes probable to include the optimal solution. Its performance behavior is compared with some existing algorithms.

1. 서 론

본 논문에서는 고정된 수의 작업(job)을 처리하는 단일 기계가 총작업지연시간을 최소로 하기 위한 작업순서를 결정하는 문제를 다룬다. 일정계획(scheduling)분야에서 가장 기본적이고 단순한 문제의 하나인 이 문제는 정수계획법(integer programming)으로 정확히 나타낼 수 있고, 이에 대하여는 지난 30여년 동안 동적계획법(dynamic programming) 또는 분지 및 한계법(branch and bound technique)과 같은 전통적인 OR기법이나 경험적(heuristic)기법을 이용하여 해를 찾는 연구가 많이 진행되어 왔다[1],[2],[5]. 그러나 이 기법들의 컴퓨터 실행시간은 문제의 크기에 指數的으로 비례하므로 규모가 큰 문제에는 아직도 적용하기 어려운 상태이다.

이러한 문제점을 해결하기 위하여 최근 그 적용 분야를 광범위하게 넓혀가고 있는 인공지능(artificial intelligence)이나 그 지류의 하나인 전문가 시스템(expert system)의 기법을 일정계획 문제에 적용하려는 시도가 이루어지고 있다. 사실 인공지능 이론의 개발과 발전으로 일정계획 연구의 흐름 자체를 새롭게 바꾸어 놓고 있는 것이 현재의 실정이다.

본 논문에서는 이러한 흐름에 따라 일정계획 문제를 인공지능 기법을 이용하여 해결하는 방법을 제시하고자 한다. 인공지능 기법을 이용하여 해를 찾는 연구는 Ow와 Morton[5]에

† 이 논문은 한국과학재단의 목적기초연구의 연구비 지원으로 이루어졌음.

* 고려대학교 공과대학 산업공학과

** 단국대학교 공학대학 산업공학과

의하여 납기보다 일찍 만들어지는 제품에 대한 비용을 고려하는 문제에서 단일 기계의 일정계획 수립에 대하여 이루어져 있다. 본 논문은 총지연시간을 최소화하는 문제에 인공지능 기법을 적용하는 연구이다.

본 논문에서는 인공지능에서 사용되는 탐색(search)기법 중의 하나인 빔 탐색(beam search)을 이용한다. 빔 탐색은 1970년대 중반에 처음 개발되었고[5], 그동안 음성 인식, 영상 인식, 가스 터빈의 부품제작 등에 관한 일정계획 문제를 다룬 ISIS(Intelligent Scheduling and Information System) 등에 사용되어 왔다[5],[7]. 빔 탐색은 解공간이 광범위할 때 의사결정나무(decision tree)를 탐색하는데 쓰이는 기법으로 경험적 해법을 사용하여 적은 수의 해를 동시에 체계적으로 탐색함으로써 최소한의 노력으로 좋은 해를 찾을 수 있는 가능성을 최대화하려는 기법이다. 이 탐색 기법의 실행시간은 문제의 크기에 대해 거듭제곱의 형태로 증가하므로 지수적으로 증가하는 분지 및 한계 기법에 비해 실행시간을 크게 단축시킬 수 있다[5],[9].

빔 탐색은 부분적 해(partial solution)의 질(quality)을 평가하는데 사용되는 평가함수(evaluation function)에 따라 그 수행척도가 좌우되는데 이 평가함수는 일반적으로 경험적 기법에 의해 정의된다[5]. 본 논문에서도 적절한 평가함수를 개발하여 탐색할 노드를 결정한다. 아울러 현재 사용되고 있는 대부분의 빔 탐색 기법이 탐색하는 노드의 수, 즉 빔 幅(beamwidth)을 항상 일정하게 고정시키고 있지만, 개발되는 해법에서는 이를 유동적으로 변화시킨다. 또한 문제의 크기를 줄일 수 있는 내부지식(domain knowledge)을 사용한다. 내부지식을 사용하는 빔 탐색 기법이므로 이를 지식베이스 빔 탐색(knowledge-based beam search)기법이라고 부르기로 한다. 개발된 해법은 전통적인 OR기법들과 비교하여 보다 우수함을 보인다.

제 2 절에서는 단일 기계에서의 총작업지연시간 최소화 문제인 일정계획을 몇 가지 가정하에서 수식화하고, 기존의 해법들에 대하여 간략히 언급한다. 제 3 절에서는 인공지능에서 사용되는 탐색 기법의 하나인 빔 탐색 기법에 대하여 설명하고, 평가함수를 개발하여 빔 폭을 유동적으로 변화시키며, 내부지식을 사용하여 문제의 크기를 줄일 수 있는 지식 베이스 빔 탐색 기법을 개발한다. 제 4 절에서는 개발된 기법을 시뮬레이션에 의하여 기존의 알고리즘들과 비교 분석한다.

2. 문제의 정의 및 기존 해법

대부분의 단일 기계의 일정계획 연구에서와 같이 다음의 가정을 한다.

(1) 모든 작업은 서로 독립이며 시점 0에서 모두 가공이 가능하다. 즉, 모든 작업의 가공시간과 납기일은 작업간에 아무런 관련이 없으며 작업장에 도착하자 마자 순서계획에 따라 준비시간 없이 즉시 처리될 수 있다.

(2) 모든 작업의 준비시간은 독립이며 가공시간에 포함된 것으로 한다. 즉, 모든 작업에 준비시간은 가공시간이나 납기일과 관계가 없으며 각 작업간에도 영향을 미치지 않는다.

(3) 기계는 계속적으로 이용 가능하며 작업이 대기중인 경우에는 쉬지(idle) 않는다. 즉, 기계는 고장나거나 정지하지 않고 항상 이용 가능하다.

(4) 일단 작업이 시작되면 그 작업을 끝낼 때까지 가공이 계속된다. 즉, 한번 가공이 시작된 작업은 완전히 완료되기 전에는 가공을 중단하지 않는다(no preemption).

(5) 각 작업의 가공시간은 작업순서와는 독립적이다. 즉, 작업의 가공 순서에 따라 가공시간이 영향을 받거나 변하지 않는다.

(6) 각 작업에 대한 납기일과 가공시간은 고정되어 있으며 이를 미리 알고 있다. 즉, 확정적이며 확률적이 아니다.

(7) 모든 작업은 동일한 우선 순위를 가지고 있다. 즉, 어떤 작업도 다른 작업에 비해 먼저 가공되거나 늦게 가공되는 제약을 갖지 않는

다.

단일 기계 문제에서 다루어지는 문제의 하나인 지연시간 최소화 문제는 다음과 같이 수식으로 모델화할 수 있다. 작업 j가 완료되어야 하는 시간, 즉 납기일(due date)을 d_j, 실질적으로 완료되는 시간(completion time)을 c_j로 표시하면, 작업 j의 지연시간(tardiness time), 즉 완료시간과 납기일과의 차는

$$\max\{0, c_j - d_j\}$$

이 된다. 본 논문에서는 단일 기계에서 n개의 작업이 주어졌을 경우 이들의 합을 최소화하는 문제를 다룬다.

이 문제에서 이미 알려진 몇 가지 정리들이 있다. 이 정리들은 본 논문의 해법에서 지식 부분이 되므로 이를 지식이라 부르기로 한다.

지식 1. 만약 주어진 작업들 중에서

$$d_k \geq \sum_{j=1}^{n} t_j$$

를 만족하는 작업 k가 존재하면, 작업 k는 최적 작업순서에서 제일 마지막이 된다. 여기에서 t_j는 작업 j의 가공시간을 나타낸다.

지식 2. 최단 납기일(EDD: earliest due date)의 순으로 배열된 작업이 하나 이하의 지연 작업을 발생시키면, 그 순서는 총지연시간을 최소로 하는 순서이다.

지식 3. 모든 작업이 동일한 납기일을 가지면, 최단 가공시간(SPT: shortest processing time)의 순으로 작업을 배열함으로써 총지연시간을 최소로 할 수 있다.

지식 4. 임의의 작업 순서에 대해 임의의 한 작업이 제 시간에 완료될 수 없는 경우에는, SPT의 순으로 작업을 배열함으로써 총지연시간을 최소로 할 수 있다.

이 문제에 대하여 다음과 같은 해법들이 개발되어 있다.

Emmons 알고리즘. Emmons[4]는 작업의 일부분을 계산 과정 없이 직접 순서를 정하고 문제의 크기를 줄일 수 있는 특별한 조건을 밝혔다. 집합 A_j를 최적의 순서에서 작업 j뒤에 놓여지는 작업들의 집합, 집합 B_j를 최적의 순서에서 작업 j앞에 놓여지는 작업들의 집합이라 하고, 집합 A'이 집합 A의 여집합을 표시하며, $j \rho k$로 표시하는 것은 작업 j와 작업 k의 가공시간(processing time) t_j, t_k간에 $t_j < t_k$이거나 $t_j = t_k$이고 $d_j \leq d_k$인 관계를 나타낸다고 할 때, Emmons는 다음과 같은 사실들을 증명하였다.

지식 5. $j \rho k$이고 $d_j \leq \max\{d_k, t_k + \sum_{i \in B_k} t_i\}$ 이면, $j \in B_k$이다.

지식 6. $j \rho k$이고 $d_j > \max\{d_k, t_k + \sum_{i \in B_k} t_i\}$ 이며, $d_j + t_j \geq \sum_{i \in A_k} t_i$이면, $j \in A_k$이다.

지식 7. $j \rho k$이고 $d_k \geq \sum_{i \in A'_k} t_i$이면, $j \in A_k$ 이다.

Emmons 알고리즘이 다른 알고리즘들과 크게 다른 특징은 주어진 문제의 크기를 효율적으로 감소시킴으로써 문제 해결에 필요한 시간과 계산의 양을 줄일 수 있다는데 있다. Baker와 Martin[2]의 실험 결과에 의하면 Emmons 알고리즘은 문제의 크기를 평균 51.7%까지 감소시키며 작업의 수가 8개일 경우 문제 크기를 최대 64.1%까지 줄일 수 있다고 한다. 일단 문제의 크기가 줄어들고 나면 Emmons 알고리즘은 분지 및 열거 기법을 사용한다.

Srinivasan 알고리즘. Srinivasan[6]은 동적 계획법의 구조에 Emmons 알고리즘에 쓰인 개념들을 결합시킨 세 단계 혼합 알고리즘을 제시하였다.

단계 1에서는 $d_j \geq \sum_{i=1}^{n} t_i$ 인 모든 작업 j를 최적 순서의 맨 마지막에 배치한다. 단계 2에서는 부분적인 작업의 순서를 결정하고, 가능하

다면 최적 순서의 처음과 마지막에 위치할 작업들을 찾아내기 위해 Emmons의 지식 5와 지식 6을 이용한다. 단계 3에서는 단계 2에서 개발된 부분적 작업 순서를 이용할 수 있도록 수정된 동적 계획법 알고리즘을 이용하여 나머지 작업들을 최적의 순으로 배치한다.

Srinivasan은 다음과 같은 4가지 사실을 밝혀 동적 계획법 알고리즘을 수정함으로써 계산량을 줄일 수 있었다. 여기에서 $|A_j|$ 는 A_j에 속한 작업의 갯수를 나타낸다.

지식 8. 단계 k에서 $|A_j| \geq k, j \in J$이면, J는 동적 계획법에서 더 이상 고려할 필요가 없다.

지식 9. 단계 k에서 $|A_j| = k-1, j \in J$ 이면, 작업 j는 최적 순서의 맨앞에 온다.

지식 10. 단계 k에서 $|A_j| < k-1, j \in J$ 이면, $j \in A_j$인 모든 j를 포함한 J만이 순서계획에 고려된다.

지식 11. 단계 k에서 $|B_j| \geq n-k+1$인 작업 j는 순서의 맨앞에 배치하는 것을 고려하지 않아도 된다.

Srinivasan 알고리즘 역시 Emmons 알고리즘에서 쓰였던 지식들을 이용하여 문제의 크기를 줄이는 방법을 이용하고 있다. Baker와 Martin[2]의 실험에 의하면, 이 알고리즘은 Emmons 알고리즘에 비하여 문제 크기를 0.7 %정도 더 감소시킬 수 있다고 한다.

Srinivasan 알고리즘은 Emmons 알고리즘과는 달리 일단 문제의 크기를 줄이고 나면 수정된 동적 계획법을 이용하여 남은 문제를 해결한다. 이 방법은 Emmons 알고리즘에서 사용한 분지 및 열거 기법보다 효과적인 것으로 밝혀졌다. 이 알고리즘이 최적해를 찾는 데까지 걸리는 시간의 중앙값(median)은 문제 크기에 대해 비례적으로 증가한다. 그러나 평균 시간은 동적 계획법의 계산시간만큼 심하지는 않지만 여전히 지수적인 증가 형태를 나타내고 있다. Emmons 알고리즘이 납기일의 범위에 따라 계산 시간이 크게 달라지는 것과는 달리, 이 알고리즘은 어떠한 요인에도 계산 시간이 큰 영향을 받지 않는다는 장점을 지니고 있다.

Wilkerson-Irwin 알고리즘. 이 알고리즘은 인접한 두 작업(adjacent pairwise)을 교환하여 개선된 해를 찾아내는 경험적 방법에 기본을 두고 있다[2]. Wilkerson-Irwin 알고리즘은 두 단계로 이루어진 해법으로 단계 1에서는 주어진 모든 작업에 대하여 완전한 순서 계획이 얻어질 때까지 계획되거나 계획되지 않은 작업들의 리스트를 조정하기 위하여 의사결정 규칙을 적용한다. 단계 2에서는 현재까지 얻어진 해를 개선시키기 위해 가능한 작업들의 일부를 서로 교환하는 과정을 포함한다. 이 때 다음의 사실을 이용한다.

지식 12. 단계 1에서 얻어진 순서계획중의 임의의 한 점 t에서 임의의 두 작업 i와 j에 대하여 $t \in I_j$이고 $t \in I_i$인 t가 존재하지 않으면, 이 순서계획은 최적해이다. 이 때 I_i와 I_j는 각각 i와 j작업의 지연구간을 나타낸다.

Wilkerson-Irwin 알고리즘은 앞서 제시한 어떤 최적 알고리즘들보다 빠른 시간안에 해를 찾아낼 수 있다는 장점을 지니고 있지만 경험적 기법이므로 항상 최적해가 되지는 않는다. Baker와 Su[3], Baker와 Martin[2]에 의하면 Wilkerson-Irwin 알고리즘은 해를 찾는 데까지 걸리는 시간이 비례적으로 증가하는 반면 주어진 문제의 60%정도에 대해서만 최적해를 찾는 것으로 밝혀졌다. 이 알고리즘은 빠른 시간안에 해를 찾아낼 수 있는 반면 납기일의 범위가 작고 지연이 예상되는 작업의 수가 많을수록 효율이 떨어지는 단점을 가지고 있다. 또한 작업의 수가 많아짐에 따라 최적해에 가까운 해를 찾는 비율이 감소한다는 단점도 지니고 있다.

3. 지식 베이스 빔 탐색 기법

1) 빔 탐색(beam search)

빔 탐색 기법은 1970년대 중반에 한 인공지능 연구 단체에 의해 개발된 것으로 해 공간이

광범위할 때 의사결정나무를 탐색하는데 효과적으로 쓰일 수 있는 경험적 기법이다. 이 기법은 1976년 Lowerre에 의해 HARPY라고 하는 음성 인식 시스템에 최초로 사용되었다. 그 목적은 몇 개의 잠재적인 최적의사경로를 동시에 탐색함으로써 백트랙킹(backtracking)을 줄이고 해를 빨리 얻고자 하는 것이었다. 1978년에는 Rubin이 ARGOS라는 영상인식 시스템에 빔 탐색 기법을 사용하였다. 이러한 빔 탐색 기법의 원리는 일정 계획과 같은 복잡한 문제에서 해를 찾는데 유용하게 사용될 수 있는 1983년에는 Fox에 의해 ISIS라는 비교적 복잡한 일정계획 문제를 해결하는데 빔 탐색 기법이 사용되었다.

빔 탐색은 인공지능 분야에서 다루어지는 탐색기법 중 너비우선탐색(breadth-first search)을 보다 효율적으로 개선한 기법으로서 너비우선탐색과 최상우선탐색(best-first search)의 혼합된 형태라고 볼 수 있다. 즉, 빔 탐색은 의사결정나무의 각 단계에서 가장 좋은 노드만을 찾아내어 다음 단계로 분지시키는 최상우선탐색의 성질과 이 노드들을 각 단계에서 모두 탐색한 후 다음 단계로 넘어가는 너비우선탐색의 성질을 결합시킨 것이다.

빔 탐색 기법의 효율성은 탐색하는 노드의 수인 빔 폭을 어떻게 정하느냐에 따라 크게 달라질 수 있다. 즉, 빔 폭을 작게 하는 경우에는 그만큼 탐색해야 할 노드의 수가 감소하므로 해를 찾는 데까지 걸리는 시간이 줄어드는 반면 더 좋은 해를 찾아내지 못할 위험성이 커지게 된다. 반대로 빔 폭이 커지는 경우 최적해를 찾아낼 수 있는 가능성은 커지지만 그만큼 시간이 오래 걸리게 된다.

빔 탐색은 최적해를 포함한다고 판단되는 노드만이 탐색의 대상이 된다. 이 판단은 적절한 평가 함수(evaluation function)를 개발하여 수치로 나타낸다. 즉, 평가함수란 주어진 목적식을 반영하여 각 단계에서 각 노드가 얼마나 목적식을 만족시키는가의 정도를 수치로 나타내는 함수이다. 경험적 기법에 의해 만들어지는 평가함수는 언제나 동일한 형태를 취하는 것이 아니고 주어진 문제의 형태와 목적에 따라 달라지게 된다. 따라서 평가함수 자체는 문제 해결 과정에서 정의되거나 개발되어야 하며, 어떻게 정의하느냐에 따라 그 형태가 달라질 수 있다. 결국 평가함수의 값에 따라 빔 폭을 결정하게 되므로 결국 빔 탐색기법의 효율성은 평가함수가 얼마나 정확하게 노드들을 평가하느냐에 달려 있다.

평가함수가 각 노드의 상태를 정확히 평가할수록 좋은 노드를 기각할 가능성은 적어지므로 최적해를 찾아낼 수 있는 확률은 높아지고, 그렇지 못하면 좋은 노드를 기각할 가능성이 커지므로 그만큼 최적해를 찾아낼 수 있는 확률은 낮아지게 된다. 따라서 평가함수를 얼마나 올바르게 찾아내는가 하는 것이 빔 탐색 기법에 있어서 가장 중요한 문제가 된다. 평가함수가 모든 노드에 관하여 언제나 100%정확한 평가를 할 수 있다면 각 단계에서 하나의 노드만을 선택할 수 있으므로 최적해를 찾을 수 있는 확률은 1이 된다. 그러나 평가함수가 모든 노드의 상태를 정확히 반영하지 못한다면 각 단계에서 찾아야 하는 노드의 수는 커지게 되며 최적해를 찾는 시간도 더 길게 되며 최적해를 찾을 수 있는 가능성도 그만큼 줄어들게 된다.

2) 지식 베이스 빔 탐색(knowledge based beam searc)

빔 탐색에서는 일반적으로 빔 폭이 일정하게 고정되어 있다. 그러나 평가함수가 얼마나 정확한가에 따라 빔 폭도 달라질 수 있을 것이다. 평가함수가 노드들의 상태를 비교적 정확히 평가한다면 굳이 빔 폭을 크게 하지 않아도 좋은 해를 구할 수 있다. 이제 평가함수가 어떻게 개발, 정의되는 가를 설명한다.

먼저 전체 작업 중에서 지연되리라고 예상되는 작업, 즉 납기일보다 늦게 가공이 끝나리라고 예상되는 작업의 수가 적은 경우(전체 작업수의 반 이하)에는 납기일의 범위에 관계 없이 납기일이 빠른 순으로 작업을 배치하는 것이 최적해에 가까울 것이라고 예상할 수 있다. 따

라서 납기일이 빠른 작업의 경우에는 가중치를 크게 주어 가능한 한 빨리 가공되도록 하고, 납기일이 늦은 경우에는 가중치를 작게 주어 늦게 가공될 수 있도록 작업 j에 대한 가중치 w_j를 다음과 같이 부여한다.

$$w_j = e^{-x}.$$

이 식에서

$$x = d_j / \sum_{j=1}^{n} d_j$$

는 작업 전체의 납기일에 대한 작업 j의 납기일의 비율을 나타낸다. 우선적인 가공을 필요로 하는 작업에는 가중치를 크게 주어야 하므로 이를 위해서 지수형태를 취하였으며 모든 작업에 대한 가중치는 이 식을 통해 0과 1사이의 값을 갖게 된다.

그러나 전체 작업 중에서 지연되리라고 예상되는 작업의 수가 많아질 경우(전체 작업수의 반 이상)에는 납기일이 빠른 순으로 작업을 배치하는 것보다 여유시간이 빠른 순으로 작업을 배치하는 것이 최적해에 가까울 것이라고 예상할 수 있다. 따라서 여유시간이 작은 작업에는 가중치를 크게 주어 가능한 한 작업순서의 앞쪽에 오도록 하고 여유시간이 큰 작업에는 가중치를 작게 줌으로써 작업순서의 뒷 쪽에 오도록 작업 j에 대한 가중치 w_j를 다음과 같이 부여한다.

$$w_j = e^{-y}.$$

여기에서

$$y = s_j / \sum_{j=1}^{n} s_j$$

이며, 이 비율이 작을수록 작업 j의 여유시간은 작은 것을 나타내며 따라서 우선적인 가공이 필요하게 된다.

이렇게 각 작업에 대한 가중치가 결정되면 이를 이용하여 의사결정나무의 각 단계에서 주어진 노드들이 목적식에 얼마나 가까운 상태에 있는지를 나타내는 평가함수를 개발할 수 있다. S를 이미 순서가 정해진 작업들의 집합이라 하고 S′을 S의 여집합이라고 하면 임의의 한 단계에서 S다음에 놓일 작업 i에 대한 평가함수값 P_{Si}는 다음과 같은 식에 의해 얻어진다.

$$P_{Si} = \begin{cases} k_i w_i, & k_i \geq 0\text{일 때} \\ \dfrac{k_i}{w_i}, & k_i < 0\text{일 때.} \end{cases}$$

여기에서

$$k_i = \sum_{j \in S} [d_j - (t_i + t_j)]$$

이며, T_i는 이미 순서가 결정되거나 순서를 고려중인 작업들의 가공시간의 합, $T_i = \sum_{j \in S} t_j + t_i$와 같다.

윗 식에서 k_i는 총지연시간을 최소화해야 한다는 목적을 나타내고 있다. 즉, $d_j - (T_i + t_j)$는 임의의 단계에서 작업 j의 지연시간을 나타낸다. 그러므로 순서가 정해지지 않은 작업들의 지연 시간의 합은 앞으로 일정계획이 계속될 경우 그만큼의 지연이 되리라는 예측치로 생각할 수 있다. 따라서 남은 작업들의 지연시간의 합이 큰 경우에는 그 순서로부터 일정계획을 계속해 나가는 것을 방지하는 반면 지연시간의 합이 작은 경우에는 그 작업순서를 계속 고려해 나가도록 해야 할 것이다. 이를 위해 남은 작업들의 지연시간이 큰 경우에는 가중치를 작게 주고 그렇지 않은 경우에는 가중치를 크게 준다. 즉, k_i가 양수가 되는 경우에는 가중치를 곱해주며 k_i가 음수값인 경우에는 가중치를 나누어 준다.

위와 같은 평가함수를 써서 각 노드에 대한 평가함수의 값이 결정되고 나면 이를 바탕으로 해서 탐색해야 할 노드들의 수, 즉 빔 폭을 결정하게 된다. 그러나 이 빔 폭은 일정한 수로 고정되어 있는 것이 아니고 상황에 따라 커지거나 작아진다.

이제 이러한 평가함수의 값을 數 직선상에 작은 것으로부터 큰 것의 순으로 배열한 후 각

평가함수 간의 차이를 계산한다. 이렇게 각 평가함수 값에 대한 차이를 계산한 후 그에 대한 평균 A를 구한다. 이 값을 이용하여 다음과 같이 빔 폭을 결정하는 규칙을 사용하기로 한다.

규칙 1. 주어진 작업에 대하여 공통된 작업들이 없이 오직 하나씩의 작업 순서만 존재할 경우, 각 평가함수 값으로부터 다음 크기의 평가함수 값의 차이가 E의 범위내에 포함되면, 두 평가함수 값은 차이가 나지 않는다고 보고 함께 다음 단계로 분지시킨다.

이 규칙에 대한 타당성은 다음과 같다. 평균적으로 평가함수의 값들이 E만큼씩의 차이를 가지고 있으므로 두 노드간의 평가함수 값이 그보다 작은 차이를 나타내면 이것은 차이가 적은 것이라고 볼 수 있으며 따라서 두 노드가 서로 다른 것이라고 보지 않는다. 그와 반대로 만약 두 노드간의 차이가 E보다 크다면 이것은 분명히 차이가 나는 것이라고 볼 수 있으며 따라서 두 노드는 서로 다른 것이라고 본다.

단계 1에서 모두 다음 단계 2로 분지된 노드들의 작업들로만 이루어진 작업순서로부터 최적해가 나올 가능성이 크다. 따라서 이와 같은 지식을 이용해 다음의 규칙 2를 만들어낼 수 있다.

규칙 2. 단계 2이상에서는 전 단계에서 넘어온 노드의 수가 2이상일 때 이들의 조합으로 이루어진 작업순서만을 다음 단계로 분지시킨다. 만약 전단계에서 넘어온 작업의 수가 하나뿐일 경우에는 규칙 1을 이용하여 빔 폭을 결정한다.

의사결정의 각 단계마다 위와 같이 규칙 1과 규칙 2를 이용하여 빔 폭을 결정하고 나면 그 노드들만 탐색을 계속하고 나머지 노드들은 탐색 대상에서 제외시키는 방법으로 주어진 모든 작업들에 대해 완전한 순서계획이 이루어질 때까지 탐색을 계속해 나간다. 이와 같은 과정을 반복하는데 있어서 최적해를 구하는데 필요한 계산의 양을 줄이고 탐색하는 노드의 수를 감소시키기 위하여 다음과 같은 규칙을 알고리즘에 포함시킨다.

규칙 3(tie-break규칙). 주어진 작업의 개수가 n이라고 할 때, $1<k<n$인 임의의 단계 k에서 동일한 작업들로 구성된 작업순서가 여러가지 있을 경우, 그 단계까지의 각 작업 순서에 대한 지연시간을 계산하여 가장 작은 값을 갖는 순서만을 다음 단계로 분지시키고 나머지 순서들은 더 이상 고려하지 않는다. 만약 모든 작업순서에 대한 지연시간이 0이면 다음 단계에서 모든 작업순서를 고려한다.

지식 베이스 빔 탐색 알고리즘은 두 단계(phase)로 이루어진다. 단계 1에서는 주어진 작업에 대해 지식들을 이용하여 부분적인 작업순서를 결정하고, 단계 2에서는 단계 1에서 순서가 정해지지 않은 나머지 작업들의 순서를 빔 탐색을 이용하여 결정한다. 결국 이 알고리즘은 지식을 사용하여 문제 크기를 줄인다는 점에서 Srinivasan 알고리즘, Emmons 알고리즘과 같다. Emmons 알고리즘은 분지 및 열거기법을 사용하고, Srinivasan 알고리즘은 동적계획법을 이용하는 반면, 이 알고리즘은 빔 탐색 기법을 이용하여 문제를 해결하는 점이 다르다. 본 논문에서는 단계 1을 지식 부분으로, 단계 2를 빔 탐색 부분으로 구별한다.

지식 베이스 빔 탐색 알고리즘

단계 1(지식 사용). 주어진 작업에 대하여 지식 1~지식 12를 적용시켜 부분적인 작업순서를 결정한다. 만약 이 지식들에 의해 주어진 모든 작업에 대한 작업순서가 결정되면 그 때의 작업순서가 최적순서가 되고 알고리즘을 중단한다. 만약 이 지식들에 의해 순서가 정해지지 않은 작업들이 존재하면 그 작업들에 대해 단계 2를 적용하며 나머지 작업에 대한 작업순서를 완성한다.

단계 2(빔 탐색). 다음과 같은 빔 탐색 절차를 적용한다.

절차 1(초기화). 단계 1에서 작업순서가 결정된 작업들의 가공시간의 합을 남은 작업들

의 납기일로부터 빼준다.

절차 2. 단계 1에서 순서가 결정되지 않은 작업들에 대하여 평가함수 값 P_{si}를 구한다. 규칙 1과 규칙 2에 따라 다음 단계로 분지시킬 노드들을 결정한다.

절차 3. 규칙 3을 적용한다. 주어진 모든 작업에 대한 순서 계획이 끝날 때까지 절차 2, 절차 3을 반복한다. 모든 작업에 대한 완전한 순서 계획이 이루어지면 절차 4로 간다.

절차 4. 앞에서 찾아낸 가능한 해들에 대하여 총 지연 시간을 구한다. 이들 중 최소의 값을 갖는 해를 최종해로 선택한다.

3) 예제

다음과 같은 8개의 작업에 대한 최적의 작업순서를 결정하는 문제의 해를 구하여 보기로 한다.

작 업	1	2	3	4	5	6	7	8
가공시간	61	52	126	101	110	111	112	82
납 기 일	521	459	489	469	541	460	460	502

단계 1. 앞에서 정의한 지식들을 이용하면 다음의 표에서 보는 바와 같이 최적 순서에서 i작업뒤에 오는 작업의 갯수, A_i는 2번 작업이 최적 순서의 맨앞에 놓이고 7번, 6번, 4번 작업의 순으로 작업순서가 결정됨을 알 수 있다. 마찬가지로 최적 순서에서 i작업의 앞에 놓이는 작업의 갯수, B_i의 경우도 2번 작업이 최적 순서의 맨 앞에 놓이고 다음으로 7번, 6번, 4번 작업이 놓이게 된다. 따라서 단계 1에서 결정되는 작업 순서는 2−7−6−4−?−?−?−?가 된다.

i/j	2	1	8	4	5	6	7	3	A_i
2	0	1	1	1	1	1	1	1	7
1	0	0	0	0	1	0	0	0	1
8	0	0	0	0	1	0	0	0	1
4	0	1	1	0	1	0	0	1	4
5	0	0	0	0	0	0	0	0	0
6	0	1	1	1	1	0	0	1	5
7	0	1	1	1	1	1	0	1	6
3	0	0	0	0	0	0	0	0	0
B_i	0	4	4	3	6	2	1	4	

단계 1에서 1,3,5,8 네 개의 작업이 단계 2로 넘어왔으므로 이 작업에 대해서만 빔 탐색 기법을 적용하면 된다.

단계 2. 절차 1. 단계 1에서 순서가 정해진 작업들의 가공시간의 합을 나머지 작업들의 납기일에서 빼주어 문제를 다음과 같이 수정한다.

작 업	1	3	5	8
가공시간	61	126	110	82
납 기 일	145	113	165	126

작 업	1	3	5	8
여유시간	84	-13	55	44
가 중 치	0.61	1.08	0.72	0.77

절차 2. 각 작업에 대한 평가함수 값을 계산한다.

$P_1=-159.0\dot{2}$, $P_3=-180.56$, $P_5=-298.61$, $P_8=-155.84$.

이 평가함수 값들간의 범위를 구하여 보면 다음과 같다. 여기서 평가함수 값의 범위에 대한 평균은 E=35.69이고, $P_8-P_1=3.18<E$, $P_1-P_3<E$, $P_3-P_5>E$이므로 단계 1에서 S={8} 또는 S={1} 또는 S={3}이 된다.

절차 2. 단계 2에서는 규칙 2에 의해 여섯개의 작업순서가 생기게 된다. 즉 S={1, 3} 또는 S={3, 1}, S={1, 8} 또는 S={8, 1}, S={3, 8} 또는 S={8, 3}과 같이 여섯개의 작업순서가 생긴다.

절차 3. 규칙 3에 의해 S={8, 1}, S={3, 1}, S={8, 3} 세 개의 작업순서만이 다음 단계로 분지된다.

절차 2. 단계 2에서 넘어온 3개의 작업순서에 대해 단계 3에서는 다음과 같이 6개의 작업순서가 만들어진다.

S={8, 1, 3}, S={8, 1, 5}, S={3, 1, 5},
S={3, 1, 8}, S={8, 3, 1}, S={8, 3, 5}.

그런데 이 여섯개의 작업순서에 대해 1,3,8번 작업만으로 이루어진 작업순서가 3개이므로 단계 3에서는 이 작업순서들만을 고려한다.

절차 3. 이 세 개의 작업순서에 규칙 3을 적용하면 S={8, 1, 3}일 때의 총 지연시간이 최소가 되므로 단계 3에서의 작업순서는 S={8, 1, 3}이 된다.

절차 4. 따라서 최종적인 작업순서는 S={8, 1, 3, 5}이고 이 때 총 작업 지연시간은 370이 된다.

4. 해법의 효율

본 논문에서 개발한 지식 베이스 빔 탐색 알고리즘의 효율을 기존의 알고리즘들 중 경험적 알고리즘으로서는 유일한 Wilkerson-Irwin 알고리즘과 최적알고리즘 중 가장 효율적이라고 할 수 있는 Srinivasan 알고리즘과 비교한다.

1) 문제의 생성

모의실험은 문제 구조의 특성들이 알고리즘의 수행 척도에 어떠한 영향을 미치는지 분석할 수 있도록 설계된다. 이러한 특성들은 다음과 같다.

지연 요소(trardiness factor). 어떤 문제에 대한 지연 요소는 주어진 작업들을 임의의 순으로 배열했을 때 지연되리라고 예상되는 작업들의 비율을 대략적으로 나타낸 것이다. μ_p를 주어진 작업들의 평균 가공 시간이라고 하면, 보통 kμ_p시간까지 k개의 작업이 처리될 수 있다. 만약 μ_d를 평균 납기일이라고 하면 μ_d=kμ_p일 때 k개의 작업은 주어진 납기안에 가공을 끝마칠 수 있다. 따라서 t를 지연 요소라고 정의하면

$$\mu_d = k(1-t)\ \mu_p$$

를 만족하게 된다.

본 논문에서는 지연 요소가 작은 경우와 큰 경우로 t=0.2와 t=0.6를 살펴본다. 가공 시간은 평균이 100이고 표준 편차가 25인 정규 분포로부터 생성된다.

납기일의 범위(due-date range). Wilkerson과 Irwin[8]은 그들의 알고리즘이 주어진 작업들의 납기일의 범위와 관계가 있음을 밝혀내었다. 즉, 납기일의 범위가 가공 시간의 합의 95%보다 크면 아주 효율적으로 사용될 수 있지만, 85%이하가 되면 최적해를 찾을 수 있는 가능성은 그보다 작아진다는 것이다. 따라서 이 알고리즘 뿐만 아니라 다른 알고리즘들도 이러한 납기일의 범위에 따라 알고리즘의 효율이 영향을 받는지 알아볼 필요가 있다.

납기일은 평균이 μ_d=k(1−t)μ_p이고 범위가 (a, b)인 구형 분포(uniform distribution)로부터 생성된다. 이제 납기일의 범위와 총 평균가공시간간의 비를

$$R = \frac{(b-a)}{k\mu_p}$$

로 표시한다. 납기일의 범위가 큰 경우와 작은 경우로 R=0.20과 R=0.95를 살펴본다.

따라서 다음의 네 가지 경우를 실험한다.

(i) 지연요소가 작고(t=0.2) 납기일의 범위가 좁은(R=0.20) 경우

(ii) 지연요소가 작고(t=0.2) 납기일의 범위가 넓은(R=0.95) 경우

(iii) 지연요소가 크고(t=0.6) 납기일의 범위가 좁은(R=0.2) 경우

(iv) 지연요소가 크고(t=0.6) 납기일의 범위가 넓은(R=0.95) 경우.

이 네 가지 경우에 대하여 작업의 수를 각각 20개, 50개, 80개, 100개로 변화시켜 가며 각 알고리즘을 분석한다. 작업의 수가 100개 정도에 이르면 실제 상황에서 고려되는 문제의 크기와 거의 동일하므로 최대 문제의 크기는 100개로 한다. 또 각각의 문제 크기에 대해서는 10개 씩의 서로 다른 문제를 실험하였다. 따라서 본 연구에서는 모두 160개의 문제를 서로 다른 세 가지 알고리즘에 대하여 실험한다.

2) 실험 결과 및 분석

본 실험에 사용된 알고리즘은 C언어로 프로그래밍하였으며 IBM PC 386 기종을 이용하여 실행하였다. 160개의 문제를 세 가지 알고리즘에 적용한 결과가 표 1부터 표4까지에 종합되

어 있다. 여기에서 각 표에 주어진 알고리즘의 실행시간은 문제를 해결하는데 필요한 순수한 런 타임만을 1/100초까지 측정한 10개 실행시간들의 평균이다. 또한 최적 시간으로부터 5%를 초과하는 범위내에 포함되는 지연시간을 갖는 문제의 수를 문제갯수에 나타내었고 괄호안에 10%를 초과하는 문제의 갯수를 나타내었다.

단계 2의 효율. 표 1부터 표 4까지의 결과에 의하면 예상했던 대로 최적 알고리즘에 비해 두 가지 경험적 알고리즘이 해를 찾는 데까지 걸리는 시간이 훨씬 작음을 알 수 있다. 최적 알고리즘의 실행시간은 문제 크기가 커짐에 따라 지수의 거듭제곱 형태로 증가하지만 두 경험적 알고리즘의 실행시간은 문제 특성이 t=0.6, R=0.2일 때의 Wilkerson-Irwin 알고리즘을 제외하고는 문제 크기에 선형으로 비례하여 증가함을 알 수 있다. t=0.6, R=0.2일 때의 Wilkerson-Irwin 알고리즘은 문제 크기가 커짐에 따라 실행시간도 급격히 증가하는 형태를 취한다.

표 1. t=0.2, R=0.2일 때의 각 알고리즘의 실행 결과

문제크기	최적 알고리즘 (Srinivasan기법)	경험적 알고리즘			
		Wilkerson-Irwin		지식베이스빔탐색	
	시 간	시간	문제갯수	시간	문제갯수
20	2.16	0.24	10(10)	0.19	10(10)
50	824.44	0.65	8(9)	0.55	8(8)
80	4340.13	1.19	6(9)	0.97	7(9)
100	12347.18	1.71	4(8)	1.24	5(8)

표 2. t=0.2, R=0.95일 때의 각 알고리즘의 실행 결과

문제크기	최적 알고리즘 (Srinivasan기법)	경험적 알고리즘			
		Wilkerson-Irwin		지식베이스빔탐색	
	시 간	시간	문제갯수	시간	문제갯수
20	2.17	0.25	10(10)	0.18	10(10)
50	831.02	0.60	9(9)	0.54	10(10)
80	4349.17	0.93	10(10)	0.91	10(10)
100	12257.38	1.16	10(10)	1.03	10(10)

표 3. t=0.6, R=0.2일 때의 각 알고리즘의 실행 결과

문제크기	최적 알고리즘 (Srinivasan기법)	경험적 알고리즘			
		Wilkerson-Irwin		지식베이스빔탐색	
	시 간	시간	문제갯수	시간	문제갯수
20	2.15	0.50	5(8)	0.20	4(8)
50	828.62	6.71	3(4)	0.60	10(10)
80	4358.98	104.17	2(5)	1.03	7(9)
100	12317.92	1208.7	2(3)	1.32	6(10)

표 4. t=0.6, R=0.95일 때의 각 알고리즘의 실행 결과

문제크기	최적 알고리즘 (Srinivasan기법)	경험적 알고리즘			
		Wilkerson-Irwin		지식베이스빔탐색	
	시 간	시간	문제갯수	시간	문제갯수
20	2.17	0.34	5(9)	0.22	10(10)
50	824.53	1.89	5(8)	0.61	10(10)
80	4341.01	2.94	3(7)	0.99	5(10)
100	12307.08	12.51	2(9)	1.22	5(10)

또한 표 1부터 표 4까지의 결과로부터 경험적 알고리즘의 실행시간은 문제가 가지는 특성과 관계가 있음을 알 수 있다. 두 알고리즘의 실행시간은 문제 특성이 t=0.2, R=0.95인 경우 다른 특성을 갖는 문제에 비해 가장 빠른 것으로 나타났으며 t=0.6, R=0.2인 경우 가장 느린 것으로 나타났다.

두 가지 경험적 알고리즘을 비교해 볼 때 두 알고리즘 간의 실행시간에는 큰 차이가 없지만 항상 지식 베이스 빔 탐색 알고리즘의 실행시간이 Wilkerson-Irwin 알고리즘에 비해 빠르다는 것을 알 수 있다. 지연요소가 크고 납기일의 범위가 작은 경우 지식 베이스 빔 탐색 알고리즘의 실행시간 역시 다른 특성을 갖는 문제에 비해서는 해를 찾는 시간이 오래 걸리지만 Wilkerson-Irwin 알고리즘의 실행시간에 비하면 상당히 빠르게 나타난다.

지연요소가 크고 납기일의 범위가 작은 문제에 대해 Wilkerson-Irwin 알고리즘의

경우 실행시간에 매우 큰 차이가 있다는 것을 알 수 있다. 그 이유는 이 알고리즘이 인접한 두 작업을 서로 자리를 바꾸어가며 최적해를 찾아내는 것이므로 가능한 최소의 비교 회수와 최대의 비교 회수를 모두 고려하기 때문인 것으로 여겨진다. 즉, 인접한 두 작업을 비교하는 횟수가 적은 경우에는 실행시간이 짧게 나타나고 비교하는 횟수가 많은 경우에는 실행시간이 길게 나타나는 것으로 여겨진다. 그러나 지식 베이스 빔 탐색 알고리즘의 경우 빔 폭이 예외적으로 커지지 않는 한 언제나 일정한 실행시간을 가질 수 있으므로 주어진 문제가 어떤 특성을 갖느냐에 상관없이 실행시간이 거의 변함이 없는 것으로 생각할 수 있다.

따라서 실행시간에 의해 알고리즘의 효율을 따지는 경우 주어진 문제가 어떠한 특성을 갖느냐에 상관없이 실행시간의 평균과 분산이 적은 지식 베이스 빔 탐색 알고리즘이 Wilkerson-Irwin 알고리즘간에 비해 언제나 효율적이라고 할 수 있다.

해의 正確度 역시 문제의 특성과 관계가 있는 것으로 나타났다. 지식 베이스 빔 탐색 알고리즘은 모든 문제에 대해 비교적 안정된 수의 최적해 또는 그에 근사한 해를 찾아내며 특히 t값에 관계 없이 R=0.95인 경우 최적해에 가까운 해를 찾아내는 것으로 나타났다. 반면에 Wilkerson-Irwin 알고리즘은 문제의 특성이 t=0.2, R=0.95인 경우에 한해서만 최적해에 근사한 해를 효율적으로 찾아낸다는 것을 알 수 있다. 다시 말해서 Wilkerson-Irwin 알고리즘은 지연요소가 커지고 납기일의 범위가 작아질 경우 최적해에 근사한 해를 그리 많이 찾아내지 못하므로 이러한 특성을 갖는 문제에 대해서는 그리 효율적이지 못하다고 할 수 있다. 실제 실행 결과를 보면 t=0.6, R=0.2인 경우 최적해에 근사한 해를 찾아내는 갯수가 다른 특성을 갖는 문제의 경우보다 훨씬 적은 것으로 나타났다.

알고리즘의 효율에 영향을 미치는 또 하나의 요소는 문제의 크기이다. Wilkerson-Irwin 알고리즘은 작업의 갯수가 증가함에 따라 최적해에 근사한 해를 찾는 비율이 감소하는 것으로 나타났다. 즉, 작업의 수가 적은 경우에는 최적해 또는 이에 근사한 해를 찾아내는 비율이 높은 반면 작업의 수가 많은 경우에는 이 알고리즘에 의한 해가 최적해로부터 상당해 차이가 난다는 것을 알 수 있다. 지식 베이스 빔 탐색 알고리즘 역시 문제의 크기가 커짐에 따라 최적해에 근사한 해를 찾는 비율이 감소하지만 Wilkerson-Irwin 알고리즘에 비해 감소폭이 적으며 알고리즘의 효율도 크게 떨어지지 않는다. 이러한 결과로 미루어 작업의 수가 커질수록 지식 베이스 빔 탐색 알고리즘의 효율은 높아지지만 Wilkerson-Irwin 알고리즘의 효율은 더욱 감소하게 될 것이다.

지금까지 분석한 결과를 종합하여 보면 결국 실행시간과 해의 정확도 측면 모두에서 지식 베이스 빔 탐색 알고리즘이 Wilkerson-Irwin 알고리즘에 비해 우수하다는 결론을 내릴 수 있다. 특히 지연요소가 크고 납기일의 범위가 작은 경우와 문제의 크기가 커질수록 지식 베이스 빔 탐색 알고리즘이 Wilkerson-Irwin 알고리즘에 대해 상당히 우수하다고 말할 수 있다.

단계 1의 효율. 이제까지는 지식 베이스 빔 탐색 기법 중 단계 2의 효율성만을 분석하였다. 이제 단계 1, 즉 문제의 크기를 효과적으로 줄이고 해의 정확도를 높일 수 있는 지식을 적용시킬 때의 효율을 분석하여 보기로 한다. 앞의 모의실험에서 사용된 문제들 중 t=0.6, R=0.2인 경우에 대하여 단계 1과 단계 2를 모두 적용시킨 결과를 표 5에 나타내었다.

단계 1이 적용되면 빔 탐색으로 처리해야 할 작업의 수가 줄어들기 때문에 처음부터 똑같은 크기의 문제에 대해 빔 탐색을 적용하는 것보다 시간이 단축될 수 있을 뿐만 아니라, 단계 1에서 부분적으로 결정되는 작업의 순서는 최적 순서이므로 그만큼 해의 정확도는 높아진다.

표 5의 결과를 분석하여 보면 지식 베이스 빔 탐색 기법 중 단계 2만을 적용시켰을 때 보다 단계 1을 포함한 경우가 더욱 최적해에 근사한 해를 찾아내며 그 시간 또한 단계 2만을 적용시킨 경우보다 빠르다는 것을 알 수 있다. 단계 1을 포함시킨 경우 해를 찾는 데까지 걸리는 시간은 평균 31%단축되었으며 최적해의 값으로부터 5%내에 포함되는 해의 수도 40문제 중 10개가 증가했고 모든 해가 최적해로부터 10%를 넘지 않는 범위내에 포함되는 것으로 나타났다. 따라서 단계 1을 포함시킬 경우 지식 베이스 빔 탐색 알고리즘은 앞서 제시된 모의실험 문제들에 대해 Wilkerson-Irwin 알고리즘보다 훨씬 좋은 결과를 얻게 될 것이다. 이것은 하나의 문제 해결 방법에 새로운 지식을 추가시킴으로써 해를 개선시키고 실행시간을 단축시키는 것이므로 이러한 결과로부터 지식 베이스 빔 탐색 기법은 문제 해결에 도움이 되는 지식을 더욱 많이 포함시킬수록 단순한 빔 탐색 기법보다 좋은 해를 찾아낼 것이라고 예상할 수 있다.

표 5. t=0.6, R=0.2일 때의 지식 베이스 빔 탐색의 효율 비교

문제크기	단계2만 적용시켰을경우		단계1만 적용시켰을경우	
	시 간	문제갯수	시 간	문제갯수
20	0.20	4(8)	0.16	10(10)
50	0.60	10(10)	0.45	10(10)
80	1.03	7(9)	0.82	9(9)
100	1.32	6(10)	0.94	8(10)

5. 결 론

인공지능 분야에서 쓰이는 탐색 기법 중의 하나인 빔 탐색 기법을 변형하여 일정계획의 가장 기본적인 분야 중 하나인 총지연시간을 최소화하는 단일기계에 대한 순서계획 문제를 해결하는 경험적 알고리즘을 개발하였다.

대부분의 기존 빔 탐색에서 매 단계마다 일정하게 고정되어 있는 노드의 수(빔 폭)를 상황에 따라 변화시키고 적절한 빔 폭을 결정하기 위하여 필요의 정도를 적절히 반영하는 평가함수를 개발하였다. 또한 문제의 크기를 효과적으로 줄이는데 이용되는 내부지식은 기존 연구에서 밝혀진 여러가지 사실 및 정리들을 사용하였다.

모의실험을 통하여 분석한 결과, 본 연구에서 개발한 지식 베이스 빔 탐색기법은 문제의 특성에 상관없이 현재까지 가장 우수하다고 알려진 Wilkerson-Irwin 알고리즘에 비해 해를 찾는 시간과 정확도 측면 모두에서 우수한 것으로 나타났다. 또한 개발된 해법에서 내부 지식이 시간과 정확도에 기여하는 정도를 알아보았다.

앞으로 평가함수를 보다 개선하고 내부지식을 보다 많이 포함시킴으로써 본 알고리즘의 효율을 높일 수 있을 것이다. 또한 문제의 범위를 단일 기계에 대한 총작업지연시간을 최소화하는 것으로 국한시켰지만, 이를 FMS와 같이 보다 복잡하고 다양한 목적식을 가지는 일반적인 일정계획 분야로 점진적으로 확대시켜 적용하는 연구가 필요할 것이다.

참 고 문 헌

1. Baker, K.R., *Introduction to Sequencing and Scheduling*, John Wiley & Sons, New York, New York, 1974.
2. Baker, K.R. and Martin, J.B., "An Experimental Comparison of Solution Algorithms for the Single-Machine Tardiness Problem," *Naval Research Logistics Quarterly*, 21(187－199), 1974.
3. Baker, K.R. and Su, Z.S., "Sequencing with Due-Dates and Early Start Times to Minimize Maximum Tardiness," *Naval Research Logistics Quarterly*, 21(171－176), 1974.
4. Emmons, H., "One Machine Sequencing Problem to Minimize Certain Functions of Job Tardiness," *Operations Research*, 17(701－715), 1969.

5. Ow, P.S. and Morton, T.E., "Filtered Beam Search in Scheduling," *International Journal of Production Research*, 26(35-62), 1988.
6. Srinivasan, V., "A Hybrid Algorithm for the One Machine Sequencing Problem to Minimize Total Tardiness," *Naval Research Logistics Quarterly*, 18(317-327), 1971.
7. Waterman, D.A., *A Guide to Expert Systems*, Addison-Wesley, Reading, Massachusttes, 1986.
8. Wilkerson, L.J. and Irwin, J.D., "An Improved Method for Scheduling Independent Tasks," *AIIE Transactions*, 3(239-245), 1971.
9. Winston, P.H., *Artificial Intelligence*, Addison-Wesley, Reading, Massachusttes, 1984.