개선된 Randomizing 알고리즘을 이용한 Job Shop 일정계획에 관한 연구

- A Study on the Job Shop Scheduling Using Improved Randomizing Algorithm -

이 화 기 *
Lee Hwa Ki
김 민 석 *
Kim Min Suk
이 승 우 **
Lee Seung Woo

Abstract

The objective of this paper is to develop the efficient heuristic method for solving the minimum makespan problem of the job shop scheduling.

The proposed heuristic method is based on a constraint satisfaction problem technique and a improved randomizing search algorithm. In this paper, ILOG programming libraries are used to embody the job shop model, and a constraint satisfaction problem technique is developed for this model to generate the initial solution. Then, a improved randomizing search algorithm is employed to overcome the increased search time of constrained satisfaction problem technique on the increased problem size and to find a improved solution.

Computational experiments on well known MT and LA problem instances show that this approach yields better results than the other procedures.

Keyword : Job Shop Scheduling, Constrained Satisfaction Problem

* 인하대학교 산업공학과
** 한국기계연구원

1. 서 론

Job Shop 일정계획 문제는 해결하기에 가장 어려운 NP-hard 조합최적화 문제들 중에 하나이다. Job Shop 일정계획문제는 문제가 작은 경우에 최적해를 구할 수 있지만 문제의 크기가 커지면 최적해를 구하기가 어렵다. 이러한 문제들을 해결하기 위해선 최적해에 가까운 좋은 근사해를 합리적이고 적절한 계산 시간 내에 탐색 할 수 있는 발견적 탐색 방법이 요구된다.

본 연구의 목적은 문제의 크기가 커짐으로서 해결이 어려운 Job Shop 일정계획 문제를 인공지능기법의 일종인 제약만족기법을 기반으로한 발견적 해법으로 해결하고자 하는 것이다. 본 연구의 발견적 해법은 제약만족기법을 적용할 수 있는 ILOG Solver 및 ILOG Scheduler libraries를 이용하여 문제에 대한 모형을 구현하고 문제가 크고 복잡하여 계산 시간이 증가 할 경우 제약만족기법의 탐색 만으로는 해결이 불가능 한 경우, 이를 극복하기 위해 개선된 Randomizing 탐색 알고리즘을 적용하여 탐색의 한계를 극복하고 최대한 최적해에 가까운 근사해를 탐색 할 수 있는 기법을 제안하였다.

2. 이론적 배경

2.1 제약만족 기법

제약 만족 기법은 여러 가지 제약조건을 만족하는 해를 구하는 것으로 제약 만족기법의 구성은 변수 $V=v_1, v_2, v_3 \cdots v_n$도메인 $D=d_1, d_2, d_3 \cdots d_n$그리고 변수들 간의 제약 조건 $C=c_1, c_2, c_3 \cdots c_m$로 이루어진다. 제약 만족 기법은 변수 v_n에 대하여 각 변수에 들어갈 값의 도메인이 d_n라고 하면 지수의 비율인 $O(v_n!)^{dn}$의 복잡도를 갖기 때문에 변수나 도메인이 큰 경우에는 많은 시간이 소요된다. 이러한 문제에서 각 변수를 전향검사(Forward Checking)를 이용하여 제약조건의 일치성(Consistency)을 검사하고 이에 따라 도메인 여과(Domain Reduction)기법을 적용하여 백트랙킹(Backtraking)을 할 경우에는 전체 탐색 공간의 크기를 줄일 수 있다.

제약 만족 기법에서 불필요한 도메인을 제거하는 도메인 여과기법은 제약조건의 일치성 검사로 이루어진다. 일치성 검사는 제약조건의 불일치성(Inconsistency)을 일으키는 부분을 찾아 불일치를 유발하는 도메인을 제거함으로서 탐색공간, 도메인의 일치성을 유지하며 이러한 일치성 검사를 위해서 일반적으로 제약조건 네트워크를 구성하게 된다. 제약 조건 네트워크는 유향성 네트워크로 표현이 되며 제약조건이 적용된 노드(Node)에서 가지(Arc)를 거쳐 제약조건을 다른 노드에 전파하여 충돌된 도메인을 제거함으로써 제약조건의 일치성을 유지한다.

일반적으로 제약 만족 기법을 해결하는 기법은 가능해 집합을 나열하여 가능해나 최적해를 찾아내는 것이다. 문제들을 수학적인 프로그램밍 또는 모델링을 통한 방법이

아닌 제약만족 기법의 모델로 모델링을 하고 해를 찾고자 하는 이유는 문제 표현을 좀더 문제 그 자체에 가깝게 표현할 수 있기 때문이다. 다시 말해서 제약 만족 기법의 변수들은 논리적인 표현으로 표현이 되고 형식적인 표현 면에서도 간단하며 이해하기 쉬울 뿐만 아니라 좋은 경험적 탐색 방법의 선택도 직관적으로 이루어 질 수 있다.

제약 만족 기법을 이용한 시스템의 기본적인 특징으로는 첫째, 제약식의 정의 부분과 제약전파 부분, 문제 해결을 위한 검색알고리즘 부분을 서로 분리하고 있다. 둘째, 각 제약식은 가능한 한 국부적으로 전파되도록 한다. 즉, 선언부에서 선언된 제약식만을 가지고 불필요한 도메인을 전파를 통해 줄인 다음, 적절한 검색 알고리즘을 이용하여 해를 구하게 된다.

2.2 ILOG

ILOG Solver는 제조, 운송, 통신 등의 여러 분야에서 조합 최적화 문제에 적합한 Constraint-Based Programming을 기반으로하는 프로그램밍 C++라이브러리로 미리 정의 된 변수 클래스, 새로운 제약과 결합이 가능한 제약 클래스, 새로운 발견적 기법을 적용할 수 있는 탐색 알고리즘으로 이루어져 있다. 즉 클래스를 통해서 객체지향프로그래밍 개념을 실천함으로서 코드의 재활용 및 확장성이 용이하고 실제 발생할 수 있는 여러 상황에도 충분히 적용 될 수 있다.

ILOG Solver의 특징은 제약조건 프로그램밍과의 결합을 통해 조합최적화와 관련된 문제들을 풀기 위한 효과적인 방법과 알고리즘을 제공하여 이런 두 가지 프로그램밍 방법을 이용할 수 있도록 클래스를 설계하고 이를 라이브러리로 구축하여 문제를 해결할 수 있도록 방법을 제시하고 있다[4].

ILOG Solver의 제약조건 변수는 기존의 제약조건 프로그래밍 언어의 한계와는 대조적으로 어떠한 형태의 값도 사용할 수 있다. 정수, 실수 형은 물론 사용자 정의형의 테이터를 사용할 수 있고, 제약된 정수변수, 제약된 실수 변수, 제약된 집합 변수 등 3종류의 제약조건 변수를 제공한다. 즉, ILOG Solver는 문제의 풀이와 문제의 표현이 분리되어 있다. ILOG Solver를 통한 알고리즘의 구현은 제약조건과는 독립적으로 탐색 Logic을 다룬다.

ILOG Scheduler는 ILOG Solver에 추가되어 제약조건문제의 스케줄링 문제와 자원할당문제를 해결하기 위한 C++ 라이브러리로 Solver에서 제공하는 변수와 제약조건에 관한 Class를 이용하며 Activity와 Resource라는 개념을 사용하여 문제를 표현하고 이를 해결하는 역할을 한다. 일반적으로 Scheduler를 적용하여 문제를 해결하는 절차는 다음과 같다.

1. Scheduler가 제공하는 일반적인 제약조건을 사용하여 문제에 적당한 모델을 설계.
2. 필요하다면 문제에 따르는 특정 제약조건을 Solver를 이용하여 추가로 정의
3. Scheduler에서 제공하는 Goal을 이용하여 탐색 알고리즘을 성정하고 이를 Solver와 연결.
4. 제약조건의 Propagation을 통해 탐색이 이루어지고 결과를 출력.

2.3 Randomizing 알고리즘

기존의 반복적 개선에 근거한 발견적 기법들이 지역 최소점에 빠지는 단점을 개선하고 합리적인 시간 내에 최적 해를 탐색하지 못하는 경우 좀 더 빠른 시간 내에 좋은 해을 탐색하는 알고리즘이다. 이러한 알고리즘에 접근은 두 가지 방법으로 구성되어 있다.

1. 순차적인 백트레킹이 해를 구하기 위해 사용되지만 일정한 합리적인 수의 트레킹이 발생 후 해를 발견하지 못하면 그 탐색은 멈추고 새로운 탐색을 시작한다.
2. 전에 탐색했던 Path와는 다른 Path를 선택해서 탐색을 시작한다.

Randomizing 알고리즘은 해 탐색 시 실패수 혹은 탐색 시간을 제한하여 제한된 수 혹은 시간 내 해를 탐색하지 못한 경우 해 탐색을 멈출 수 있어 좋은해를 찾는데 실패 할 수 있다. 하지만 실패수 혹은 탐색 시간과 반복수를 합리적으로 제한 한다면 좋은 해를 구할 수 있는 장점이 있다.[7].
Randomizing 알고리즘을 사용하는 것은 해 탐색 시 모든 과정에서 램덤하게 진행이 된다는 것을 의미하지 않는다. 새로운 해를 찾고, 더 좋은 해를 찾아 갱신하는 알고리즘의 구조는 최적해 알고리즘들과 비슷하게 진행이 되지만 값 정렬을 하는 과정에 있어서 차이점이 발생한다.

<Randomizing 알고리즘의 절차>

1. 초기해를 생성하고 알고리즘의 반복수 및 실패 수를 초기화 한다.
2. 초기해와 비교를 하여 갱신 여부를 판단하기 위해 개선해를 생성시킨다.
3. 모든 자원(Resource) 내 작업(Activity)을 탐색하여 처음에 일정계획을 할 수 있는 작업의 수를 구한다.
4. 처음에 일정계획을 할 수 있는 작업의 수가 가장 적은 자원을 선택하고 이 작업의 수중에서 램덤하게 정렬(Ordering)을 한다.
5. 모든 작업이 자원에 정렬 될 때 까지 3단계와 4단계를 반복한다.
6. 개선해가 초기해보다 값이 작을 경우 개선해로 해를 갱신한다.
7. 모든 반복이 끝났는지를 확인하고 종료한다.

Randomizing 알고리즘만을 사용하여 해를 구할 경우 좋은 해을 구할 수 있지만 만족할 만한 해를 탐색 하지 못하는 문제들도 있다. 따라서 Randomizing 알고리즘과 최적해 알고리즘을 조합하여 문제에 적용을 할 경우 좀 더 좋은 해가 나올 수 있기 때문에 고려해 볼 필요가 있다.

3. 제안된 기법을 이용한 Job Shop 일정계획

3.1 문제 해결 방법

제약만족기법을 기반으로 하여 일정 계획의 문제들을 해결하고자 할 때 일정 계획의 문제들의 크기가 커질 경우 해를 탐색하기가 불가능하거나 해를 탐색했을 경우 계산 시간이 길어지는 단점이 있다. 본 연구는 이러한 단점을 해결하고자 적절한 탐색 시간 내에 효율적인 해를 탐색하기 위해서 개선된 Randomizing 알고리즘을 적용한 발견적 기법을 제시한다.

2장에서 소개한 ILOG Solver와 Scheduler는 제약만족기법을 기반으로 하는 Constraint Programming C++ Library로 구성되어 있고 Solver와 Scheduler의 장점은 주어진 문제를 해결하기 위해 기존에 연구되어진 여러 다른 알고리즘들과의 결합이 쉽게 된다. 즉 제약조건 기반의 최적화는 문제를 해결하기 위한 알고리즘의 틀을 제공하는 것이며, 사용자의 알고리즘은 Solver에서 제공하는 제약변수를 통하여 적용될 수 있다.

ILOG Solver에서 기본적으로 제공하는 탐색알고리즘은 2장에서 소개한 것처럼 모든 해 영역을 탐색하여 가능해 또는 최적해를 탐색하는 것이다. 이때 계산시간을 줄이기 위해 제약전파를 이용한 도메인 여과기법을 이용하여 탐색 공간을 줄이지만 문제의 크기가 큰 경우 최적해를 탐색하는데 있어 아직 어려움이 있기 때문에 이러한 문제들을 해결하기 위해서 탐색과정에 발견적 기법을 적용하는 것이다.

본 연구는 Job Shop 일정계획 문제에서 규모가 큰 문제들에 적용하여 합리적인 계산 시간 내에서 근사해를 탐색하기 위한 알고리즘의 개발을 목적으로 하고 있다. 따라서 좋은 초기해를 생성하여 전체적인 해의 탐색 공간의 범위를 줄이고 계산시간의 감소를 위해서 일정한 횟수의 탐색이 실패할 경우 탐색을 멈추고 그 시점까지 발견된 해 중에서 가장 좋은해를 선택하여 다음 단계의 개선해로 정하고 해를 개선해나가는 방법을 이용한다.

3.2 제안된 알고리즘

본 연구에서는 제 2장에서 설명한 Randomizing 알고리즘의 초기해 생성과정과 개선해 탐색과정을 개선하기 위하여 초기해 탐색 알고리즘과 개선해 탐색 알고리즘을 구현하여 제시하였다.

3.2.1 초기해 탐색 알고리즘

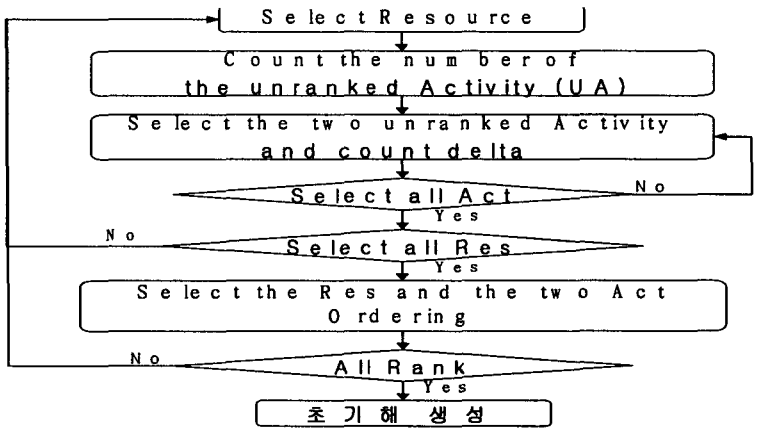
제약만족기법에서 문제의 계산시간과 탐색공간을 줄이기 위한 방법으로 총처리시간 변수의 도메인 값을 정해 주고 제약 전파를 해줌으로서 각 공정 변수들의 도메인을 어느 정도 정해준다. 하지만 초기해의 도메인이 너무 작게 되면 후에 해를 탐색하지 못하는 경우가 있기 때문에 정당한 크기를 가지는 좋은 초기해를 탐색해야 한다.

본 연구에서 이러한 방법을 적용하여 좋은 초기해를 탐색하는 알고리즘을 구현하였으며 절차는 다음과 같다.

<초기해 탐색 알고리즘 절차>

1. 일정계획 되지 않는 자원들 중 하나를 선택 그 자원 내 선후관계가 결정되지 않은 Activity(Unranked Avtivity)의 수를 계산 한다.
2. 선택된 자원 내에 한 쌍의 Unranked Activity를 선택 후 이 Activity들의 순서(Precedence Constraint)를 결정할 때 발생되는 Domain Reduction의 크기인 delta를 계산한다.
3. 모든 자원내의 Unranked Activity들의 수를 계산하고. 모든 Activity간의 delta를 계산할 때까지 1단계, 2단계를 반복한다.
4. 자원들 중 Unranked Activity의 수가 가장 적은 자원을 선택하고 그 자원 내에 delta가 가장 큰 두 Activity를 선택하여 일정계획하고 모든 Activity들 이 일정계획 될 때까지 1단계, 2단계, 3단계를 반복한다.
5. 초기해 생성

이러한 절차에 의해서 해 탐색 시 반복 횟수와 탐색 시간을 줄일 수 있는 좋은 초기해를 탐색할 수 있고, 이 초기해를 이용하여 새로운 개선해에 더욱 빨리 접근할 수 있을 것이다. < 그림 1 >는 초기해 탐색 알고리즘의 흐름도를 보여 준다.



< 그림 1 > 초기해 탐색 알고리즘의 흐름도

3.2.2 개선해 탐색 알고리즘

탐색 되어진 초기해 내에서 개선해를 탐색하는 알고리즘으로는 Randomizing 알고리즘을 개선하여 제시하였으며 그 절차는 다음과 같다.

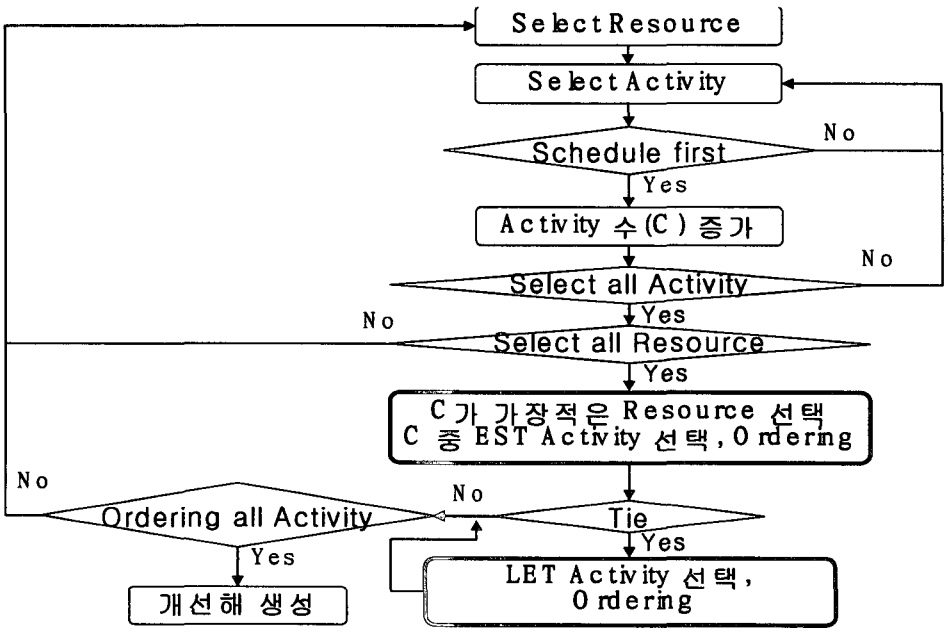
<개선해 탐색 알고리즘 절차>

1. 일정 계획 되지 않는 자원을 선택하여 이 자원 내 모든 Activity 중에서 먼저 일정

계획이 될 수 있는 Activity의 수를 검사한다.

2. 검사된 자원들 중 검사된 Activity들의 수가 가장 적을 자원을 선택하고 Activity들 중 가장 작은 EST(Earliest Start Time)의 Activity를 일정계획하고 만약 EST가 같은 다른 Ativity가 존재한다면 같은 Activity 중 가장 큰 LET(Latest End Time)한 Activity를 선택하고 일정계획을 한다.
3. 모든 Activity들이 일정 계획이 될 때까지 1단계, 2단계를 반복 한다.
4. 개선해 생성

본 연구는 기존의 Randomizing 알고리즘에서 Activity들을 랜덤하게 일정계획을 하는 것을 달리하여 앞에서 소개한 것 같이 모든 Activity들의 일정계획을 한다. < 그림 2 >는 개선해 탐색 알고리즘의 흐름도를 보여준다.

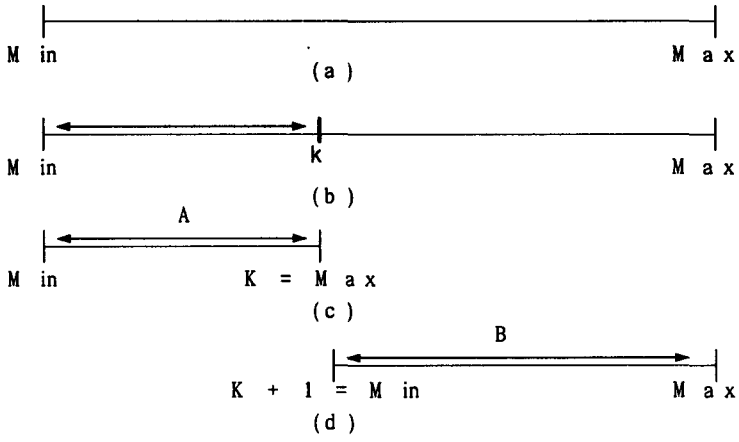


< 그림 2 > 개선해 탐색 알고리즘의 흐름도

3.2.3 비용함수의 탐색방법

본 연구에서는 비용함수를 구하기 위해서 모든 Activity들의 총처리시간 변수의 도메인의 영역 변화를 적용한다. 이 절차는 발생한 총처리시간의 변수의 값의 범위를 등분하는 과정으로 비용함수의 값 즉 해를 구할 수 있고 다음과 같이 진행된다. < 그림 3 >의 (a)와 같이 총처리시간 변수의 도메인을 초기해의 Lower bound를 Min으로 Upper bound를 Max로 하고 다음의 해를 탐색을 위한 총처리시간의 도메인을 결정하기 위하여 (Min+Max)/(Min+Iteration)을 만족하는 값을 K라 하고 이를 총처리시간의 Upper bound

로 하여 < 그림 3 > (b)같이 해를 탐색한다. < 그림 3 > (C)와 같이 A구간에서 해가 탐색된다면 K를 총처리시간의 Upper bound로 정하여 새로운 해를 탐색하고 만약 해가 탐색되지 않는다면 < 그림 3 > (d)와 같이 K+1를 총처리시간의 Lower bound(Min) 으로 정하여 B구간에서 새로운 해를 탐색한다.



< 그림 3 > 총처리시간의 도메인 영역변화

본 연구에서는 비용함수의 도메인 영역을 이용하여 해를 탐색하였으며 일정한 횟수의 반복을 통해서 해를 탐색하고 멈추게 된다.

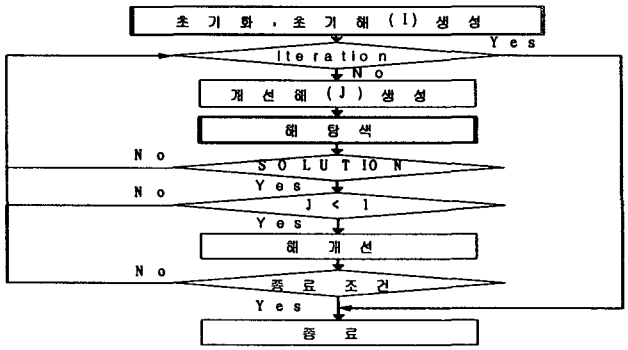
3.3 개선된 Randomizing 알고리즘

위에서 제안된 초기해 탐색 알고리즘과 개선해 탐색 알고리즘을 적용한 개선된 Randomizing 알고리즘의 절차는 다음과 같다.

<개선된 Randomizing 알고리즘 절차>

1. 반복횟수, 실패 횟수 등의 파라메타들을 초기화 하고 초기해 탐색 알고리즘에 의해서 초기해를 생성한다.
2. 초기해의 도메인을 총처리시간의 도메인으로 하고 (Min+Max)/(Min+Iteration)을 만족하는 값을 총처리시간의 Upper bound로 하며 이를 개선해로 생성한다.
3. 개선해 탐색 알고리즘을 적용하여 개선해를 탐색하고 초기해와 비교하여 개선해가 좋은 해를 가졌다면 갱신한다.
4. 정해진 반복횟수를 확인하고 종료여부를 판단한다.

2단계에서 개선해 탐색 알고리즘을 이용 개선해를 탐색할 때 정해진 실패수내에 해를 찾지 못하면 해 탐색 과정을 멈추고 다음 단계로 진행이 된다. 이러한 절차를 흐름도로 나타내면 다음과 같다.



< 그림 4 > 개선된 Randomizing 알고리즘의 흐름도

3.4 프로그램의 구현 및 구성

본 연구에서 일정계획을 구현하기 위해 사용된 프로그래밍 언어는 ILOG Solver5.1, ILOG Scheduler5.1 라이브러리와 이를 사용하기 위한 플랫폼으로는 Visual C++ 6.0을 기반으로 하여 프로그램을 구현하였다. 또한 구현된 일정계획의 결과는 ILOG View4.1을 이용하여 간트차트로 출력을 하였다. 이들을 위해 사용된 컴퓨터는 Windows를 OS로 사용한 CPU P-Ⅱ 800Mhz 인 삼보 PC이다.

4. 실험 및 분석

본장에서는 Job Shop일정계획 벤치마킹 문제를 본연구에서 제안된 발견적 기법에 적용한 결과와 제약만족기법이 적용된 기존의 다른 연구들의 결과를 비교한 것을 제시한다. 벤치마킹에 적용된 문제들은 Muth와 Thompson의 MT문제들과 Lawrence가 제안한 LA문제를 중 문제 크기별로 몇 가지를 선택하여 적용하였다.

4.1 실험 방법 및 결과

실험에 필요한 파라미터로 제시되는 실행횟수와 실패수는 문제의 크기에 따라 임의의 수를 정하여 실시하였다. 본 연구는 일정한 실패수내에 해를 발견하지 못하면 해의 탐색이 멈추어 탐색을 멈추거나 다시 시작하는 알고리즘을 구현하였고 해가 탐색이 되는 최대의 반복수와 실패수로 탐색을 하여 근사해를 탐색하였다.

본 연구의 결과를 보면 해는 각 문제에서 탐색된 근사해를 나타낸 것이고 CPU Time은 해가 발견 되었을 때 CPU시간을 나타낸다.

다음 < 표 1 >은 제안된 본 연구에서 적용되는 MT문제들 중에 하나인 MT10 문제를 보여주며 < 표 2 >는 MT06, MT10, MT20 문제들을 적용하여 탐색되어진 결과를 나타낸 것이다.

< 표 1 > MT10 벤치마킹 문제

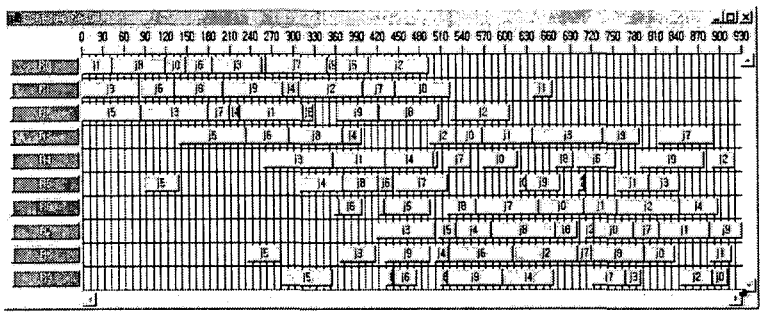
작업																			
J_1		J_2		J_3		J_4		J_5		J_6		J_7		J_8		J_9		J_{10}	
m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i	m_i	p_i
1	29	1	43	2	91	2	81	3	14	3	84	2	46	3	31	1	76	2	85
2	78	3	90	1	85	3	95	1	6	2	2	1	37	1	86	2	69	1	13
3	9	5	75	4	39	1	71	2	22	6	52	4	61	2	46	4	76	3	61
4	36	10	11	3	74	5	99	6	61	4	95	3	13	6	74	6	51	7	7
5	49	4	69	9	90	7	9	4	26	9	48	7	32	5	32	3	85	9	64
6	11	2	28	6	10	9	52	5	69	10	72	6	21	7	88	10	11	10	76
7	62	7	46	8	12	8	85	9	21	1	47	10	32	9	19	7	40	6	47
8	56	6	46	7	89	4	98	8	49	7	65	9	89	10	48	8	89	4	52
9	44	8	72	10	45	10	22	10	72	5	6	8	30	8	36	5	26	5	90
10	21	9	30	5	33	6	43	7	53	8	25	5	55	4	79	9	74	8	45

m_i : 작업의 operation이 점유하는 기계. p_i : operation의 가공시간.

< 표 2 > 제안된 기법의 MT문제들의 적용 결과

	작업 * 기계	최적해	제안된 알고리즘	
			해	CPU Time (단위:초)
MT06	6 * 6	55	55	0.03
MT10	10 * 10	930	930	1.482
MT20	20 * 5	1165	1165	0.241

제안된 발견적 기법에서 탐색된 해들 중에 MT10문제의 결과를 Gantt Chart로 출력하면 다음 < 그림 5 >과 같이 나타난다.



< 그림 5 > MT10 해의 Gantt Chart

본 연구에서 제안된 발견적 기법은 MT문제에선 모두 최적해를 구하였다. 특히 까다롭고 크기가 큰 문제인 MT10문제를 빠른 시간에 해를 탐색한 것은 더욱 큰 문제에서도 해를 탐색할 수 있는 가능성을 주었다.

< 표 3 >은 LA19, LA24, LA27, LA40 문제들을 적용하여 탐색되어진 결과를 나타낸 것이며 최적의 근사해를 탐색하지는 못했지만 큰 문제들을 대상으로 빠른 시간에 문제들의 근사해를 얻었다.

< 표 3 > 제안된 기법의 LA문제들의 적용 결과

	작업 * 기계	최적해	제안된 알고리즘	
			해	CPU Time
LA19	10 * 10	842	848	0.971
LA24	15 * 10	935	937	3.395
LA27	20 * 10	1235	1261	5.258
LA40	15 * 15	1222	1237	6.439

4.2 제안된 알고리즘과 기존 연구와의 비교

본 연구에서 제안된 발견적 기법과 제약만족기법을 적용한 기존의 방법인 Randomizing 알고리즘 및 Dichotomizing binary search 알고리즘과의 비교를 하였다. 이와 같이 비교하는 것은 동일한 컴퓨터 환경에서 구현하여 비교함으로서 본 연구에서 제안한 발견적 기법의 객관적인 수행도 평가를 제시할 수 있기 때문이다. 따라서 최적해에 가장 근접한 근사해를 구하기 위한 계산시간을 상호 비교하면서 본 연구의 알고리즘의 우수성을 제시할 수 있다.

4.2.1 MT 문제들에서의 비교

본 연구에서 제안된 발견적 기법과 제약만족기법을 적용한 기존의 방법들을 비교한 결과는 < 표 4 >과 같다.

< 표 4 > 제안된 발견적 기법과 기존의 방법과의 비교

		Randomizing 알고리즘		Dichotomizing binary search 알고리즘		제안된 알고리즘	
	최적해	해	CPU Time	해	CPU Time	해	CPU Time
MT06	55	55	0.16	55	0.11	55	0.03
MT10	930	996	6.43	930	155.28	930	1.482
MT20	1165	1165	6.92	1165	40	1165	0.241

< 표 4 >에서 나타난 것 같이 기존의 방법들과 비교해 본 결과 본 연구에서 제안된 발견적 기법과 Dchotomizing 알고리즘은 모든 MT문제들에서 최적해와 동일한 해를 탐색하였고 Randomizing 알고리즘은 MT10문제에서 최적해와 많이 차이가 있는 값을 탐색하였다. 전체적인 해를 살펴보면 Randomizing 알고리즘이 MT10문제에서 최적해

를 탐색하지 못한 반면 본 알고리즘은 최적해를 탐색하였고 Dichtomizing 알고리즘 역시 모든 MT문제들에서 최적해를 탐색하였지만 계산시간에 있어서 본 제안된 알고리즘이 우수하다는 것을 보여주고 있다.

결론적으로 MT문제들을 해결하는데 있어 본 제안된 알고리즘이 기존의 방법들 보다 우수하다 할 수 있겠다.

4.2.2 LA 벤치마킹문제들에서의 비교

본 연구에서 제안된 기법과 제약만족기법을 적용한 기존의 방법들을 비교한 결과는 < 표 5 >와 같다.

< 표 5 > 제안된 발견적 기법과 기존의 방법과의 비교

		Randomizing 알고리즘		Dichotomizing binary search 알고리즘		제안된 알고리즘	
	최적해	해	CPU Time	해	CPU Time	해	CPU Time
LA19	842	851	13.46	842	81.4	848	0.971
LA24	935	967	24.66	-		937	3.395
LA27	1235	1310	23.61	-		1261	5.258
LA40	1222	1281	20.54	-		1237	6.493

- : 1시간 내 해를 찾지 못함

< 표 5 >에서 Randomizing 알고리즘은 LA문제들 모두 합리적인 시간 내에 해를 탐색하였지만 탐색 되어진 해가 최적해와 많은 차이를 보여주고 있고 Dichotomizing binary search 알고리즘은 LA19문제에서는 최적해를 탐색하였지만 계산시간이 효율적이지 못하며 LA24, LA27, LA40 문제에서는 해를 탐색하지 못하였다. < 표 5 >에서는 해를 탐색하기 위해 이용된 컴퓨터에서 1시간의 계산시간 내에 해를 탐색하지 못한다는 것을 의미한다. 본 연구에서 제안된 알고리즘도 역시 최적해를 탐색하지 못했지만 기존의 방법인 Randomizing 알고리즘보다는 최적해에 가까운 좋은 근사해를 적은 시간 내에서 탐색을 하였고 Dichotomizing binary search 알고리즘에서는 전혀 탐색하지 못한 해를 탐색할 수 있었다. 따라서 본 연구에서 제안된 알고리즘이 기존의 두 방법 보다 효율적이고 우수한 알고리즘이라 제시할 수 있다.

5. 결 론

본 연구에서는 대규모 일정계획의 문제들에서 합리적이고 효율적인 시간 내에서 최적해에 최대한 근접한 근사해를 탐색해 내기 위한 발견적 기법에 관한 것이다. 이를 구현하기 위해서 제약만족기법을 기반으로 하는 발견적인 기법을 제시하였다. 전체 탐색공간에 대하여 탐색하며 적절한 시간내에서 해를 찾지 못하는 문제점을 가지고 있는 최적화기법의 단점과 최적해에서 멀리 떨어진 해를 구하는 근사해법의 문제점을 본 연구에서 제안된 발견적 기법에 의해서 해결할 수 있다고 사료된다. 이는 본 연구

에서의 발견적 기법과 기존의 방법들을 MT문제, LA문제들을 가지고 비교한 결과 문제가 커질수록 본 제안된 발견적 기법이 기존의 방법들 보다 근사해와 계산시간에 있어서 우수하다는 것으로 알 수있다. 결론적으로 본 연구에서 제안된 발견적 기법인 개선된 Randomizing 알고리즘은 대규모의 Job Shop 일정계획문제에서 해의 탐색에서 합리적이고 효과적인 방법이라고 할 수 있다.

추후 연구과제는 본 연구에서 제안된 발견적 기법에 Tabu Search 등의 메타휴리스틱의 적용가능성 여부에 대한 것이다.

6. 참 고 문 헌

[1] Brailsford, S. C., C. N. Potts and B. M. Smith, "constraint satisfaction problems: Algorithms and applications", *European Journal of Operational Research*, 119 (1999) : 557-581.

[2] Dorndorf, U., Pesch, E. and T. Phan-Huy, "Constraint propagation techniques for the disjunctive scheduling problem", *Artificial Intelligence*, 122 (2000) : 189-240.

[3] ILOG, ILOG Solver manual, ILOG, (2002).

[4] ILOG, ILOG Scheduler manual, ILOG, (2002).

[5] Nuijten, W. and L. P. Claude, "Constraint-Based Job Shop scheduling with ILOG Scheduler", *Journal of Heuristic*, 3 (1998) : 271-286.

[6] Nuijten, W., L. P. Claude and P. Baptiste, "constraint-based Optimization an Approximation for Job-Shop scheduling", *Proceedings of the AAAI-SIGMAN Workshop on intelligent Manufacturing Systems*, (1995).

[7] Walsh, T.,"Depth-bounded Discrepancy Search", *APES Group,* (1998).

[8] Wang, L. and D. Z. Zheng, "An Effective hybrid optimization strategy ofr job-shop scheduling problems", *Computers & Operations Research*, 28 (2001) : 585-596.

[9] Weigel, R. and B. Faltings, "Compiling constraint satisfaction problems", *Artificial Intelligence*, 115 (1999) : 257-287.

저 자 소 개

이 화 기 : 서울대학교 원자핵공학과 학사, 미 Texas A&M 대학교 산업공학과 석사 및 박사. 현재 인하대학교 산업공학과 교수.
관심분야는 생산분야 및 물류분야의 일정계획, 시뮬레이션 등이다.

김 민 석 : 관동대학교 산업공학과 학사, 인하대학교 산업공학과 석사.
관심분야는 생산관리, 생산정보시스템 등이다.

이 승 우 : 인하대학교 산업공학과 학사, 석사 및 박사과정 수료.
현재 한국기계연구원 지능형정밀기계연구부 선임연구원으로 재직.
관심분야는 생산정보시스템, 전문가시스템 등이다.