Journal of the Korean Institute
of Industrial Engineers
Vol.21, No.3, September, 1995

Job Shop 일정계획을 위한 Tabu Search*

Tabu Search for Job Shop Scheduling

김여근**, 배상윤**, 이덕성**

Yeo Keun Kim**, Sang Yun Bae**, Deog Seong Lee**

Abstract

Job shop scheduling with *m* different machines and *n* different jobs is a NP-hard problem of combinatorial optimization. The purpose of this paper is to develop the heuristic method using tabu search for job shop scheduling to minimize makespan or mean flowtime.

To apply tabu search to job shop scheduling problem, in this paper we propose the several move methods that employ insert moves in order to generate the neighbor solutions, and present the efficient rescheduling procedure that yields active schedule for a changed operation sequence by a move of operations. We also discuss the tabu search techniques of diversifying the search of solution space as well as the simple tabu search. By experiments, we find the appropriate tabu list size and tabu attributes, and analyze the proposed tabu search techniques with respect to the quality of solutions and the efforts of computation.

The experimental results show that the proposed tabu search techniques using long-term memory function have the ability to search a good solution, and are more efficient in the mean flowtime minimization problem than in the makespan minimization.

* 이 연구는 1994년도 한국과학재단 연구비 지원에 의한 결과임(과제번호:941-1000-016-1)

** 전남대학교 공과대학 산업공학과

1. 서론

본 연구는 m대의 기계에서 상이한 가공순서와 가공시간을 갖는 n개의 부품(job)을 처리하는 job shop의 일정계획(scheduling) 문제를 다루고 있다. Job shop일정계획 문제(이하 JSP라 함)는 부품 생산을 위한 공정(operation)간의 순서제약(precedence constraint)과 가공가능 시점에서의 기계제약(capacity constraint) 아래서 각 기계에 효과적으로 각 부품의 작업(공정)을 할당하는 일종의 조합최적화 문제이다.

JSP의 연구는 크게 최적해를 구하는 기법과 최적근사해를 구하는 발견적(heuristic) 기법으로 나누어 볼 수 있다. 분지한계법 등을 이용하여 job shop일정계획의 최적해를 구하는 문제는 NP-hard문제로서 부품과 공정의 수가 많아지면 최적해를 구하기 어렵게 된다. 따라서 그 해결 방안으로 발견적 기법이 흔히 사용된다.

발견적 기법의 연구는 active일정이나 nondelay일정의 집합을 생성해 가는 과정에 할당규칙(priority dispatching rules)을 적용하는 기법[1, 5, 9, 14, 19]과 기계들의 부하를 평활화하여 부품의 총처리시간(makespan)을 최소화하는 Yang et al.[25] 등의 연구가 있다. Active일정은 어떤 공정도 다른 공정의 가공지연없이 더 빨리 시작될 수 없는 일정이고 nondelay일정은 기계에서 부품 가공이 가능할 때 그 기계가 쉬지 않도록 작업을 할당하는 일정을 의미한다. Baker[1]와 Hutchison et al.[14]은 active일정과 nondelay일정의 생성방법을 제안하였고 Blackstone et al.[5], Fry et al.[9], Panwalkar et al.[19]은 기존에 제안된 여러가지 할당규칙들을 분석하였다.

최근에는 JSP의 해결을 위하여 인공지능의 여러 기법들이 제안되고 있다. 자연의 진화과정과 유전법칙을 모방한 유전알고리듬(genetic algorithm)[6]과 금속의 냉각과정을 응용한 simulated annealing[16], 인간의 뇌기능을 모방한 신경망(neural network) 이론[8]과 인간의 기억과정을 이용한 tabu search(이하 TS라 함)[7, 23, 24]등이 연구되었다. 본 연구는 기존의 발견적기법들이 갖는 부분최적 또는 불충분한 탐색과정을 극복하는 방안으로 JSP를 위한 TS 기법을 개발하고자 한다.

TS는 Glover[11]에 의해 연구되기 시작한 일종의 탐색기법이다. 이는 해를 탐색한 과정을 기억하고 되돌아 가는 것을 금지(tabu)하여, 해의 순환(cycling)을 방지하고 부분최적에서 벗어날 수 있는 기법이다. TS를 특정 문제에 적용할 때, 해공간의 탐색성능은 현재해로 부터 이웃해들을 생성하는 이동(move) 방법, 이동에 관한 정보를 규정하는 tabu속성(attribute), 이것을 기억하고 일정 기간이 지나면 이동의 tabu상태를 해제할 수 있게 하는 tabu목록과 tabu로 지정된 이동을 해제하는 열망수준(aspiration level)등에 영향을 받게 된다.

TS기법은 모형화가 쉬우며[3], 조합최적화 문제에서 빠른 시간에 근사최적해를 찾는데 적합한 기법으로 알려져 있다[20, 21]. TS의 이러한 장점으로 인해 단일기계 일정계획[17], 복수기계 일정계획[2], 이차배정문제[15, 20, 22], 차량경로문제[10], flow shop일정계획[3, 4, 21] 및 job shop일정계획[7, 23, 24] 등의 다양한 분야에 응용되고 있다.

TS의 이론 및 응용에 관한 활발한 연구에도 불구하고 이 기법을 job shop일정계획에 적용한 연구는 미비한 실정이다. Dell'Amico et al.[7]와 Sun et al.[23]은 TS기법을 이용하여 부품의 총처리시간을 최소로 하는 job shop일정계획방법을 제시하였다. Widmer[24]는 TS를 이용하여 tool제약을 고려한 일정계획 문제를 다루고 있으나 단일공정인 경우만을 연구하였다. 총처리시간 최소화는 모든 부품의 처리를 완료하는 최소 시간을 구하는 의미를 갖지만 각 부품의 처리시간을 고르게 하지는 못한다. 평균처리시간(mean flowtime)은 각 부품의 완료시간의 평균으로 공장의 혼잡도와 재공재고의 수준을 나타내는 척도로서 사용된다[1]. 따라서 본 연구에서는 총처리시간 또는 평균처리시간 최소화의 목적에 사용 가능한 TS기법을 연구하고자 한다.

JSP에 TS를 적용함에 있어서 탐색성능을 향상시킬 수 있는 이웃해 생성방법과 TS의 파라메터에 대한 연구가 요구된다. 부품의 총처리시간과 평균처리시간의 최소화를 목적함수로 갖는 JSP의 최적해는 active일정 집합에 존재[1]하기 때문에, 이 집합만을 탐색하기 위하여 생성된 이웃해의 active를 유지시키는 방법이 개발되어야 한다. 또한 TS기법에서 좀 더 좋은 해를 얻기 위해서는 해공간을 다양하게 탐색하여야 한다.

Job shop일정계획을 위한 TS기법을 개발하기 위하여, 본 연구에서는 TS에서 이웃해를 생성하는 여러 이동방법을 제시하고, 가공순서의 변경(이동)에 따른 재일정계획을 위하여 현재의 일정으로부터 효율적으로 active일정을 재생성하는 방법을 개발한다. 또한 JSP에 적합한 단순TS기법과 TS의 다양화 기법들을 제안한다. 실험을 통하여 총처리시간 또는 평균처리시간의 최소화를 목적으로 하는 job shop일정계획에 적합한 tabu속성, tabu목록크기 등에 관하여 연구하고, 제안한 여러 TS기법들을 비교분석하며 또한 기존의 할당규칙들과의 알고리듬 성능을 비교분석한다.

본 연구의 구성은 다음과 같다. 제 2장에서 job shop일정계획과 TS기법을 소개하고, 제 3장에서는 TS의 이동방법과 active재일정계획 방법에 의한 이웃해 생성방법을 제안하며, 제 4장에서는 TS에 의한 job shop일정계획절차를 제시하고 해 공간의 탐색을 다양화하는 기법을 제안한다. 제 5장에서는 실험을 통하여 tabu파라메터와 제안한 이동방법을 비교분석하고, 개발한 기법을 기존의 할당규칙과 비교한다. 제 6장은 결론으로 구성되어 있다.

2. Job Shop 일정계획과 Tabu Search

본 연구에서 다루는 JSP은 아래의 가정을 갖는다.

(1) 한 부품을 생산하기 위한 공정간의 가공순서는 제약되지만 서로 다른 부품의 공정간의 가공순서는 제약되지 않는다.

(2) 공정을 가공하는 기계는 특정한 하나의 기계로 주어진다.

(3) 기계는 어떤 가공 시점에 하나의 공정만을 가공한다.

(4) 기계가 어떤 공정을 가공하는 도중에 다른 공정이 그 기계에서 가공을 시작할 수 없다.

2.1 단순 Tabu Search

TS는 초기 가능해를 생성하여 이를 현재해로 하고, 이로 부터 이웃해 집합을 생성하여 tabu이동(tabu목록에 있는 tabu속성을 갖는 이동)이 아니거나 tabu이동이지만 열망수준을 만족하는 이웃해에서 가장 좋은 해, 즉 최선해(best solution)를 (새로운)현재해로 하고, 이 이동 속성을 tabu목록에 일정기간 저장하면서 이 과정을 종료조건이 만족될 때까지 반복하는 기법이다. 이러한 TS를 단순(simple) tabu search라 부른다[11]. 이 기법의 구체적인 절차는 아래와 같다.

〈단순Tabu Search 절차〉

단계 1. (초기화)

1) Tabu속성, tabu목록크기, 열망수준과 종료조건 등을 결정한다.

2) 초기 가능해를 구하여 현재해와 최선해로 둔다.

3) Tabu목록을 비어 둔다.

단계 2. (이웃해 생성)

현재해로 부터 tabu이동이 아니거나 tabu이동이지만 열망수준을 만족하는 이동에 대해 이웃해를 생성한다.

단계 3. (현재해와 최선해 수정)

1) 생성된 이웃해 중에서 가장 좋은 해를 현재해로 한다.

2) 현재해가 최선해보다 좋으면 현재해를 최선해로 둔다.

단계 4. (Tabu목록수정)

새로운 현재해의 이동속성을 tabu목록에 기록한다. 그리고 만약 tabu목록에 저장된 tabu속성의 수가 tabu목록크기 보다 크면 가장 먼저 기록된 tabu속성을 삭제한다.

단계 5. (종료기준 검사)

만약 종료기준을 만족하면 끝내고, 그렇지 않으면 단계 2.로 간다.

TS기법에서는 현재해의 여러 이웃해를 생성함으로써 해 공간의 탐색을 강화(intensification)하는 한편, tabu목록에 있는 해로의 이동을 막음으로써 해 공간의 탐색을 확장해 나간다[11]. 또한 단순TS기법에서는 최근에 일어난 한정된 이동만을 tabu목록에 기억하는 단기기억(short-term memory)을 사용하고 있다. 해공간을 다양하게 탐색(diversification)하기 위하여 해의 이동에 관한 정보를 처음부터 현재까지 기억하는 장기기억(long-term memory)을 이용하는 여러 연구가 이루어 졌다[15, 20, 21, 22].

2.2 이웃해

본 연구에서 해의 표현은 3장에서 제안하는 이동방법과 active재일정계획에 적합한 표현으로써, 각 기계에서의 가공순서와 각 공정의 시작시간으로 표현한다. 이는 Gantt chart와 같은 표현으로 그림 2.1과 같이 된다.

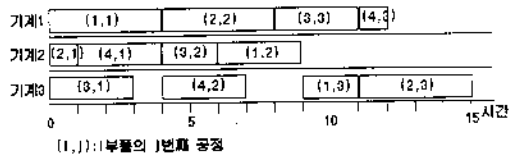


그림 2.1 해의 표현방법

TS에서 이동방법으로는 흔히 해의 원소를 교환, 삽입, 삭제, 첨가하는 방법을 사용한다. 특히 해에서 두개 원소의 위치를 바꾸는 교환이동(swap move)과 해의 한 원소를 다른 위치로 삽입하는 삽입이동(insert move)이 가

TS의 이론 및 응용에 관한 활발한 연구에도 불구하고 이 기법을 job shop일정계획에 적용한 연구는 미비한 실정이다. Dell'Amico et al.[7]와 Sun et al.[23]은 TS기법을 이용하여 부품의 총처리시간을 최소로 하는 job shop일정계획방법을 제시하였다. Widmer[24]는 TS를 이용하여 tool제약을 고려한 일정계획 문제를 다루고 있으나 단일공정인 경우만을 연구하였다. 총처리시간 최소화는 모든 부품의 처리를 완료하는 최소 시간을 구하는 의미를 갖지만 각 부품의 처리시간을 고르게 하지는 못한다. 평균처리시간(mean flowtime)은 각 부품의 완료시간의 평균으로 공장의 혼잡도와 재공재고의 수준을 나타내는 척도로서 사용된다[1]. 따라서 본 연구에서는 총처리시간 또는 평균처리시간 최소화의 목적에 사용 가능한 TS기법을 연구하고자 한다.

JSP에 TS를 적용함에 있어서 탐색성능을 향상시킬 수 있는 이웃해 생성방법과 TS의 파라메터에 대한 연구가 요구된다. 부품의 총처리시간과 평균처리시간의 최소화를 목적함수로 갖는 JSP의 최적해는 active일정 집합에 존재[1]하기 때문에, 이 집합만을 탐색하기 위하여 생성된 이웃해의 active를 유지시키는 방법이 개발되어야 한다. 또한 TS기법에서 좀 더 좋은 해를 얻기 위해서는 해공간을 다양하게 탐색하여야 한다.

Job shop일정계획을 위한 TS기법을 개발하기 위하여, 본 연구에서는 TS에서 이웃해를 생성하는 여러 이동방법을 제시하고, 가공순서의 변경(이동)에 따른 재일정계획을 위하여 현재의 일정으로부터 효율적으로 active일정을 재생성하는 방법을 개발한다. 또한 JSP에 적합한 단순TS기법과 TS의 다양화 기법들을 제안한다. 실험을 통하여 총처리시간 또는 평균처리시간의 최소화를 목적으로 하는 job shop일정계획에 적합한 tabu속성, tabu목록크기 등에 관하여 연구하고, 제안한 여러 TS기법들을 비교분석하며 또한 기존의 할당규칙들과의 알고리듬 성능을 비교분석한다.

본 연구의 구성은 다음과 같다. 제 2장에서 job shop일정계획과 TS기법을 소개하고, 제 3장에서는 TS의 이동방법과 active재일정계획 방법에 의한 이웃해 생성방법을 제안하며, 제 4장에서는 TS에 의한 job shop일정계획절차를 제시하고 해 공간의 탐색을 다양화하는 기법을 제안한다. 제 5장에서는 실험을 통하여 tabu파라메터와 제안한 이동방법을 비교분석하고, 개발한 기법을 기존의 할당규칙과 비교한다. 제 6장은 결론으로 구성되어 있다.

2. Job Shop 일정계획과 Tabu Search

본 연구에서 다루는 JSP은 아래의 가정을 갖는다.

(1) 한 부품을 생산하기 위한 공정간의 가공순서는 제약되지만 서로 다른 부품의 공정간의 가공순서는 제약되지 않는다.

(2) 공정을 가공하는 기계는 특정한 하나의 기계로 주어진다.

(3) 기계는 어떤 가공 시점에 하나의 공정만을 가공한다.

(4) 기계가 어떤 공정을 가공하는 도중에 다른 공정이 그 기계에서 가공을 시작할 수 없다.

3. 이웃해 생성

본 장에서는 JSP에 적합한 이웃해의 생성 방법을 개발하기 위하여 각 공정의 삽입이동 범위와 삽입위치를 정의하고, 이로 부터 공정의 이동방법을 제안한다. 그리고 공정의 이동으로 변경된 가공순서를 가능한 유지하는 active일정계획을 효율적으로 다시하는 방법을 제안한다. 사용되는 기호는 아래와 같다.

표 3.1 기호 정의

기호	정의
(i,j) :	부품 i의 j번째 공정.
$t^s(i,j)$:	(i,j)의 일정계획된 시작시간.
$t^f(i,j)$:	(i,j)의 일정계획된 완료시간.
mc(i,j) :	(i,j)를 가공하는 기계.
NSO :	재일정계획을 필요로 하는 공정들의 집합.
FJ :	NSO의 공정중에서 직선행공정이 NSO 집합에 포함되어 있지 않은 공정의 집합, 즉 FJ={(i,j) | (i,j)∈NSO and (i,j-1)∉NSO} 이다.
pt_{ij} :	(i,j)의 가공시간.
σ_{ij} :	공정 (i,j)∈FJ의 가공을 가장 빨리 시작할 수 있는 시간.
τ_{ij} :	공정 (i,j)∈FJ의 가공을 가장 빨리 완료할 수 있는 시간, 즉 $\tau_{ij}=\sigma_{ij}+pt_{ij}$

3.1 이동방법

JSP에서 이웃해를 구하는 쉬운 방법은 현재의 해(일정)에서 임의의 공정을 선택하여 그 공정을 임의의 다른 위치로 삽입하는 방법이다. 이때 삽입이동으로 인하여 변경된 기계의 가공순서를 유지하는 가능이웃해는 모든 공정을 left-shift하는 방법으로 구할 수 있다. 그러나 이와 같은 방법은 semiactive일정을 생성할 뿐만 아니라, 모든 공정을 삽입이동함으로써 계산시간이 많이 소요되게 된다.

따라서 본 연구에서는 이웃해의 효율적 탐색과 공정이동에 따른 효과적인 재일정계획을 위하여 각 공정의 삽입이동범위를 구하고, 이로부터 이동가능한 삽입공정과 그 공정의 삽입위치를 제한하는 방법을 제시하고자 한다. 먼저 공정의 삽입이동범위를 구하기 위하여 좌측한계와 우측한계를 다음과 같이 정의한다.

좌측한계: 공정의 좌측한계는 그 공정이 부품의 첫 번째 가공공정이면 시간을 0으로 두고, 그렇지 않으면 그 공정의 직선행공정의 완료시간으로 둔다.

우측한계: 공정의 우측한계는 그 공정이 부품의 마지막 가공공정이면 총처리시간으로 두고, 그렇지 않으면 그 공정의 직후행공정의 시작시간으로 둔다.

본 연구에서는 삽입이동범위를 공정의 좌측한계와 우측한계를 이용하여 다음 세가지로 분류한다.

삽입이동범위 1: 공정의 좌측한계에서 우측한계까지의 범위

삽입이동범위 2: 공정의 좌측한계에서 그 공정의 시작시간까지의 범위

삽입이동범위 3: 공정의 시작시간에서 그 공정의 우측한계까지의 범위

또한 이동공정은 아래와 같이 정의한다.

이동공정: 이동공정은 어떤 공정을 가공하는 기계상에서 그 공정의 삽입이동범위에서 가공이 완료되는 다른 공정이 있는 공정으로 한다. 그리고 이동공정의 집합을 SMO로 둔다.

삽입이동범위 이내에서 가공이 완료되는 공정들의 가공순서를 SI로 두자. 가공순서 SI에서 이동공정을 현재와 다른 모든 위치로

삽입이동시킨다면, 이때 발생하는 삽입이동의 수는 SI에 있는 공정의 수에서 하나를 뺀 수가 된다. 삽입이동의 총수는 모든 이동공정에 대해 삽입이동의 수를 합한 것이 된다. 가공공정의 수가 많아지면 이웃해를 생성하는 데 많은 시간이 소요되므로 효과적으로 삽입이동의 수를 좀 더 제한하는 방법을 고려할 수 있다. 그러나 삽입이동의 수를 제한하는 데 있어서 임의로 총 삽입이동의 수를 감소하면 탐색능력을 개선시킬 수 없다[4]. 이동공정을 가공순서 SI내에 있는 현재 위치와 다른 모든 위치에 삽입하는 방법 이외에 여러 삽입이동방법을 고려할 수 있다. 본 연구에서는 이동공정의 삽입방법은 아래와 같이 네가지로 분류 한다.

삽입방법 A: 삽입이동범위에 있는 현재의 가공순서 SI에서 이동공정을 현재와 다른 모든 위치로 삽입이동하는 방법

삽입방법 B: 삽입이동범위에 있는 현재의 가공순서 SI에서 이동공정을 가장 앞과 가장 뒤의 두곳으로 삽입이동하는 방법

삽입방법 C: 삽입이동범위에 있는 현재의 가공순서 SI에서 이동공정을 가장 앞으로 삽입이동하는 방법

삽입방법 D: 삽입이동범위에 있는 현재의 가공순서 SI에서 이동공정을 가장 뒤로만 삽입이동하는 방법

삽입이동의 범위와 방법의 예로써, 그림 3.1에서 삽입이동범위를 삽입이동범위1로 두면, 공정 x의 좌측한계와 우측한계는 각각 직선행공정(공정(x-1))의 완료시간과 직후행공정(공정(x+1))의 시작시간이 되고, 공정 x는 이동공정이 된다. 이때 SI=(c d e x f g)가 되고 이동공정 x를 삽입방법 A로 이동시킨다면 삽입이동은 (x c d e f g), (c x d e f g), (c d x e f g), (c d e f x g), (c d e f g x)의 5가지가 된다.

공정을 이동시키기 위하여 먼저 삽입이동범위를 정의하고, 현재의 일정으로 부터 각 공정의 삽입이동범위를 구하여, 모든 이동공정을 구하고 이들의 집합을 SMO로 둔다. SMO의 모든 공정에 대하여 각각 삽입이동시켜 이 공정을 가공하는 기계의 가공순서를 변경한다. 이 절차를 단계별로 보면 아래와 같다.

〈삽입이동절차〉

단계 1. 삽입이동범위를 구하고, 현재의 일정으로 부터 이동공정을 모두 구하여 SMO로 둔다.

단계 2. SMO= ϕ이면 끝낸다. 그렇지 않으면 임의의 공정 $(i,j)\in$SMO를 선택하고, SMO에서 공정 (i,j)를 제거한다.

단계 3. mc(i,j)의 가공순서 중에서 삽입이동범위의 가공순서 SI에 대해 이동공정(i,j)를 삽입 방법에 의해 이동하여 변경한다.

단계 4. 단계 2.로 간다.

위 절차에서 단계 1.의 삽입이동범위는 앞에서 다룬 삽입이동범위1, 2, 3 중에서, 그리고 단계 3의 삽입방법을 삽입방법A, B, C, D 중에서 각각 하나를 선택하여 사용한다. 위 단계 1에서 삽입이동범위에 따라 이동공정이 결정되고, 단계 3의 이동공정의 삽입방법에 따라 하나의 이동공정으로 부터 생성되는 이웃해의 수가 달라지게 된다.

본 연구에서 이동방법은 삽입이동범위와 이동공정의 삽입방법에 따라 표 3.2와 같이 여섯가지로 분류한다.

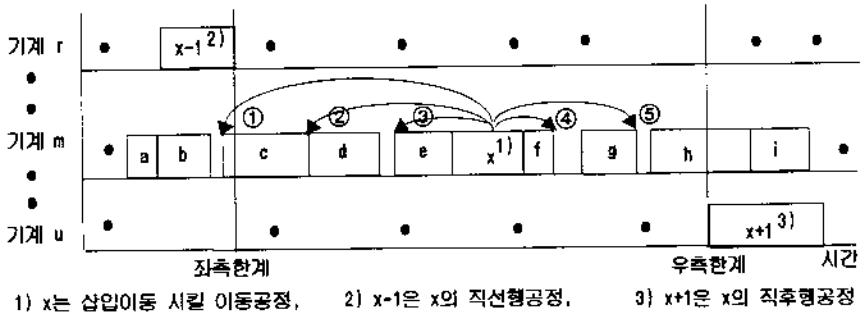


그림 3.1 삽입이동범위와 이동공정

표 3.2 공정의 삽입이동방법

이동방법	1	2	3	4	5	6
삽입이동범위	1	2	3	1	2	3
삽입방법	A	A	A	B	C	D

이동방법에 있어서 이동공정의 수와 공정의 삽입이동으로 생성되는 이웃해의 수는 삽입이동범위가 넓을수록, 그리고 삽입이동범위 내에서 이동회수가 많을수록 많게 된다. 따라서 이동방법1이 다른 이동방법에 비하여 가장 많은 이웃해를 생성하게 된다. 이동방법4는 이동방법1에 비하여 이웃해의 탐색강도는 낮으나 해의 이동에 소요되는 시간이 짧아 해 공간을 다양하게 탐색한다는 장점을 갖는다. 일정 시간내에서 해를 탐색하는 경우, 이웃해의 탐색 강화(intensification)와 탐색공간의 다양화(diversification)를 잘 조화시켜야 한다. 이들 이동방법에 대한 해의 효율과 계산시간은 제 5장에서 실험을 통하여 분석한다.

3.2 Active 재일정계획 방법

앞에서 제시한 삽입이동방법에서 이동공정을 이동한 후, 변경된 공정순서를 유지하면서 각 공정이 가장 빨리 시작하는(공정을 left-shift 시킨) 일정은 active일정을 보장하지 못한다. 따라서 active일정을 위한 재일정계획이 필요하게 된다.

Sun et al.[23]은 Baker[1]가 제시한 active 일정 생성방법을 이용하여 공정순서가 변경된 후, 변경된 공정순서를 가능한 유지하는 재일정계획 방법을 제안하고 있으나, 이 방법은 모든 공정에 대하여 새로이 일정계획을 하고 있다. 본 연구에서 제시하는 active재일정계획 방법의 기본 착상은 이동공정의 삽입으로 active일정이 유지되지 못하는 공정들 만을 재일정계획하자는 것이다. 이렇게 함으로써 계산시간을 줄이고자 한다.

재일정계획을 위하여 아래와 같이 공정을 SO와 NSO의 집합으로 구분한다.

SO: 재일정계획을 하지 않는 공정의 집합으로,

(1) 이동공정이 현 일정보다 앞으로 삽입이동되는 경우에는 이동공정의 직선행공정의 완료시간 이전에 가공이 완료되는 공정들의 집합이고,

(2) 이동공정이 현 일정보다 뒤로 삽입이

동되는 경우에는 삽입이동 전의 이동공정의 시작시간 이전에 가공이 완료되는 공정들의 집합이다.

NSO: 재일정계획을 필요로하는 공정들로, SO에 포함되지 않는 공정들의 집합이다.

본 연구에서는 TS의 초기가능해로 active일정을 사용한다. 이동공정이 현일정보다 앞으로 삽입이동되는 경우, 삽입이동전의 상태가 active일정이고 이동공정이 삽입범위 이내에서 이동하므로 SO에 있는 공정들의 현 일정은 active일정이 된다. 마찬가지로 이동공정이 현일정보다 뒤로 삽입이동되는 경우에도 SO에 있는 공정들의 현 일정은 active일정이 된다.

따라서 본 연구에서는 현재의 일정에서 삽입이동으로 인하여 active일정이 유지되는 공정들을 찾아 이들은 현재의 일정을 그대로 따르고, 변하는 공정의 집합 NSO에 속하는 공정들 만을 가능한 현재 기계에 할당된 가공순서로 active일정을 생성하고자 한다. 제안된 재일정방법은 Baker[1]의 active생성방법을 변형하여 사용한다.

재일정계획의 절차를 아래와 같다.

〈재일정계획 절차〉

단계 1. 모든 공정을 SO와 NSO로 분류한다.

단계 2. NSO로 부터 FJ를 구하고, SO의 부분일정으로 부터 $(i,j)\in FJ$ 의 σ_{ij}, τ_{ij}를 구한다.

단계 3. $\tau^*=\min_{(i,j)}\in FJ\{\tau_{ij}\}$를 구하고, τ^*인 공정을 가공하는 기계 mc*를 구한다. τ^*인 (i,j)가 둘이상이면 임의로 선택한다.

단계 4. 공정 $(i,j)\in FJ$ 이고 $\sigma_{ij} < \tau^*$인 (i,j)중에서, 기계 mc*에서 가장 빠른 가공순서를 갖는 공정 (s,t)를 선택한다.

단계 5. $t^s(s,t)=\sigma_{st}$로 두고, $t^f(s,t)$를 계산한다. (s,t)를 NSO와 FJ에서 삭제한다. 공정 (s,t)가 부품 s의 마지막 공정이면 단계 6으로 가고, 그렇지 않으면 FJ에 (s,t+1) 을 넣는다.

단계 6. NSO=ϕ이면 종료하고, 그렇지 않으면 단계 3.으로 간다.

위의 절차 중 단계 3에서 공정 $(i,j)\in FJ$을 할당가능공정으로 함으로써 부품 가공의 순서제약을 만족시키고, 단계 4에서 mc*에서 가장 빠른 가공순서를 갖는 공정(s,t)를 선택함으로써 현재의 일정을 가능한 유지하도록 한다. 제안된 절차는 계산시간을 줄이기 위하여 재일정계획이 필요한 공정들만 다룬다는 점에서 Sun et al.[23]의 방법과 차이가 있다.

예로써 그림 2.1에서 삽입이동방법에 의해 기계 2에서 (1,2)가 4번째의 가공순서에서 (3,2)의 앞인 3번째의 가공순서로 삽입이동되면 이웃해는 그림 3.2의 (a)와 같이 된다. 이때 재일정계획 절차 단계 1에서 NSO={(2,2),(1,2),(4,2),(3,3),(3,2),(1,3),(4,3),(2,3)}이 되고, 단계2에서 FJ={(2,2),(1,2),(4,2),(3,2)}이 된다. 단계 3에서 $\tau^*=7$를 갖는 공정은 (1,2),(4,2)이고 이중에서 임의로 (1,2)를 선택하면 (1,2)의 가공기계는 기계2가 된다. 단계 4에서 $(i,j)\in FJ$ 이고 $\sigma_{ij} < \tau^*$을 만족하고 기계2에서 가장 빠른 가공순서를 갖는 공정 (1,2)를 선택한다. 단계 5에서 $t^s(1,2)=4$로 부터 $t^f(1,2)=7$이 계산된다. NSO={(2,2),(4,2),(3,3),(3,2),(1,3),(4,3),(2,3)}, FJ={(2,2),(1,3),(4,2),(3,2)}로 두고

단계3으로 가서 절차를 반복하면 그림 3.2의 (b)와 같은 active일정이 생성된다.

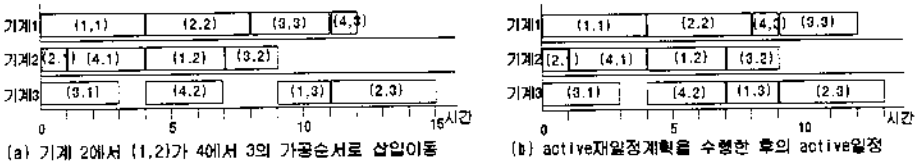


그림 3.2 Active 재일정계획 생성의 예제

4. Job Shop 일정계획을 위한 Tabu Search 기법

4.1 단순 Tabu Search 기법

2장에서 제안한 단순TS기법을 job shop일정계획에 적용한다. 먼저 단계 1(초기화)에서 tabu속성, tabu목록의 크기는 5장에서 실험에 의해 비교분석한다. 열망수준은 현재까지 얻은 가장 좋은 해로 하고, 종료기준은 최대이동회수 또는 계산시간 등으로 한다. 그리고 초기해는 임의의 active일정을 생성하여 사용한다.

단계 2의 이웃해 생성은 제 3장에서 다룬 삽입이동방법과 재일정계획방법에 의하여 tabu이동이 아니거나 tabu이동이지만 열망수준을 만족하는 active일정을 생성한다. 이렇게 각 이동에 의해 생성된 active 일정들이 현 일정의 이웃해 집합이 된다. 이와 같이 단계 1과 단계 2에서 각각 JSP에 적절한 초기화와 이웃해를 생성하고 나면, 그 다음 단계들은 2장에서 제시한 단순TS절차의 단계 3, 단계 4, 단계 5와 동일하다.

4.2 다양화 기법

해공간이 좁으면 앞에서 제시한 단순TS기법만으로 좋은 해를 찾을 수 있으나 해공간이 넓은 어려운 문제에서는 흔히 좋은 해를 찾지 못하게 된다. 해의 개선을 위하여 해공간의 탐색이 다양하게 이루어져야 한다.

TS기법에서 해공간의 탐색을 다양화하는 여러 기법[11, 13, 15]들이 제안되었다. 가장 단순하게 탐색을 다양화하는 방법은 단순TS에서 종료기준(최대이동회수 또는 계산시간이나 해가 개선되지 않는 이동회수 등을 사용)을 만족할 때, 임의의 새로운 초기해로 tabu크기나 종료기준 등을 변경하여 단순TS를 반복하는 방법이다[20]. 이와 같이 임의로 탐색 영역을 찾지 않고 과거의 이동에 관한 정보를 모두 기억하는 장기기억을 이용하는 방법이 제안되었다[11, 15]. 이 방법은 새로운 초기해를 구할 때 장기기억을 이용하여 가능한 지금까지 탐색하지 않는 영역을 찾는 것이다. TS기법에서 이동에 따른 정보를 장기기억함수에 저장시키면서 이동하여 미리 지정된 이동회수나 계산소요시간 또는 최선해가 개선되지 않는 이동회수를 만족하면 장기기억함수를 이용하여 새로운 초기해를 구하고 이 초기해로 부터 이동을 다시 시작한다. 이와 같은 과정을 종료조건이 만족될 때까지 반복한다.

본 연구에서는 장기기억함수가 이동하는 공정만을 기억하는 경우와, 이동하는 공정과

그 이동위치를 함께 기억하는 경우를 다룬다. 즉, 본 연구에서는 장기기억함수의 형태로 LTM = (i, j)인 이차원 배열과 LTM = (i, j, v)인 삼차원 배열인 두 경우를 비교분석한다. 여기서 (i, j)는 i부품의 j번째 공정, (i, j, v)는 공정 (i, j)가 가공되는 기계에서 v번째 가공되는 것을 나타낸다. 각 공정의 가공기계는 특정한 하나의 기계로 주어지므로 기계에 대한 배열은 사용하지 않아도 된다. 초기에는 LTM의 모든 원소의 값을 영으로 둔다. 이동하는 공정만을 기억하는 경우에는, 이동하는 공정이 공정 (i, j)이면 $LTM_{ij} = LTM_{ij} + 1$로 두고, 이동하는 공정과 그 이동위치를 함께 기억하는 경우에는 공정 (i, j)가 가공기계에서 v번째 가공된다면, $LTM_{ijv} = LTM_{ijv} + 1$로 변화시킨다. 이와 같은 방법으로 해가 이동될 때마다 장기기억함수를 변화시킨다. 즉, 장기기억함수는 공정의 이동회수 또는 공정과 그 위치로의 이동이 이루어진 회수로 표현된다.

앞에서 언급했듯이 장기기억함수의 사용목적은 가능한 지금까지 탐색하지 않은 영역을 찾는 데 있다. 따라서 현재까지 찾은 가장 좋은 해를 현재해로 하여 이동공정을 구하고, 장기기억함수에서 가장 작은 값을 갖는 이동공정(과 이동위치로)을 삽입하여 초기해를 구한다. 또한 이동공정 중에서 장기기억함수의 값이 최소인 하나의 공정만을 선택하지 않고, 이 과정을 미리 지정한 회수만큼 반복하여 초기해를 구할 수 있다. 이때 장기기억함수에서 최소값이 둘 이상이면 이 초기해를 구하는 과정에서 가공순서가 변하지 않는 기계에서 가공되는 이동공정을 선택한다.

5. 비교분석

이 장에서는 JSP에서 목적함수가 총처리시간의 최소화 또는 평균처리시간의 최소화인 두 종류의 문제에 대해 각각 실험을 통하여 제안한 기법들을 비교분석하고자 한다.

실험에 사용된 문제는 Muth와 Thompson [18]의 6(부품 수)×6(공정 수, 기계 수), 10×10과 20×5를 벤치마크(benchmark)문제로 사용하였다. 이 문제들은 그동안 JSP에 관한 많은 연구에서 비교분석 문제로 사용되어 왔다. 분석의 정확성을 높이기 위하여, 각 문제에 대하여 임의로 생성한 10개의 active일정을 초기해로 하여 얻은 결과의 평균을 사용하였다. 실험에 사용한 컴퓨터는 CPU 60Mhz의 펜티엄이며 계산시간은 CPU시간이다.

5.1 Tabu목록크기와 Tabu속성

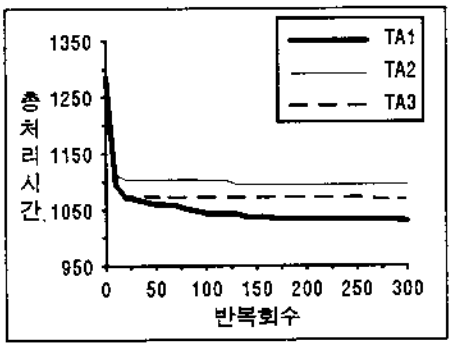
Tabu목록크기와 tabu속성을 분석하기 위하여 세개의 벤치마크 문제에 대해 tabu목록크기를 변경해 가며 세가지 tabu속성과 여섯가지 이동방법에 대해 반복회수 250회의 종료조건을 사용하여 실험하였다.

표 5.1은 각 문제에서 이동방법에 따라 tabu속성과 tabu목록크기를 변화하면서 실험하여, 이동방법에 따른 이동공정 수의 평균, 이웃해 수의 평균과 250회 반복에 소요되는 계산시간을 나타낸 것이다.

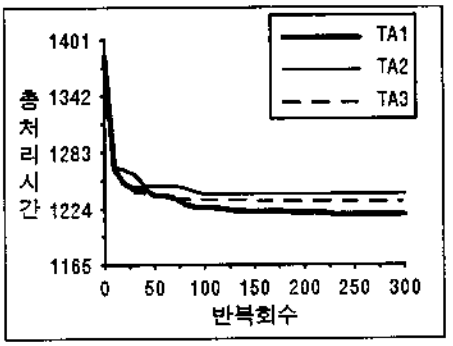
위의 실험 분석결과, tabu목록크기는 tabu속성, 이동공정의 수, 이웃해의 수 등에 영향을 받는 것으로 나타났다. 목록크기는 Tabu속성이 TA1인 경우에는 이동공정의 수에 크게 영향을 받지만, tabu속성이 TA2와 TA3인 경우에는 반복당 생성되는 이웃해의 수에 영

표 5.1 평균 이동공정 수와 이웃해 수

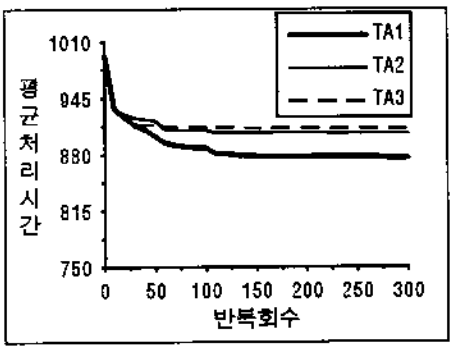
이동방법 \ 문제	6×6 문제			10×10 문제			20×5 문제		
	평균 이동 공정 수	평균 이웃해 수	계산소요 시간(초)	평균 이동 공정 수	평균 이웃해 수	계산소요 시간(초)	평균 이동 공정수	평균 이웃해 수	계산소요 시간(초)
1	19	33	10	64	140	161	90	535	980
2	16	25	8	45	99	121	79	292	541
3	7	10	2	22	39	44	58	248	455
4	19	24	8	64	72	92	90	132	252
5	16	16	5	45	45	60	79	79	140
6	7	7	2	22	22	25	58	58	86



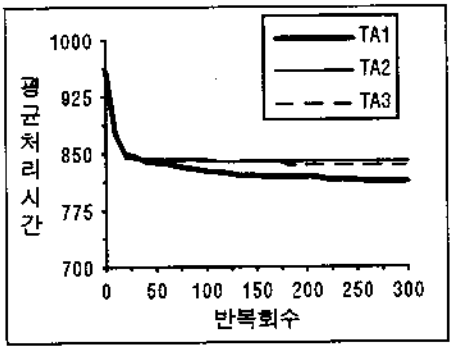
(a) 총처리시간 최소화:10×10문제



(b) 총처리시간 최소화:20×5문제



(c) 평균처리시간 최소화:10×10문제



(d) 평균처리시간 최소화:20×5문제

그림 5.1 Tabu속성의 비교

향을 더 받는 것으로 나타났다. TA1의 경우, 이동방법1, 2, 3은 평균 이동공정 수의 30%~40% 정도, 이동방법 4, 5, 6은 평균 이동공정 수의 15%~20%정도의 tabu목록크기가 효

과적인 것으로 나타났다. 이동방법4, 5, 6이 이동방법1, 2, 3보다 tabu목록크기가 작은 것은 생성되는 이웃해의 수가 적은 데 기인한 것으로 보인다. 그리고 tabu속성이 TA2인 경우에는 평균 이웃해 수의 15%~25% 정도, TA3인 경우에는 평균 이웃해 수의 20%~30%정도의 tabu목록크기가 좋은 것으로 나타났다.

Tabu목록크기가 너무 작거나 크지 않고 적정 범위내에서 변화하는 경우, 이웃해의 탐색 강화와 해 공간의 탐색 다양화가 상호 상쇄되어 이에 크게 영향을 받지 않는 것으로 나타났다.

Tabu속성을 비교하기 위하여, 앞에서 제시한 tabu목록크기(TA1은 평균 이동공정 수의 35%, TA2는 평균 이웃해 수의 20% , TA3는 평균 이웃해 수의 25%)와 이동방법1을 사용하여 각 초기해에 대해 300회 반복 실행한 결과의 평균을 구하였다. 실험결과는 그림 5.1과 같았다. 해의 효율면에서 tabu제약이 가장 강한 TA1이 가장 좋은 결과를 나타냈다. 다른 이동방법에 대한 tabu속성의 실험결과도 거의 유사하였다.

따라서 다음 절의 이동방법과 다양화기법의 비교분석 실험에서 tabu속성은 TA1을 사용하고, tabu목록크기는 이동공정 수의 약 35%을 사용하기로 한다.

5.2 이동방법의 비교분석

이동방법을 비교하기 위하여 각 이동방법에 대하여 10개의 active일정을 초기해로 사용하여 6×6 문제는 20초동안, 10×10과 20×5 문제는 300초 동안 단순TS기법으로 반복 실험하였다. 표 5.2와 그림 5.2는 각 이동방법에 따른 실험결과를 나타낸다. 표 5.2에서 가장 좋은 해는 각 초기해로 부터 구한 10개의 해 중에서 가장 좋은 해를 나타내며, 6×6 문제에서 괄호 안의 값은 10번 실험에서 최적해를 얻은 회수이고, 개선율은 (초기해 평균 - 해 평균)/초기해 평균으로 구하였다.

이동방법의 성능에 있어서 삽입이동범위1을 갖는 이동방법1과 4가 가장 좋았고, 삽입이동범위2를 갖는 이동방법2와 5는 다음으로 좋았으며, 삽입이동범위3을 갖는 이동방법3과 6이 가장 나쁜 결과를 보였다. 이로부터 이동공정의 삽입위치가 해의 효율에 영향을 크게 준다는 것을 알 수 있으며, 이동공정이 앞으로만 이동하거나 또는 뒤로만 이동하는 것보다는 앞뒤로 이동하는 것이 좋은 해를 구할 수 있음을 알 수 있다.

앞에서 언급 했듯이 이동방법1과 4는 삽입이동범위는 같으나, 이웃해를 구하는 삽입방법에 있어서 이동방법1은 이동공정이 삽입범위의 모든 위치로 이동하는데 반하여, 이동방법4는 삽입범위의 가장 앞과 뒤의 두 위치로만 이동한다. 따라서 이동방법1은 이웃해의 탐색이 강화되는 반면에 이동에 필요한 계산시간이 많이 소요되고, 이동방법4는 이동방법1에 비해 이동에 필요한 계산시간은 적게 소요되나 이웃해 탐색이 약화된다(표 5.1 참조). 표 5.2와 그림 5.2로 부터 총처리시간의 최소화와 평균처리시간의 최소화 문제에서 크기가 작은 문제에서는 이동방법1이 이동방법4보다 더 좋고, 비교적 크기가 큰 문제에서는 이동방법 4가 이동방법1보다 약간 더 좋음을 알 수 있다.

표 5.2 삽입이동방법의 실험결과

(a) 총처리시간 최소화

문제 / 이동 방법	6 × 6문제			10 × 10문제			20 × 5문제		
	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차
1	55(6)	56(0.253)	1.90	969	1021(0.206)	28.97	1191	1230(0.112)	26.23
2	55(4)	58(0.227)	3.41	975	1024(0.204)	25.83	1203	1242(0.103)	22.16
3	58(0)	64(0.147)	3.44	1025	1100(0.145)	64.77	1234	1263(0.088)	25.89
4	55(2)	57(0.240)	1.43	966	1017(0.210)	22.13	1180	1224(0.116)	22.80
5	55(1)	58(0.227)	1.73	1050	1090(0.152)	36.99	1217	1260(0.090)	27.20
6	58(0)	64(0.147)	3.44	1014	1105(0.141)	49.25	1222	1252(0.096)	17.68
초기해평균	75			1286			1385		
최적해	55			930			1165		
계산시간	20초			300초			300초		

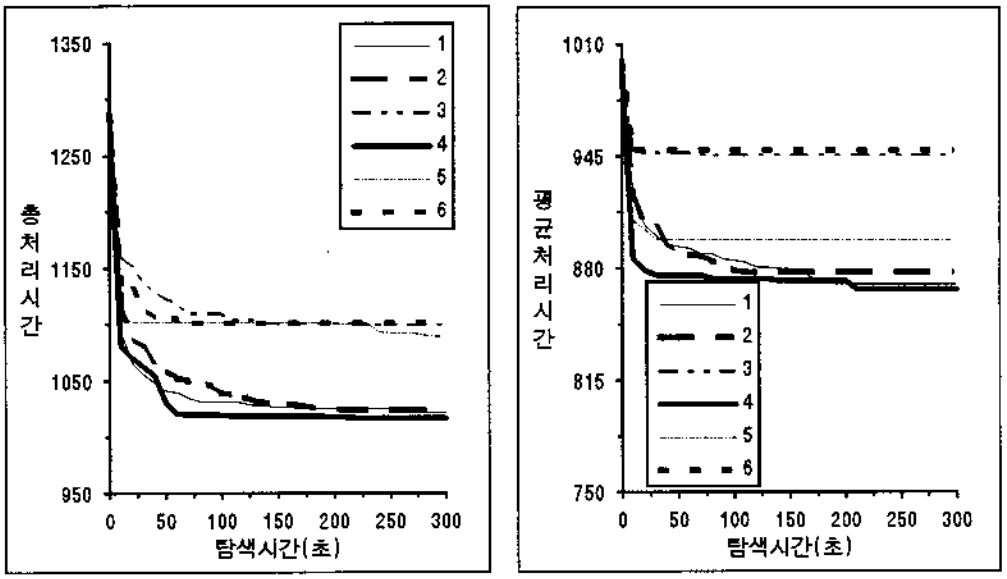
(b) 평균처리시간 최소화

문제 / 이동 방법	6 × 6문제			10 × 10문제			20 × 5문제		
	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차
1	48	49(0.125)	2.20	850	871(0.124)	19.70	793	822(0.144)	15.46
2	47	49(0.125)	1.23	842	878(0.117)	35.13	787	823(0.143)	17.81
3	50	54(0.036)	2.67	904	946(0.048)	36.93	818	857(0.107)	27.47
4	48	49(0.125)	1.03	850	868(0.127)	11.18	783	825(0.141)	23.43
5	48	49(0.125)	0.92	814	897(0.098)	48.91	764	822(0.144)	30.99
6	50	54(0.036)	2.83	905	949(0.045)	35.89	822	857(0.107)	21.02
초기해평균	56			994			960		
계산시간	20초			300초			300초		

5.3 다양화기법의 비교분석

TS에서 해공간의 탐색을 다양화하는 방법으로 아래 세가지 방법을 실험하였다. 이들 실험에서 종료조건은 계산시간으로 600초를 사용하였다. 첫째 방법은 단순히 초기해를 달리하여 단순 TS를 반복하는 방법으로, 하나의 초기해에 대해 일정회수(실험에서는 250회)만큼 이동한 후 새로운 초기해로 다시 시작하는 과정을 종료조건을 만족할 때까지 반복하는 방법이다.

둘째 방법과 세째 방법은 장기기억을 사용하는 방법으로, 둘째 방법은 장기기억으로 이동하는 공정을 기억하는 방법이고, 세째 방법은 이동하는 공정과 그 위치를 함께 기억하는 방법이다. 이들 방법은 장기기억함수만 다를 뿐 그 절차는 아래와 같다. TS에서 임의의 초기해로 부터 시작하여 일정회수(p)를 이동한 후 그 이후로 해의 개선이 일정회수



(a) 총처리시간 최소화:10×10문제 (b) 평균처리시간 최소화:10×10문제

그림 5.2 삽입이동방법의 비교

(q)의 이동동안 이루어지지 않으면 지금까지의 가장 좋은 해를 현재해로 하여 이 해로부터 초기해를 구하여 다시 시작하는 방법으로 종료조건이 만족될 때까지 반복하는 방법이다. 여기서 초기해는 장기기억함수에서 가장 작은 값을 갖는 이동공정을 삽입이동하여 이웃해를 구하고 또 이를 현재해로 하여 다시 삽입이동하는 방법으로 일정회수(r)만큼 행하여 구하였다. 본 연구에서는 p, q, r을 변화하면서 실험하여 좋은 결과를 보여준 p=100, q=20, r=7을 사용하였다. 그리고 단순TS와 TS다양화 방법과의 성능비교를 위하여 앞절의 실험에서 좋은 결과를 보여준 이동방법 4를 600초 동안 실험하였다.

표 5.3과 그림 5.3은 다양화기법의 10회 반복 실험한 결과를 나타낸 것이다. 여기서 단순TS는 이동방법4를 사용한 단순TS기법을 나타내고, 단순반복은 초기해를 달리하여 반복한 첫째 방법을, 장기기억1은 이동하는 공정을 기억하는 둘째 방법을, 장기기억2는 이동하는 공정과 그 위치를 기억하는 세째 방법을 나타낸다. 개선율은 (초기해 평균 - 해 평균)/초기해 평균으로 구하였다.

실험결과, 표 5.3과 그림 5.3으로 부터 총처리시간과 평균처리시간을 최소화하는 두 문제 모두에서 단순TS기법은 300초 이후 해가 거의 개선되지 않음을 알 수 있다. 초기해를 임의로 반복하는 단순반복은 단순TS기법보다는 해를 더 개선하였으나 장기기억1과 장기기억2의 방법보다는 좋지 않았다. 특히 장기기억의 효과는 총처리시간 최소화 문제보다는 평균처리시간 최소화 문제에서 크게 나타났다. 이는 본 연구에서 제시한 이동방법과 다양화기법은 총처리시간 최소화 문제

표 5.3 다양화기법의 실험결과

(a) 총처리시간 최소화

문제 / 방법	10×10 문제						20×5 문제					
	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차
단순TS	966	1017(0.210)	22.13	966	1015(0.211)	24.07	1180	1224(0.116)	22.80	1180	1222(0.118)	27.81
단순반복	983	1002(0.221)	10.44	983	998(0.224)	5.26	1191	1238(0.106)	21.22	1191	1232(0.110)	18.81
장기기억1	972	994(0.227)	21.21	958	987(0.233)	20.54	1198	1236(0.108)	21.64	1180	1216(0.122)	24.10
장기기억2	967	1007(0.217)	22.45	967	993(0.229)	18.62	1228	1239(0.105)	14.05	1184	1218(0.121)	17.17
계산시간	300초			600초			300초			600초		
초기해평균	1286						1385					

(b) 평균처리시간 최소화

문제 / 방법	10 × 10 문제						20 × 5 문제					
	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차	가장 좋은해	해평균 (개선율)	표준 편차
단순TS	850	868(0.127)	11.18	841	866(0.129)	14.23	738	825(0.141)	23.43	783	819(0.147)	23.52
단순반복	833	866(0.129)	16.87	829	851(0.144)	14.31	780	801(0.166)	14.94	774	788(0.179)	10.67
장기기억1	773	819(0.176)	33.23	768	801(0.194)	32.70	721	773(0.195)	24.38	709	761(0.207)	27.21
장기기억2	757	800(0.195)	31.35	757	791(0.204)	28.81	732	780(0.189)	27.46	730	757(0.211)	20.17
계산시간	300초			600초			300초			600초		
초기해평균	994						960					

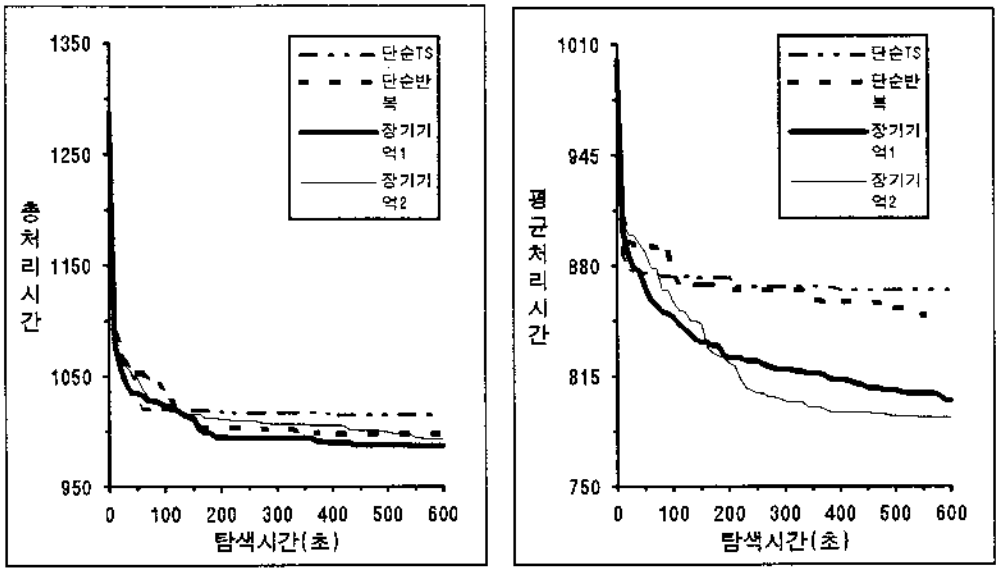
보다는 평균처리시간 최소화 문제에 더 적합함을 보여준다.

장기기억을 이용한 장기기억1과 장기기억2를 비교할 때, 표 5.3과 그림 5.3으로 부터 두 방법간에 해의 효율에 있어서는 큰 차이가 없으나, 총처리시간 최소화 문제에서는 장기기억1이, 평균처리시간 최소화문제에서는 장기기억2가 약간 더 좋았다.

5.4 할당규칙과의 비교

이 절에서는 제안한 TS기법의 성능분석을 위하여 여러 할당규칙에 따른 active일정계획과 비교하였다. 표 5.4는 7종류의 할당규칙과 제안한 TS기법을 비교한 것이다. 여기서 TS기법은 총처리시간 최소화 문제에서는 장기기억1의 방법을, 평균처리시간 최소화 문제에서는 장기기억2의 방법을 사용하여 얻은 결과이다(표 5.3 참조). 할당규칙의 실험은 각 할당규칙에서 공정의 우선순위가 같은 것이 둘 이상 발생하면 이들 공정중에서 임의로 할당하는 방식으로 100개의 해를 구하여 평균을 구하였다.

표 5.4로 부터 제안된 TS절차는 총처리시간 또는 평균처리시간의 최소화 문제에서 할당규칙들보다 월등히 좋은 해를 구함을 알 수 있다. 가장 좋은 해를 구한 할당규칙과 제안된 TS기법을 비교하면 10×10문제에서 총처리시간 최소화문제는 제안된 TS기법이



(a) 총처리시간 최소화:10×10문제 (b) 평균처리시간 최소화:10×10문제

그림 5.3 다양화기법의 비교

표 5.4 할당규칙과의 비교

문제 / 이동방법	10×10 문제		20×5 문제	
	총처리시간	평균처리시간	총처리시간	평균처리시간
Random	1309	998	1402	1002
SPT	1305	991	1504	978
Twork	1474	1032	1500	966
MWKR	1282	983	1525	1043
LWKR	1323	1000	1442	967
MOPNR	1292	997	1518	1045
LOPNR	1318	1001	1443	956
TS기법	987	791	1216	757

SPT: Shortest Processing Time,
TWork: Total Work
MWKR: Most WorK Remaining,
LWKR: Least WorK Remaining
MOPNR: Most OPeratioNs Remaining,
LOPNR: Least OPeratioNs Remaining

MWKR보다 25% 만큼, 평균처리시간 최소화 문제는 SPT보다 23% 개선되었고, 20×5문제에서도 제안된 TS기법이 총처리시간 최소화 문제는 Random보다 21% 만큼, 평균처리시간 최소화문제는 LOPNR보다 23% 정도 개선되었다.

6. 결 론

본 연구에서는 job shop을 위한 TS기법을 개발하고 이를 비교분석하였다. TS기법에서 이웃해의 생성을 위하여 여러 삽입이동방법들을 제안하고, 효율적으로 active재일정계획을 할 수 있는 알고리듬을 개발하였다. 또한 이동방법에 따른 여러 단순TS기법들과 함께, 장기기억을 이용하여 해공간의 탐색을 다양하게 할 수 있는 여러 TS기법을 제시하였다.

제안한 TS기법은 JSP에서 총처리시간을 최소로 하는 문제와 평균처리시간을 최소로 하는 문제에 적용하였다.

총처리시간 최소화문제와 평균처리시간 최소화문제의 jop shop일정계획을 위한 TS기법의 비교분석을 위하여 먼저, 실험을 통하여 tabu 목록크기에 영향을 미치는 요인들을 찾고 JSP에 적합한 tabu목록크기와 tabu속성을 구하였다. 그리고 여러 삽입이동방법과 TS의 다양화 기법들을 해의 효율과 계산시간 면에서 비교분석하였다.

본 연구의 주요 결과는 아래와 같다.

(1) Tabu목록 크기는 tabu속성, 이동공정의 수, 이웃해의 수 등에 영향을 받는 것으로 나타났다.

(2) Tabu속성에서는 삽입이동공정을 속성으로 하는 경우가 삽입이동공정과 그 위치 등을 속성으로 하는 경우보다 더 좋은 결과를 가져왔다.

(3) 삽입이동방법에서는 삽입이동범위를 직선행공정의 완료시간과 직후행공정의 시작시간 범위 내에서 이동공정을 가장 앞과 뒤로 보내는 방법이 계산시간과 해의 효율면에서 가장 좋았다.

(4) TS기법들 간의 비교에서는 장기기억을 이용한 TS기법이 가장 좋았으며, 다음은 초기해를 임의로 달리 하는 단순반복TS기법이 좋았고, 단순TS기법은 가장 좋지 않았다.

(5) 본 연구에서 제시한 TS기법은 총처리시간 최소화 문제보다는 평균처리시간 최소화 문제에 더 적합하였다. 특히 평균처리시간 최소화 문제에서 장기기억을 이용한 TS기법은 타 기법에 비하여 해의 개선효과가 아주 높았다.

(6) 본 연구에서 제안한 TS기법과 기존의 여러 할당규칙들과의 비교에서 제안한 TS기법이 아주 좋은 결과를 보였다.

본 연구에서 제시한 이웃해 생성방법은 JSP를 위한 타 기법에 활용될 수 있다. JSP를 위한 simulated annealing기법에서의 이웃해 생성방법에 그대로 적용될 수 있으며, JSP를 위한 유전알고리듬의 유전 연산자(genetic operators) 개발에도 응용될 수 있다. 또한 이런 해의 이동방법을 약간 변형하면 납기를 갖는 JSP에 활용될 수 있다.

참 고 문 헌

[1] Baker, K., Introduction to Sequencing and Scheduling, Ch.2-8, John Wiley & Sons Inc., New York, 1974.

[2] Barnes, J. W. and Laguna, M., "Solving the Multiple-Machine Weighted Flow Time Problem using Tabu Search", IIE Transactions, Vol. 25, No. 2, pp. 121-128, 1993.

[3] Barnes, J. W. and Laguna, M., "A Tabu Search Experience in Production Scheduling", Annals of operations research, Vol. 41, pp. 141-156, 1993.

[4] Belarmino, A. D., "Restricted Neighborhood in the Tabu Search for the Flowshop Problem", European Journal of Operational Research, Vol. 62, pp. 27-37, 1992.

[5] Blackstone, J. H., Phillips, D. T., and Hogg, G. L., "A State-of-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations", International Journal of

Production Research, Vol. 20, No. 1, pp. 27-45, 1982.

[6] Croce, F. D., Tadel, R., and Volta, G., "A Genetic Algorithm for the Job Shop Problem", Computers & Operations Research, Vol. 22, No. 1, pp. 15-24, 1995.

[7] Dell'Amico and Trubian, M., "Applying Tabu Search to the Job-Shop Scheduling Problem", Annals of operations research, Vol. 41, pp. 231-251, 1993.

[8] Foo, Y. S. and Takefuji, Y., "Stochastic Neural Networks for Solving Job-Shop Scheduling:Part 1. Problem Representation", International joint conference on neural networks Ⅱ, pp. 275-282, 1988.

[9] Fry, T. D., Philipoom, P. R., and Blackstone, J. H., "A Simulation Study of Processing Time Dispatching Rules", Journal of operations management, Vol. 7, No. 4, pp. 77-92, 1988.

[10] Garcia, B. L., Potvin, J. Y., and Rousseau, J. M., "A Parallel Implementation of the Tabu Search Heuristic for Vehicle Routing Problems with Time Window Constraints", Computers & Operations Research, Vol. 21, No. 9, pp. 1025-1033, 1994.

[11] Glover, F., "Tabu Search-Part Ⅰ", ORSA Journal on Computing, Vol. 1, No. 3, pp. 190-206, 1989.

[12] Glover, F., "Tabu Search-Part Ⅱ", ORSA Journal on Computing, Vol. 2, No. 1, pp. 4-32, 1990.

[13] Hubscher, R. and Glover, F., "Applying Tabu Search with Influential Diversification to Multiprocessor Scheduling", Computers & Operations Research, Vol. 21, No. 8, pp. 877-884, 1994.

[14] Hutchison, J. and Chang, Y. L., "Optimal Nondelay Job Shop Schedules", International Journal of Production Research, Vol. 28, No. 2, pp. 245-257, 1990.

[15] Kelly, J. P., Laguna, M., and Glover, F., "A Study of Diversification Strategies for the Quadratic Assignment Problem", Computers & Operations Research, Vol. 21, No. 8, pp. 885-893, 1994.

[16] Laarhoven, P., Aarts, E., and Lenstra, J. K., "Job Shop Scheduling by Simulated Annealing", Operations Research Society of America, Vol. 40, No. 1, pp. 113-125, 1992.

[17] Laguna, M., Barnes, J.W., and Glover, F., "Tabu Search Methods for a Single Machine Scheduling Problem", Journal of Intelligent Manufacturing, Vol. 2, pp. 63-74, 1991.

[18] Muth, J. F. and Thompson, G. L., Industrial Scheduling, Ch15., Prentice Hall Inc., Englewood Cliffs, New Jersey, 1963.

[19] Panwalkar, S. S. and Iskander, W., "A Survey Scheduling Rules", Operations Research, Vol. 25, No. 1, 1977.

[20] Skorin-Kapov, J., "Tabu Search Applied to the Quadratic Assignment Problem", ORSA Journal on Computing,Vol. 2, No. 1, pp. 33-45, 1990.

[21] Skorin-Kapov, J., "Scheduling a Flow-Line Manufacturing Cell: A Tabu Search Approach", International Journal of Production Research, Vol. 31, No. 7, pp. 1721-1734, 1993.

[22] Skorin-Kapov, J., "Extensions of Tabu Search Adaptation to the Quadratic Assignment Problem", Computers & Operations Research, Vol. 21, No. 8, pp. 855-865, 1994.

[23] Sun, D., Batta, R., and Lin, L., "Effective Job Shop Scheduling through Active Chain Manipulation", Computers & Operations Research, Vol. 22, No. 2, pp. 159-172, 1995.

[24] Widmer, M., "Job Shop Scheduling with Tooling Constraints: A Tabu Search Approach", Journal of the Operational Research Society, Vol. 42, No 1, pp. 75-82, 1991.

[25] Yang, T. Y., Ignizio, J. P., and Deal, D. E., "An Exchange Heuristic for Generalized Job Shop Scheduling", Engineering Optimization, Vol. 15, pp. 83-96, 1989.

95년 5월 최초 접수, 95년 7월 최종 수정