IE Interfaces
Vol. 15, No. 4, pp. 338-348, December 2002.

병렬 기계 스케줄링을 위한 제한적 이웃해 생성 방안

신현준[1†] · 김성식[2]

[1]Texas A&M University 산업공학과 / [2]고려대학교 산업시스템정보공학과

A Restricted Neighborhood Generation Scheme for Parallel Machine Scheduling

Hyun Joon Shin[1] · Sung Shick Kim[2]

[1]Department of Industrial Engineering, Texas A&M University

[2] Department of Industrial Systems and Information Engineering., Korea University, Seoul, 136-701

In this paper, we present a restricted tabu search(RTS) algorithm that schedules jobs on identical parallel machines in order to minimize the maximum lateness of jobs. Jobs have release times and due dates. Also, sequence-dependent setup times exist between jobs. The RTS algorithm consists of two main parts. The first part is the MATCS(Modified Apparent Tardiness Cost with Setups) rule that provides an efficient initial schedule for the RTS. The second part is a search heuristic that employs a restricted neighborhood generation scheme with the elimination of non-efficient job moves in finding the best neighborhood schedule. The search heuristic reduces the tabu search effort greatly while obtaining the final schedules of good quality. The experimental results show that the proposed algorithm gives better solutions quickly than the existing heuristic algorithms such as the RHP(Rolling Horizon Procedure) heuristic, the basic tabu search, and simulated annealing.

Keywords: Restricted Neighborhood Generation Scheme, Tabu Search, Parallel Machine Scheduling

1. 서 론

현대에 들어오면서부터 제조업체들은 생산성 향상이나 최대 이윤창출이라는 제조전략에서 고객만족이라는 제조전략으로의 새로운 패러다임(paradigm)을 형성해 가고 있으며, 이러한 패러다임하에서 납기 준수는 기업의 생존과 직결된 중요한 관리목표가 되고 있다. 그러나 제품의 생산방식과 과정이 복잡해지고 다양해지면서 제조업체에서는 생산능력을 효율적으로 사용하는데 어려움을 겪고 있다. 제조공정의 효율성을 방해하는 대표적인 요인들로 순서 의존적인 작업 준비시간(sequence-dependent setup time)과 작업 투입시점(release time)을 들 수 있다. 순서 의존적인 작업 준비시간은 반도체 공정, 자동차 도장 공정 및 플라스틱 사출성형공장 등의 작업현장에 많이 존재하는데, 일반적으로 순서 의존적인 작업 준비시간이 존재하는 경우, 그 스케줄링 결과에 따라 자원의 가동률과 그에 따른 납기준수율이 크게 좌우된다(Kim and Bobrowski, 1994). 작업 투입시점은 일반적으로 BOM(bill of material)상의 하위 부품 조달시점 또는 해당 작업의 전 공정 완료시점에 의해 결정되는데, 작업 투입시점이 존재하는 제조환경에서는 기계의 유휴시간(idle time)을 최소화하기 위해 정확한 스케줄링이 필요하다. 특히 반도체 공정과 같이 높은 설비효율을 요구하는 생산 환경에서는 이러한 요인들을 고려하는 효율적인 스케줄링 방법이 필수적이다.

본 연구에서는 순서 의존적인 작업 준비시간과 작업 투입시

† 연락저자: 신현준, 136-701 서울시 성북구 안암1가 1번지 고려대학교 부설 정보통신기술공동연구소, Fax: 02-929-5888, e-mail: mrshin@korea.ac.kr

2002년 8월 접수, 1회 수정 후 2002년 10월 게재 확정.

점이 존재하는 동적인 환경에서 납기 지연시간(lateness time)의 최대값(이하 L_{max})을 최소화하기 위해 주어진 N개의 작업을 M대의 동일한 병렬기계(identical parallel machines)에 스케줄링하는 알고리듬을 제시한다. 각 작업의 인덱스를 $i(i = 1, \cdots, N)$라고 했을 때, 이들 작업은 각각 작업 투입시점 r_i와 납기(due date) d_i, 그리고 가공시간(processing time) p_i를 갖는다. 여기서 작업 준비시간은 그 작업들이 가공되는 순서에 따라 달라지는데, 이러한 순서 의존적인 작업 준비시간은 작업 i의 가공을 마친 후 작업 j를 준비하는 데 소요되는 시간을 s_{ij}라 할 때, $s_{ij} \neq s_{ji}$인 성질을 갖는 작업 준비시간을 의미한다.

각 기계의 인덱스를 $m(m = 1, \cdots, M)$이라고 했을 때, 알고리듬에 의해 생성된 전체 스케줄 결과는 집합 $\Pi = \{\Pi_1, \Pi_2, \cdots, \Pi_m, \cdots, \Pi_M\}$으로 표현하고, 다시 각 기계별 스케줄 결과는 집합 $\Pi_m = \{\pi(1, m), \pi(2, m), \cdots, \pi(N_m, m)\}$으로 표현한다. 여기서 $\pi(j, m)$는 스케줄상의 m번째 기계의 j번째 위치해있는 작업의 인덱스를 뜻하고, N_m은 m번째 기계에 할당되어있는 작업의 수를 뜻한다. 본 연구의 스케줄링 목적은 L_{max}를 최소화하는 것으로 다음과 같이 표현될 수 있다.

$$\text{Minimize } L(\Pi) = \max\{L_i \mid i = 1, \cdots, N\} \quad (1)$$

이때 $L_i(=C_i - d_i)$는 작업 i의 납기지연시간이고 C_i는 작업 i의 가공 완료시간(completion time)이다. 이와 같이 정의되는 본 연구의 문제는 Lawler *et al.*(1993)의 표현식에 의해 $P|r_j, s_{ij}|L_{max}$로 나타낼 수 있으며, NP-hard 문제인 것으로 알려져 있다(Ovacik and Uzsoy, 1995). 본 연구에서는 작업 분할(job split) 및 아직 도착하지 않은 작업에 대한 작업준비를 허용하지 않는다고 가정한다.

Ali *et al.*(1999)의 조사 논문에는 작업 준비시간을 고려한 병렬 기계 스케줄링에 대한 기존 연구 결과들이 자세히 정리되어 있다. Parker *et al.*(1977)은 순서 의존적인 작업 준비시간이 존재하고 제한된 능력을 가진 병렬 기계 문제를 차량경로문제(vehicle routing problem)의 형태로 나타내었고, 총 기기변경비용(total changeover costs)을 최소화하기 위한 알고리듬을 제시하였다. Franca *et al.*(1996)은 $P|s_{ij}|C_{max}$ 문제를 풀기 위해 타부 탐색(tabu search; 이하 TS)을 적용한 3단계의 발견적 기법(heuristic)을 개발하였고 이 연구에서 그들은 TS의 이웃해 생성방안으로 외판원 문제(traveling salesman problem)의 해법으로 고안된 발견적 기법(Gendreau *et al.*, 1992)을 응용하였다.

Ovacik and Uzsoy(1995)는 $P|r_j, s_{ij}|L_{max}$ 문제를 풀기 위해 일련의 RHP(rolling horizon procedure) 알고리듬들을 제시하였으며, 이 알고리듬들이 기존의 할당규칙(dispatching rule)들은 물론 이들 할당규칙에 국지 탐색(local search)을 결합한 방법들보다 우수한 성능을 나타낸다는 것을 보였다.

Guinet(1993)은 $P|s_{ij}|C_{max}$ 문제를 풀기 위한 발견적 기법으로 선형 할당 문제의 해법인 확장된 헝가리안 방법(hungarian method)을 이용하였다. Lee and Pinedo(1997)는 납기지연 가중치의 합(total weighted tardiness ; 이하 TWT)을 최소화하는 $P|s_{ij}|TWT$ 문제를 풀기 위해 할당규칙과 시뮬레이티드 어닐링(simulated annealing ; 이하 SA) 기법을 결합한 3단계의 발견적 기법을 제시하였고, 이 연구에서 그들은 할당규칙 방안으로 Lee *et al.*(1997)가 단일 기계 문제에서 제안하였던 ATCS(apparent tardiness cost with setups)규칙을 수정하여 사용하였다.

납기가 주어진 동일병렬기계 문제에 대해 Laguna *et al.*(1991)은 작업조기완료 가중치 비용(weighted earliness penalty costs)을 최소화하기 위한 탐색 알고리듬을 제안하였다. 이 알고리듬은 GRASP(greedy randomized adaptive search procedure)과 TS를 결합하여 사용함으로써 스케줄을 수립하는 단계와, 분지한계법(branch and bound)을 이용하여 이미 수립된 각 기계별 스케줄을 향상시키는 단계로 구성되어 있다. Armentano and Yamashita(2000)는 병렬기계 스케줄링 문제에서 납기지연의 평균을 최소화하기 위한 TS 기법을 제시하였고, 그들은 전체 해 영역 중, 탐색과정에서 탐색해 보지 못한 부분을 찾아가기 위한 방안으로 두 가지 다각화 전략을 TS 기법에 적용하였다.

이상의 기존 연구 결과들을 고찰해보면 본 연구와 동일한 문제($P|r_j, s_{ij}|L_{max}$)에 대한 스케줄링 알고리듬을 제시한 기존 연구로는 Ovacik and Uzsoy(1995)의 RHP 알고리듬이 있다. 본 연구에서 제시하는 제한적 TS(restricted TS; 이하 RTS) 알고리듬은 초기해를 구하는 단계와 제한적 이웃해 생성 방안(restricted neighborhood generation scheme)을 이용하여 초기해를 향상시키는 단계로 구성되어 있다. 본 연구에서는 제시된 알고리듬의 객관적인 성능 평가를 위하여 벤치마킹 데이터(benchmarking data; Uzsoy)를 사용하여 RHP 알고리듬과의 비교를 수행함으로써 수행속도 및 해의 성능면에서 매우 우수한 결과를 나타낸다는 것을 보인다.

다음 2절에서는 RTS 알고리듬을 제시하고 RTS 알고리듬의 초기해 생성 방법과 제한적 이웃해 생성 방안에 대하여 설명한다. 3절에서는 실험을 통하여, 본 연구에서 제안한 알고리듬을 비교 대안 알고리듬들과 비교 분석하고 그 성능을 평가한다. 마지막으로 4절에서는 본 연구의 결론과 추후 연구방향을 정리한다.

2. 제한적 타부 탐색(RTS) 알고리듬

TS는 일종의 전체최적화(global optimization) 기법으로, 특정해(Π)로부터 생성된 이웃해 $N(\Pi)$ 중 최고해(Π^*)로 이동(move)하는 과정을 반복함으로써 해 영역(solution space)을 탐색해 나가는 메타 휴리스틱(meta heuristic)이다(Glover, 1989, 1990). 기존의 언덕등산(hill climbing) 알고리듬들과는 달리, TS는 지역최적해(local optimum)에 머무르는 것을 방지하기 위해 목적함수값을 저하시키더라도 좋지 않은 해로의 이동을 허용하는 메커니즘을 가지고 있다. 즉, TS는 각 이동에 관한 정보인 타부속성(tabu attribute)을 일정기간(tabu tenure)동안 타부목록(tabu list)에

기억함으로써 탐색의 중복과 해의 순환을 방지할 수 있다.

TS와 같은 메타 휴리스틱은 시뮬레이티드 어닐링이나 유전 알고리듬(genetic algorithm)과 더불어 적용하려는 문제 특성에 맞게 그 핵심 요소들을 얼마나 효과적으로 맞추어 사용하느냐에 따라 그 성능이 좌우된다. 일반적으로 TS의 성능에 영향을 미치는 요소들은 다음과 같다(Franca *et al.*, 1996).

- 이웃해 생성 방법
- 초기해의 성능
- 타부목록 크기
- 종료 조건

스케줄링 문제에 적용되어왔던 기존의 TS 기법들과 비교해 볼 때, 본 연구에서 제안하는 RTS 알고리듬의 가장 큰 특징들 중의 하나는 탐색성능을 향상시키기 위해 고안한 이웃해 생성 방안이다. 일반적으로 스케줄링 문제에 적용되어온 TS는 현재해에서 두 작업들의 위치를 바꾸는 교환이동(swap move)이나 현재해의 한 작업을 다른 위치로 삽입하는 삽입이동(insert move) 연산자(operator)들을 통해 이웃해를 생성한다. 그러나 이러한 이동 연산자에 의해 현재해로부터 가능한 모든 이웃이 생성되고 평가된다면 작업의 수가 증가함에 따라 함께 증가하는 이웃해로 인해 계산시간이 너무 많이 소요되고 탐색의 효율이 저하되는 문제가 발생한다. 따라서 본 연구에서는 기본적인 TS(basic TS; 이하 BTS) 기법의 탐색성능을 높이고 탐색에 소요되는 계산시간을 단축시키기 위해 교환이동과 삽입이동, 그리고 교환이동과 삽입이동을 결합하여 사용하는 혼합이동(hybrid move) 방법에 대해 각각 제한적 이웃해 생성 방안을 제안한다.

메타 휴리스틱을 이용한 탐색에 있어서 전체 탐색 영역 중, 최적지점에서 가까운 곳에서 출발하는 것이 결국 짧은 시간 내에 비교적 좋은 지점에 도달할 수 있다고 알려져 있다(Crauwels *et al.*, 1998). 따라서 본 연구에서는 효과적인 초기해를 생성하기 위한 방안으로 MATCS(modified ATCS) 규칙을 새롭게 제안하여 사용한다. 그리고 RTS 알고리듬의 열망 수준(aspiration level) 및 타부 목록 관리 방법과 같은 기타 핵심 요소들의 구체적인 내용은 다음절에서 자세히 설명하도록 한다.

2.1 초기해 생성

본 연구에서는 좋은 초기해로부터 TS를 수행하기 위해 Lee and Pinedo(1997)가 제안한 ATCS 규칙을 수정하여 사용하는 MATCS 규칙을 초기해 생성 방법으로 제안한다.

ATCS 규칙은 순서 의존적인 작업 준비시간과 작업의 납기를 고려하여 납기지연 가중치의 합을 최소화하는 병렬기계 문제($P|s_{ij}|TWT$)를 풀기 위해 제안되었다. ATCS 규칙은 현재 스케줄 시점(t)에서 바로 작업이 가능한 기계의 최종 할당된 작업이 l 일 때, 작업순서가 결정되지 않은 작업들(Ω) 중 그 다음 작업으로 인덱스($I_j(t, l)$)가 가장 큰 작업 j($j = \{j \mid \max\ I_j(t, l), \text{for } j \in \Omega\}$)가 스케줄 되도록 한다. 이 규칙을 적용함으로써 작업의 우선순위가 높을수록, 가공시간이 짧을수록, 납기까지의 여유가 없을수록, 그리고 순서 의존적인 작업 준비시간이 작을수록 그 해당 작업은 높은 인덱스를 갖게 되고 결과적으로 작업 순서상 앞 부분으로 스케줄된다.

그러나 본 연구에서는 모든 작업이 작업 투입시점을 갖는다는 점, 작업의 우선순위가 고려되지 않는 점, 그리고 목적함수로 L_{max}를 사용한다는 점에서 ATCS 규칙을 그대로 적용하기 힘들다. 따라서 본 연구에서는 ATCS 규칙의 인덱스 계산 수식을 수정하여 사용하는 MATCS 규칙을 이용한다.

MATCS 규칙에서 사용하는 인덱스 계산 수식은 수식 (2)과 같다.

수식 (2)에서, t는 작업 l의 가공 완료시점(C_l), $\bar{p}$와 $\bar{s}$는 각각 작업순서가 결정되지 않은 작업들의 평균 가공시간과 평균 작업 준비시간을 의미하고, k_1과 k_2는 조정모수(scaling parameter)로서 주어진 문제 특성에 따라 산출되며 수식 (2)의 두 번째 항과 세 번째 항이 전체 인덱스 값에 미치는 영향을 조절해 주는 역할을 한다. 그리고 본 연구에서 k_1과 k_2의 값은 Lee and Pinedo(1997)의 연구에서 제시한 계산식을 이용하여 산출한다.

$$I_j(t, l) = \frac{1}{p_j} \exp\left(-\frac{d_j - p_j - t}{k_1 \bar{p}}\right) \exp\left(-\frac{s_{lj}'}{k_2 \bar{s}}\right)$$

$$\text{단, } s_{lj}' = \begin{cases} s_{lj} & \text{if } r_j \le t \\ r_j - t + s_{lj} & \text{otherwise} \end{cases} \quad j \in \Omega \qquad (2)$$

식 (2)의 두 번째 항은 납기를 어긴 작업들에 대해 납기 지연된 정도에 따라 1이상의 값을 차등 제공함으로써, 작업순서 결정 시 납기 지연 정도의 차이를 분별할 수 있게 한다. 세 번째 항에서는 작업의 투입시점을 순서 의존적인 작업준비시간인 slj 에 반영하였는데, 만약 작업 j의 투입가능시점이 작업 l 의 종료시점인 t보다 작거나 같다면, 즉 작업 j가 t 시점에 투입 가능하다면 s_{ij}'값으로 기존의 s_{ij} 값을 그대로 사용하고, 그와 반대로 작업 l의 종료시점과 작업 j의 가공시작시점 사이에 유휴시간이 발생하게 된다면 s_{ij}' 값으로 기존의 s_{ij} 값에 유휴시간을 더한 값(= $s_{ij} + rj - t$)을 사용한다. 이렇게 함으로써, 불필요한 기계 유휴시간과 순서 의존적인 작업 준비시간을 함께 고려하는 스케줄 수립이 가능하다.

2.2 제한적 이웃해 생성 방안

본 연구에서 제안하는 RTS의 이웃해는 앞서 언급한 3개의 이동 연산자와 더불어 결정모수(decision parameter)인 탐색깊이(search depth)와 탐색반복횟수(search iteration number)에 의해 규정된다. 이 결정모수들을 적절히 사용함으로써, 이웃해의 생성을 일정수준 이하로 제약함과 동시에 이웃해를 찾는 과정을 좀 더 집중적이면서 다양하게 수행할 수 있게 함은 물론 탐색

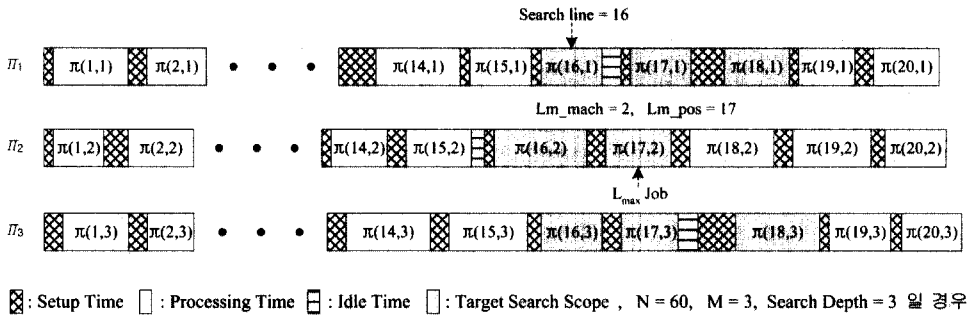

그림 1. 탐색영역 범위 설정(Search Line = 16, Lm_mach = 2, Lm_pos = 17).

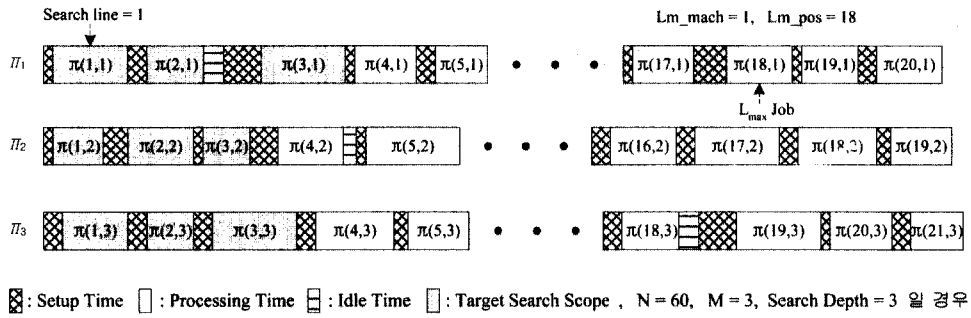

그림 2. 탐색영역 범위 설정(Search Line = 1, Lm_mach = 1, Lm_pos = 18).

의 강약을 조절할 수 있다.초기해로부터 출발하여 탐색과정에서 얻어지는 모든 현재해, Π_{cur}에는 작업 투입시점과 납기, 그리고 순서 의존적인 작업 준비시간 등이 반영되어있다.

<그림 1>의 스케줄을 Π_{cur}라 했을 때, 각 기계의 작업 순서상 선두에 위치해 있는 작업들은 대부분 납기가 급박하거나 작업 투입시점이 빠른 반면, 후미에 위치해 있는 작업들은 상대적으로 납기의 여유가 있거나 작업 투입시점이 늦는 특성을 갖는다. 그러므로 작업 순서가 위치적으로 많은 차이를 보이는 모든 작업들을 동시에 Π_{cur}로부터 생성되는 이웃해 집합인 $N(\Pi_{cur})$의 대상으로 한다는 것은 탐색의 효율을 낮추는 원인이 된다. 이와 같은 문제 특성을 반영하여 각 기계 내의 특정 작업의 위치를 탐색기준(search line)으로 하고 그 탐색기준으로부터 탐색깊이 범위 안의 작업들을 탐색영역(target search scope)으로 설정함으로써, 현재해로부터 생성되는 이웃해를 탐색영역 내로 제한하도록 한다.

<그림 1>에서 탐색기준은 $\pi(1,1), \pi(1.2), \pi(1,3)$, 작업들의 위치에 해당하는 1로 설정되어 있다. 이 탐색기준의 값은 변수로서 탐색영역 내의 작업들에 대한 탐색이 완료되고 난 후 1씩 증가하게 된다. 또한 탐색깊이는 3으로 설정되어 있는데 이 값은 알고리듬 수행 전에 일정한 값으로 주어지는 결정모수로 탐색기준이 가리키는 작업들을 포함해서 이 값이 가리키는 작업들 사이에 위치한 작업들을 탐색영역으로 지정해주는 역할을 한다. <그림 1>의 경우 작업 $\pi(1,1)$ 부터 작업 $\pi(3,3)$ 까지 총 9개의 음영 처리된 작업들이 탐색영역이 되면서 이웃해 생성의 대상이 된다.

납기지연이 L_{max}인 작업이 속한 기계 및 그 기계에서의 작업의 위치를 각각 *Lm_mach*와 *Lm_pos* 라고 했을 때, <그림 2>의 경우 L_{max} 작업이 $\pi(17,2)$이므로 *Lm_mach*값은 2이고 *Lm_pos* 값은 17이 된다. 이 *Lm_mach*와 *Lm_pos*는 탐색과정에서 이동에 의해 현재해가 바뀌고 그로 인해 L_{max} 작업이 변동될 경우 함께 변경되는 변수다.

Lm_mach 와 *Lm_pos*는 이웃해 생성을 효과적으로 제한하는데 중요한 역할을 한다. *Lm_mach* 기계에 속한 작업들만을 대상으로 탐색영역 내의 작업들과의 삽입 혹은 교환 등을 통해 이웃해가 생성되고, 탐색기준은 *Lm_pos* 값을 넘을 수 없다. 이러한 이유는 *Lm_mach* 이외의 기계들 간의 탐색 및 *Lm_pos* 이후에 위치한 작업들을 탐색영역에 포함시켜 탐색하는 것은 목적함수값, 즉 L_{max}를 감소시키지 못하기 때문이다.

<그림 2>는 탐색기준이 *Lm_pos* 에 근접해 있을 경우에 설정되는 탐색영역의 범위를 보여주고 있다. 탐색기준이 16이고 탐색깊이가 3일 때, 원칙적으로 탐색영역은 $\pi(16, 1)$부터 $\pi(18, 3)$까지 총 9개의 음영 처리된 작업들로 설정되지만 *Lm_pos* 값이 17이므로, *Lm_mach* 기계에 속한 작업들 중 작업 $\pi(18, 2)$은 탐색영역의 범위에서 제외된다. TS 알고리듬에서 사용하는 세 가지 이동법 -- 삽입이동, 교환이동, 혼합이동 -- 에 의한 제한적 이웃해 생성 방안은 다음 절에서 자세히 설명하도록 한다.

2.2.1 제한적 삽입이동 연산자

삽입이동은 현재의 스케줄(Π_{cur})에서 특정 기계의 한 작업

을 선택하여 동일 기계의 다른 위치 또는 다른 기계에 삽입하여 이웃해를 생성해 내는 방법이다. 삽입이동을 통해 이웃해를 생성하기 위해서는 옮길 작업과 옮겨갈 위치를 결정해 주어야 한다. 옮길 작업은 탐색 영역 내의 작업이면서 *Lm_mach* 기계에 속한 작업이 해당되고, 옮겨갈 위치는 다음의 삽입이동 조건을 만족하는 모든 위치이다.

[삽입이동 조건]

옮길 작업이 $\pi(j, m)$이고 옮길 위치가 기계 v의 k 위치일 때,

a) $j \in$ Target Search Scope, $m = Lm\_mach$

b) $v=m$인 경우(옮길 작업과 옮길 위치가 동일 기계일 경우),

i) $k \neq j$

ii) $r_{\pi(j,m)} \leq C_{\pi(k,v)}$ for, $k < j$

iii) 작업 $\pi(j, m)$을 위치 k로 이동하였을 때의 예상 종료시점($C_{\pi(j,m)}$) $\leq d_{\pi(j,m)} + L(\Pi_{cur})$. for, $k > j$

iv) 타부 목록에 의해 금지된 이동이 아님.

c) $v \neq m$인 경우(옮길 작업과 옮길 위치가 동일 기계가 아닐 경우),

i) $k \in$ Target Search Scope

ii) 작업 $\pi(j, m)$을 기계 v의 위치 k로 이동하였을 때의 예상 종료시점 ($C_{\pi(j,m)}$) $\leq d_{\pi(j,m)} + L(\Pi_{cur})$.

iii) 타부 목록에 의해 금지된 이동이 아님.

예를 들어, <그림 2>의 경우 탐색영역 내의 작업 중 *Lm_mach* 기계에 속한 $\pi(16,2)$과 $\pi(17,2)$ 가 옮길 작업이 된다. 옮길 작업이 $\pi(16,2)$이라면 옮겨갈 위치는 *Lm_mach* 기계의 작업순서 1부터 20까지의 위치 중, 위 b)의 삽입이동 조건을 만족시키는 곳들과 탐색영역 내의 위치 중, 위 c)의 삽입이동 조건을 만족시키는 곳들이 해당된다. 따라서 기계수가 M대이고 각 기계에 동일하게 N개의 작업이 할당되어 있다고 할 경우, 한번의 이동을 위해 생성되고 평가받는 이웃해의 수는 다음과 같다. 먼저, 옮길 작업의 수는 탐색깊이가 되고, 옮길 작업 하나당 옮겨갈 위치의 수는 *Lm_mach*기계에서 $N-1$개, 그 외의 기계에서 $(M-1)\times$(탐색깊이$+1$)개이므로, 한번의 이동에서 생성되는 이웃해의 수는 최대 (탐색깊이)$\times(N-1+(M-1)\times$(탐색깊이$+1$))를 넘지 않는다. 또한 탐색기준이 *Lm_pos*에 가까워짐에 따라 *Lm_mach* 기계 내의 탐색영역의 범위가 작아진다는 점과 함께 삽입이동 조건이라는 제약으로 인해 실제로 생성되는 이웃해의 수는 실험적인 경험에 의하면 대부분 위의 최대 이웃해 생성 수를 크게 밑돌게 된다. 이것은 BTS 삽입이동의 경우 한번의 이동을 위해 생성되는 이웃해의 수가 $(M\times N)\times(N-1+(M-1)\times(N+1))$이라는 것과 비교해 볼 때, RTS는 적절한 탐색깊이 설정에 따라 이웃해가 상당한 수로 감소된다는 것을 알 수 있다.

2.2.2 제한적 교환이동 연산자

교환이동은 두 작업의 작업 순서를 서로 교환함으로써 이웃해를 생성하는 방법이다. 교환이동에 의해 이웃해를 생성해내기 위해서는 작업 순서를 교환할 서로 다른 두 개의 작업을 선택해 주어야 하는데, 이 두 작업은 탐색영역 내에 있으면서 다음의 교환이동 조건을 만족하는 모든 작업의 조합으로 이루어진다.

[교환이동 조건]

교환할 두 작업을 각각 $\pi(j, m)$과 $\pi(k, v)$라 할 때,

i) $j \in$ Target Search Scope, $m = Lm\_mach$

ii) $k \in$ Target Search Scope, $k \neq j$

iii) $r_{\pi(j,m)} \leq C_{\pi(k,v)}$ for, $v = m, j > k$

iv) 타부 목록에 의해 금지된 이동이 아님.

즉, 탐색영역 내의 작업들을 대상으로 형성되는 조합중 위의 교환이동 조건을 만족시키는 작업들에 대해서만 이웃해가 생성된다. 따라서 기계수가 M대이고 각 기계에 동일하게 N개의 작업이 할당되어 있다고 할 경우, 한번의 이동을 위해 생성되고 평가받는 이웃해의 수는 최대 (탐색깊이)$\times$(탐색깊이$\times(2M-1)-1)/2$를 넘지 않는다. 이것은 BTS 교환이동의 경우, 한번의 이동을 위해 생성되는 이웃해의 수가 $(M\times N)\times(N\times(2M-1)-1)/2$라는 것과 비교해 볼 때, 본 연구에서 제안하는 RTS의 삽입이동 방법은 탐색영역이 적정 수준으로 설정되었을 때 매우 적은 수의 이웃해를 생성한다는 것을 알 수 있다.

2.2.3 제한적 혼합이동 연산자

혼합이동은 현재탐색횟수가 증가할 때마다 삽입 이동과 교환 이동을 서로 교환해 가면서 사용하는 방법이다. 즉, 탐색기준 값이 *Lm_pos*보다 커지면 다시 탐색기준의 값이 1이 되면서 한번의 탐색을 마감하고, 다음 회수의 탐색을 반복하게 되는데, 이 때 이동 연산자를 서로 바꾸어준다. 각 반복마다 사용되는 삽입이동과 교환이동 방법은 앞서 기술한 내용과 동일하다.

2.3 RTS 알고리듬의 타부 목록 관리

RTS 알고리듬은 단기 기억(short-term memory) 방식의 타부 목록을 사용하여 탐색 과정에서 해의 순환을 관리한다. Glover(1989, 1990)는 장기 기억(long-term memory)을 사용함으로써 탐색의 다양성을 증가시킬 수 있다고 했다. 본 연구에서는 2.2.3절에서 언급한 RTS 알고리듬의 제한적 혼합이동 연산자를 사용함으로써 TS의 장기 기억이 주는 효과와 유사한 결과를 얻을 수 있음을 제 3 장에서 보여준다.

TS가 적용된 기존 연구들에서 타부 목록 크기는 대체적으로 6에서 10 사이의 값이 사용됐는데(Laguna *et al.*, 1991), 본 연구에서는 이러한 기존 연구와 RTS 알고리듬의 실험 경험을 바탕으로 타부 목록 크기를 7로 고정하여 사용하도록 한다. 각 이동 연산자의 타부 목록 정의 방법은 다음과 같다.

2.4 스케줄링 알고리듬

RTS 알고리듬은 다음과 같이 5개의 절차로 구성된다.

Step 0. 탐색기준과 현재탐색횟수(current search number)값을 1로 각각 설정한다. 탐색깊이와 탐색반복횟수를 각각 주어진 값으로 초기화하고, 삽입이동, 교환이동, 혼합이동 중 하나의 이동 연산자를 선택한다.

Step 1. MATCS를 이용하여 초기해 Π와 목적함수값 $L(\Pi)$을 구하고, 이것을 각각 현재해 Π_{cur}와 $L(\Pi_{cur})$ 및 최우수해(best schedule) Π_{Best}와 $L(\Pi_{Best})$로 설정한다. Π에서 L_{max} 작업의 위치를 파악하여 *Lm_mach*와 *Lm_pos*의 값을 설정한다.

Step 2. Π_{cur}의 탐색영역 내에 있는 작업들에 대해 Step 0에서 선택한 이동의 제한적 이웃해 생성 방안을 이용하여, 가능한 모든 이웃해 집합 $N(\Pi_{cur})$을 생성하고 생성된 이웃해들의 목적함수값을 계산한다.

Step 3. $N(\Pi_{cur})$ 중 목적함수값이 가장 작은 이웃해 Π^*를 현재해 Π_{cur}로 바꾸고 이때의 이동을 해당 이동 연산자의 타부 목록에 저장해 둔다. 만약, $L(\Pi^*)$ 값이 $L(\Pi_{Best})$ 값보다 작다면 Π_{Best}와 $L(\Pi_{Best})$을 각각 Π^*과 $L(\Pi^*)$로 치환한다. 그리고 L_{max} 작업의 위치가 바뀌었다면, *Lm_mach*와 *Lm_pos*의 값을 변경해 준다.

Step 4. 탐색기준을 1 증가시키고 Step 2로 되돌아간다. 이때, 만약 증가시킨 탐색기준이 *Lm_pos* 값과 일치한다면 탐색기준을 1로 초기화시키고 현재탐색횟수 값을 1 증가시킨다. 그리고 만약 증가시킨 현재탐색횟수값이 미리 설정된 탐색반복횟수 값보다 크다면 TS를 종료시킨다.

3. 비교 대안 및 실험

3.1 비교 대안

RTS 알고리듬의 성능 측정을 위한 비교 대안으로는 동일한 문제에 대한 벤치마킹 데이터가 주어져 직접적인 비교가 가능한 Ovacik and Uzsoy(1995)의 RHP 알고리듬을 채택하였다. 그리고 RTS 알고리듬 내의 제한적 이웃해 생성 방안의 성능을 평가하기 위해 기존의 메타 휴리스틱인 BTS와 SA를 대안으로 사용하였으며, 각 방법들의 핵심요소는 설정(tuning)은 각 대안 설명에서 언급한다.

3.1.1 RHP 알고리듬

RHP 알고리듬은 전체 계획구간(planning horizon) 내에 존재하는 작업들을 일정한 기간의 예측구간(forecast window)으로 분해(decomposition)하여 하위문제를 구성하고, 구성된 하위문제를 대상으로 이미 설정된 값인 최대 k개의 작업에 대해 분지한계법을 이용하여 부분최적해를 구한다. 그리고 구해진 부분최적해 중에서 역시 미리 주어진 값인 λ개의 작업들만을 병렬 기계중의 한 대를 택일하여 할당한 후, 전체 스케줄로 확정시키는 일련의 과정을 반복하는 절차로 구성되어 있다.

Ovacik and Uzsoy(1995)는 k와 λ 등의 결정모수들의 조합으로 이루어지는 알고리듬들을 이용하여 해를 구하였고, 이 때 사용된 실험 데이터와 위의 알고리듬들을 이용하여 얻은 해 중 가장 우수한 결과(이하 O&U해)를 그들의 웹사이트(Uzsoy)에 제시하였다. 따라서 본 연구에서 제시한 알고리듬과 RHP 알고리듬의 성능을 그들이 사용한 실험 데이터와 결과를 이용하여 직접적으로 비교 분석하도록 한다.

3.1.2 BTS

BTS의 핵심 요소는 이웃해 생성 방법과 종료조건을 제외하고, 본 연구에서 제시한 RTS 알고리듬과 동일하게 설정한다. 이웃해 생성 방법은 2.2.1에서 언급한 일반적인 삽입이동 연산자를 사용하고, 탐색은 해가 O&U해에 도달하였을 때 종료되도록 한다.

3.1.3 SA

SA는 초기해 및 이웃해 생성 방법 등의 핵심요소에 있어서 BTS와 동일한 방법을 사용한다. 단, SA의 선택확률함수(acceptance probability function)는 Lee and Pinedo(1997)가 그들의 연구에서 Matsuo *et al.*(1989) 및 Vakharia *et al.*(1990)의 연구결과를 근거로 채택하였던 선형 감소 함수의 형식을 따르도록 설정한다. 선택확률함수의 초기 온도값은 0.8이며 내부루프인 각 단계(stage)가 증가할수록 0.04씩 감소한다. 그리고 한 단계에서의 반복횟수는 20이며 전체 탐색은 해가 O&U해에 도달하였을 때 종료되도록 한다.

3.2 실험 데이터

Ovacik and Uzsoy(1995)의 벤치마킹 실험 데이터[16]는 <표 1>과 같이 기계수, 작업수, 그리고 R값의 조합으로 생성된 1800개의 문제로 구성되어 있다. 여기서 R값이란 작업 투입시점의 범위 모수(release time range parameter)로서, 이 값이 작을수록 작업이 빈번하게 도착하여 기계 앞에서 가공을 대기하는 작업수가 증가하게 되고, 반대로 이 값이 클수록 작업이 드문

표 1. 실험 데이터 생성 기준

	사용값	합계
기계수	2, 4, 6	3
작업수	25, 50, 75, 100, 125, 150	6
작업 투입시점 범위 모수 (R)	0.2, 0.6, 1.0, 1.4, 1.8	5
조합 수		90
조합 당 문제 수		20
전체 문제 수		1800

드문 도착하기 때문에 상대적으로 기계 앞에서 대기하는 작업 수가 줄어들게 된다. 만약, R값이 0이라면 해당 문제는 정적인 문제(static problem)가 되며 모든 작업이 스케줄 시점에 가용하다. 본 연구에서 제시한 알고리듬은 모두 C++를 이용하여 구현하였고, 일반 데스크탑 컴퓨터에서 실험하였으며 이때 사용된 CPU는 Pentium Ⅱ 400이다.

3.3 초기 실험

본 연구에서 제안된 알고리듬의 성능은 초기해의 질과 제시된 세 가지 제한적 이웃해 생성 방안에 의하여 결정되고, 그 정도는 결정모수인 탐색깊이와 탐색반복횟수 값에 따라 차이를 보인다. 본 절에서는 MATCS 규칙과 기존의 할당규칙들과의 수행 성능을 비교하고, MATCS 규칙을 이용하여 얻은 초기해가 전체 알고리듬의 결과에 미치는 영향에 대하여 평가한다. 또한 탐색반복횟수가 알고리듬 수행 성능에 미치는 영향을 실험을 통하여 분석하여 효율적인 탐색반복횟수 값을 결정한다.

초기 실험은 앞서 언급한 문제집합 중 작업수가 150인 문제들을 사용하고 탐색깊이는 25로 고정하여 실험한다.

3.3.1 초기해의 성능 평가 및 탐색 횟수의 결정

본 실험에서는 식 (3)와 같이 제시된 알고리듬으로 실험하여 얻은 L_{max}값을 O&U해로 나누어 준 비교값(comparison value)을 해의 질을 가늠하는 척도로 사용한다. 즉, 비교값이 1보다 작을 경우 제시한 알고리듬의 결과가 O&U해보다 좋다는 것을, 반대로 1보다 클 경우 좋지 않다는 것을 의미한다.

$$\text{비교값} = \frac{RTS\ \text{알고리듬이 수립한 스케줄의}\ L_{max}}{\text{대안 알고리듬이 수립한 스케줄의}\ L_{max}} \quad (3)$$

<그림 3>은 주어진 작업을 병렬기계로 할당하기 위한 초기해 생성 방법으로 MATCS, ATCS, EDD, 그리고 ER(earliest release-time) 규칙을 사용하였을 경우, RTS 알고리듬의 성능을 탐색반복횟수에 따라 나타낸 결과다.

초기해의 성능은 <그림 3>의 탐색반복횟수가 0일 때의 비교값을 보면 알 수 있는데, MATCS 규칙이 나머지 세 개의 할당규칙에 비해 O&U 해에 매우 근접해있는 결과를 보여주고 있다. 그리고 RTS-MATCS는 MATCS 규칙을 초기해 생성 규칙으로 사용한 RTS 알고리듬을 의미하며, 이때의 비교값은 혼합이동 방법을 사용한 RTS 알고리듬(이하 RTS-Hybrid)과 삽입이동 방법을 사용한 RTS 알고리듬(이하 RTS-Insert), 그리고 교환이동 방법을 사용한 RTS 알고리듬(이하 RTS-Swap)을 사용하여 얻은 비교값들의 평균이다. <그림 3>에서 알 수 있듯이 네 개의 할당규칙을 사용하여 얻은 초기해 값은 큰 차이를 보이지만 RTS 알고리듬의 탐색반복횟수가 늘어날수록 그 차이가 좁혀지는 것을 볼 수 있다. 그러나 현실적으로 적은 수행시간을 요

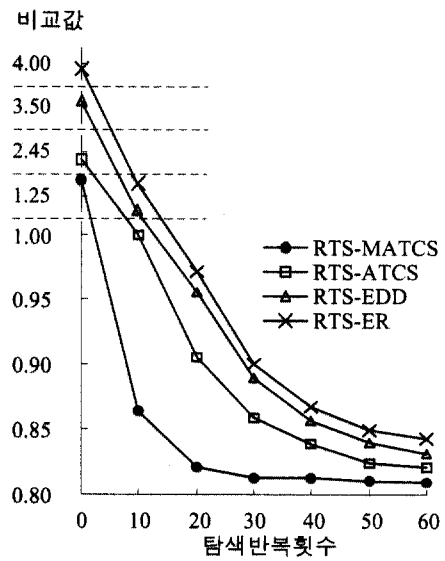

그림 3. 탐색반복횟수에 따른 초기해 생성 규칙 간의 성능 비교.

하는 문제에서는 10 이하의 적은 탐색반복횟수 값으로 RTS-MATCS 알고리듬을 사용한다면 해의 질과 수행속도 측면에서 효과적일 수 있다.

실험 경험에 의해 탐색반복횟수가 증가함에 따라 수행속도가 거의 선형적으로 비례하여 증가한다는 것을 알 수 있었다. 또한 <그림 3>에서 RTS-MATCS의 경우 탐색반복횟수가 증가할수록 해가 향상되지만 탐색반복횟수가 일정한 값에 도달하게 되면 해의 향상이 미미하다는 점을 감안하여, 본 실험에서는 30회를 탐색반복횟수값으로 고정하여 사용하도록 한다.

3.4 실험 결과 및 분석

3.1에서 언급한 모두 1,800개의 실험 데이터를 사용하여 본 연구에서 제시한 RTS 알고리듬의 성능을 O&U해와 비교해 보기로 한다. 초기 실험 결과 초기해 생성 규칙은 MATCS 규칙을 사용하고 탐색반복횟수는 30회로 고정한다. 그리고 다른 하나의 결정모수인 탐색깊이와 세 개의 이웃해 생성 방안이 TS 알고리듬의 성능에 미치는 영향을 분석하기 위하여 <표 2>와 같이 실험을 계획한다.

탐색깊이를 10단위로 10에서 80까지 변화시켜 가며 작업투

표 2. 실험 계획

결정모수	사용값
초기해 생성 규칙	MATCS 규칙
탐색반복횟수	30
탐색깊이	10, 20, 30, … , 70, 80
이웃해 생성 방안	RTS-Hybrid, RTS-Insert, RTS-Swap

입시점 범위 모수인 R값과 세 가지 이동 연산자, 그리고 작업수 N 및 기계수 M에 따른 RTS 알고리듬의 실험 결과는 <표 3>에 정리되어 있고, 이 표에 사용된 결과값은 모두 비교값이다. <표 3>에서 Avg(R,*,*)은 특정 R값이 주어졌을 때 모든 작업수 N 및 기계수 M에 대한 해(비교값)의 평균을 의미하며, Avg(*,N,*)과 Avg(*,*,M), 그리고 Avg(*,*,*)는 이와 같은 방식으로 해석한다.

먼저 이웃해 생성 방법에 따른 해의 결과를 살펴보면 다음과 같다. RTS-Insert의 결과가 RTS-Swap의 결과에 비해 훨씬 좋음을 알 수 있는데, 그 첫 번째 이유는 2.2.1과 2.2.2절에서 언급한바와 같이 삽입이동이 교환이동에 비해 *Lm_mach* 기계를 중

표 3. R값 및 탐색깊이와 이동 연산자에 따른 RTS 알고리듬 비교값

RTS-Hybrid								
탐색깊이	10	20	30	40	50	60	70	80
Avg(0.2,*,*)	0.921	0.879	0.844	0.823	0.812	0.810	0.809	0.809
Avg(0.6,*,*)	0.932	0.888	0.857	0.832	0.820	0.818	0.816	0.815
Avg(0.10,*,*)	0.934	0.898	0.867	0.836	0.829	0.819	0.820	0.819
Avg(0.14,*,*)	0.952	0.903	0.889	0.850	0.841	0.832	0.827	0.827
Avg(0.18,*,*)	0.959	0.910	0.891	0.878	0.860	0.842	0.838	0.835
Avg(*,25,*)	0.825	0.821	-	-	-	-	-	-
Avg(*,50,*)	0.830	0.819	0.818	-	-	-	-	-
Avg(*,75,*)	0.889	0.863	0.828	0.816	-	-	-	-
Avg(*,100,*)	0.917	0.872	0.844	0.821	0.817	-	-	-
Avg(*,125,*)	0.943	0.903	0.869	0.838	0.820	0.818	0.817	-
Avg(*,150,*)	0.962	0.935	0.899	0.846	0.829	0.817	0.819	0.813
Avg(*,*,2)	0.911	0.879	0.859	0.837	0.828	0.821	0.820	0.819
Avg(*,*,4)	0.904	0.876	0.858	0.834	0.825	0.820	0.819	0.818
Avg(*,*,6)	0.898	0.872	0.855	0.831	0.823	0.818	0.819	0.817
Avg(*,*,*)	0.940	0.896	0.870	0.844	0.832	0.824	0.822	0.821
RTS-Insert								
탐색깊이	10	20	30	40	50	60	70	80
Avg(0.2,*,*)	0.968	0.913	0.884	0.859	0.829	0.826	0.821	0.820
Avg(0.6,*,*)	0.977	0.926	0.892	0.863	0.832	0.828	0.827	0.828
Avg(0.10,*,*)	0.980	0.948	0.897	0.869	0.841	0.832	0.830	0.830
Avg(0.14,*,*)	0.983	0.950	0.898	0.871	0.844	0.834	0.833	0.832
Avg(0.18,*,*)	0.989	0.955	0.899	0.880	0.862	0.846	0.840	0.839
Avg(*,25,*)	0.831	0.829	-	-	-	-	-	-
Avg(*,50,*)	0.838	0.826	0.822	-	-	-	-	-
Avg(*,75,*)	0.906	0.871	0.838	0.821	-	-	-	-
Avg(*,100,*)	0.944	0.903	0.852	0.835	0.824	-	-	-
Avg(*,125,*)	0.989	0.932	0.880	0.848	0.833	0.828	0.825	-
Avg(*,150,*)	1.199	0.962	0.901	0.856	0.841	0.829	0.828	0.825
Avg(*,*,2)	0.960	0.901	0.870	0.854	0.836	0.831	0.829	0.827
Avg(*,*,4)	0.956	0.896	0.866	0.847	0.835	0.830	0.828	0.825
Avg(*,*,6)	0.852	0.890	0.861	0.841	0.832	0.829	0.827	0.823
Avg(*,*,*)	0.979	0.939	0.894	0.869	0.842	0.833	0.830	0.830
RTS-Swap								
탐색깊이	10	20	30	40	50	60	70	80
Avg(0.2,*,*)	1.299	1.138	1.008	0.984	0.936	0.907	0.886	0.871
Avg(0.6,*,*)	1.320	1.219	1.108	0.996	0.947	0.918	0.891	0.880
Avg(0.10,*,*)	1.457	1.285	1.189	1.002	0.968	0.930	0.912	0.901
Avg(0.14,*,*)	1.502	1.290	1.195	1.010	0.987	0.939	0.920	0.908
Avg(0.18,*,*)	1.588	1.352	1.220	1.088	0.992	0.944	0.931	0.918
Avg(*,25,*)	0.903	0.901	-	-	-	-	-	-
Avg(*,50,*)	0.928	0.920	0.918	-	-	-	-	-
Avg(*,75,*)	1.020	0.968	0.920	0.919	-	-	-	-
Avg(*,100,*)	1.341	1.109	0.969	0.936	0.929	-	-	-
Avg(*,125,*)	1.549	1.316	1.085	0.948	0.934	0.931	0.928	-
Avg(*,150,*)	1.764	1.374	1.190	0.997	0.945	0.930	0.926	0.925
Avg(*,*,2)	1.330	1.170	1.078	0.985	0.954	0.931	0.928	0.921
Avg(*,*,4)	1.284	1.139	1.049	0.971	0.946	0.925	0.923	0.918
Avg(*,*,6)	1.252	1.100	1.019	0.957	0.938	0.921	0.912	0.901
Avg(*,*,*)	1.433	1.257	1.144	1.016	0.966	0.928	0.908	0.896

심으로 한 전체탐색(global search)의 성격이 강하기 때문이다. 두 번째 이유는 삽입이동이 *Lm_mach* 기계 내의 *Lm_pos* 이전 작업들을 다른 기계로 이동시킴으로써 목적함수값을 보다 빨리 감소시킬 수 있는 반면, 교환이동의 경우는 항상 서로 다른 두 작업의 위치를 교환하기 때문에 이웃해의 질이 서로 상쇄되어 좋은 해로 이동하려는 성질을 방해하기 때문이다.

RTS-Hybrid는 RTS-Insert보다 더 좋은 해를 보여주는데, 그 이유는 다음과 같다. RTS-Insert나 RTS-Swap은 한 가지의 이웃해 생성 방법만 사용하므로 탐색이 진행됨에 따라 특정 지역으로 해가 이동한 이후에도 해가 더 좋아지지 않는 수렴현상이 발생할 경우 해의 순환이 발생할 확률이 커진다. 그러나 RTS-Hybrid는 삽입이동과 혼합이동을 교대로 사용하므로 타부 목록 관리 방법 중의 하나인 장기 기억의 효과를 나타낼 수 있다. 즉 RTS-Hybrid는 보다 넓은 범위에서 해의 순환을 막아주며 다양하게 해를 찾아가는 특성을 보여준다. R값에 따른 해의 결과를 살펴보면 전체적으로 R값이 작을수록 본 연구에서 제시한 알고리듬의 해가 RHP 알고리듬보다 좋다는 것을 알 수 있다.

탐색깊이와 해의 질의 관계는 <그림 4>에 나타나 있는데, 탐색깊이 값이 커질수록 탐색강도가 커짐으로 인해 해의 향상 정도가 증가한다는 것을 보여주고 있다. 그러나 탐색깊이 값이 일정한 값에 도달하게 되면 해의 향상 정도가 거의 없는 반면에 <그림 5>에서와 같이 알고리듬 수행 속도는 탐색깊이 값에 따라 일정하게 비례하여 증가한다는 점에서 효율적인 탐색깊이를 결정하는 것이 필요하다는 것을 실험결과 알 수 있다.

<그림 5>, <그림 6>, 그리고 <그림 7>에서 RTS-Hybrid(Avg)와 RTS-Hybrid(Max)는 RTS-Hybrid 알고리듬을 사용하였을 때 사용된 평균 CPU 시간과 최대 CPU 시간을 각각 의미한다. <그림 6>과 <그림 7>은 작업수와 기계수에 따른 RTS-Hybrid 알고리듬의 수행 속도 결과를 보여주고 있다. 알고리듬 수행 속도는 작업수와는 비례관계를, 기계수와는 반비례관계를 나타내는 것을 알 수 있다.

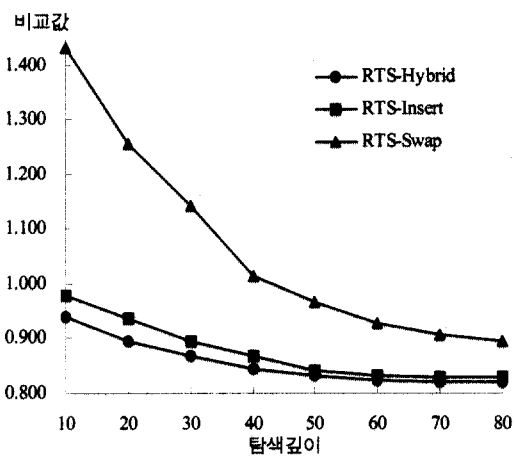

그림 4. 탐색깊이에 따른 해의 결과.

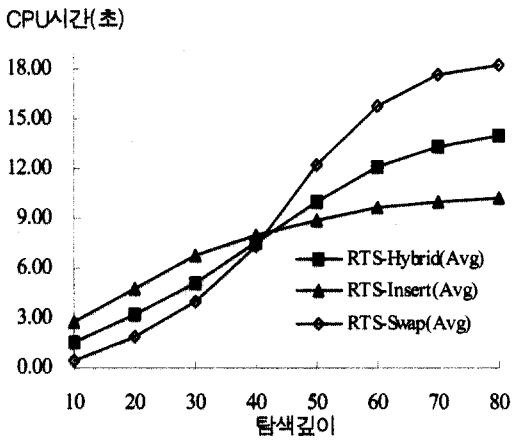

그림 5. 탐색깊이에 따른 평균 수행속도.

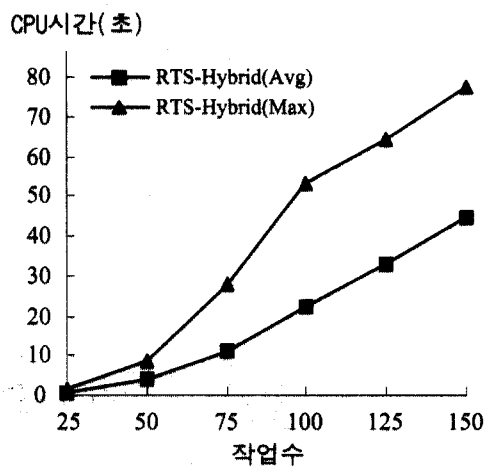

그림 6. 작업수에 따른 RTS-Hybrid 수행 속도.

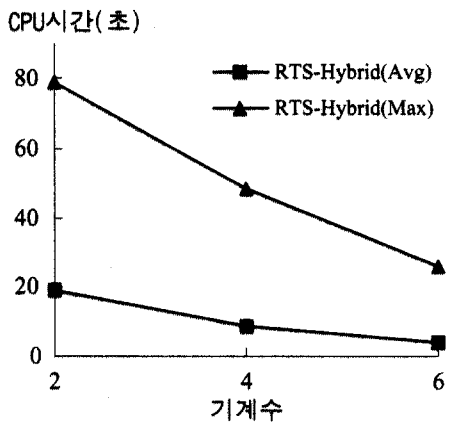

그림 7. 기계수에 따른 RTS-Hybrid 수행 속도.

<표 4>는 작업수가 150인 문제집합에 대해 RTS-Hybrid 알고리듬 및 앞서 비교대안으로 언급한 RHP 알고리듬과 BTS, 그리고 SA를 사용하여 얻은 결과가 O&U 해에 도달하는 데까지

표 4. O&U해에 도달하는 데 소요된 시간(초)

R 값	0.2	0.6	1.0	1.4	1.8
RTS-Hybrid	2.74	2.95	3.04	3.31	3.54
RHP	64.24	63.75	61.03	57.59	51.48
BTS	1089.32	1328.48	1356.63	1428.79	1542.94
SA	1002.40	1334.12	1391.35	1490.88	1603.82

표 5. RTS 알고리듬의 성능 비교

	우수한 경우 (%)	동일한 경우 (%)	열등한 경우 (%)
Avg(0.2,*,*)	98.6%	1.4%	0.0%
Avg(0.6,*,*)	96.5%	3.5%	0.0%
Avg(0.10,*,*)	94.8%	5.2%	0.0%
Avg(0.14,*,*)	92.5%	7.5%	0.0%
Avg(0.18,*,*)	92.0%	8.0%	0.0%
Avg(*,25,*)	93.5%	6.5%	0.0%
Avg(*,50,*)	94.5%	5.5%	0.0%
Avg(*,75,*)	94.5%	5.5%	0.0%
Avg(*,100,*)	95.5%	4.5%	0.0%
Avg(*,125,*)	95.5%	4.5%	0.0%
Avg(*,150,*)	96.0%	4.0%	0.0%
Avg(*,*,2)	95.0%	5.0%	0.0%
Avg(*,*,4)	95.0%	5.0%	0.0%
Avg(*,*,6)	94.8%	5.2%	0.0%
Avg(*,*,*)	94.9%	5.1%	0.0%

소요된 CPU 시간을 보여주고 있다. RTS-Hybrid 알고리듬이 RHP 알고리듬에 비해 약 15배에서 24배 이상의 매우 빠른 수행속도를 나타낸다는 것을 실험결과 알 수 있었고, 제한적 이웃해를 사용하지 않는 BTS와 SA와는 확연한 속도 차이를 보였다.

마지막으로, <표 5>는 전체적인 RTS 알고리듬의 성능을 확인하기 위하여 1800개의 문제집합에 대해 RTS 알고리듬에 의해 얻은 가장 좋은 해를 기준으로 RHP 알고리듬의 O&U해와 비교하여 우수한 결과를 보인 횟수, 동일한 결과를 보인 횟수, 좋지 않은 결과를 보인 횟수를 비율로 나타낸 것이다.

지금까지 분석한 결과들을 종합해보면 본 연구에서 제시한 RTS 알고리듬은 해의 질과 수행속도면에서 RHP 알고리듬보다 매우 우수하며, 특히 기계 앞에서 대기하고 있는 가용한 작업수가 많을수록(R값이 작을수록), 작업수와 기계수가 큰 문제일수록 RTS 알고리듬은 보다 탁월한 성능을 보여준다는 것을 알 수 있다.

4. 결론 및 추후 연구

본 연구에서는 순서 의존적인 작업 준비시간과 작업 투입시점이 존재하는 동적인 환경에서 L_{max}를 최소화하는 병렬기계 스케줄링 방법을 제시하였다. 본 논문에서 제시된 방법은 초기해를 구하는 방법과 초기해를 개선하는 알고리듬으로 구성되어 있으며, 초기해를 개선하는 RTS 알고리듬은 탐색의 효율을 높여주는 제한적 이웃해 생성 방안 사용하였다. 제시된 알고리듬의 객관적인 성능 평가를 위하여 벤치마킹 데이터(Uzsoy)를 사용하여 기존의 RHP 알고리듬에 비해 빠른 시간 내에 좋은 해를 찾음을 보여주었고, 특히 작업수가 증가할수록 작업투입시점이 집중되어 있는 환경일수록 매우 우수한 해를 제공한다는 것을 입증하였다. 또한 본 연구에서 제안한 제한적 이웃해 생성 방안의 성능의 우수성을 BTS 및 SA와의 비교 실험을 통해 평가하였다.

추후 연구로는 RTS 알고리듬을 동일 병렬 기계에서뿐만 아니라 더 복잡한 문제인 이종 병렬기계(nonidentical parallel machine) 또는 병렬기계로 구성된 유연 잡샵(flexible job shop) 환경으로 확장하여 적용하는 연구가 요구되어진다.

참 고 문 헌

Ali, A., Jatinder, N. D. G. and Tariq, A. (1999), A review of scheduling research involving setup considerations, *The International Journal of Management Science*, 27(2), 219-239.

Armentano V. A. and Yamashita, D. S. (2000), Tabu search for scheduling on identical parallel machines to minimize mean tardiness, *Journal of Intelligent Manufacturing*, 11(5), 453-460.

Crauwels, H. A. J. and Potts, C. N. (1998), Local search heuristics for the single machine total weighted tardiness scheduling problem, *INFORMS Journal on Computing*, 10, 341-350.

Franca, P. M., Gendreau, M., Laporte, G. and Muller, F. M. (1996), A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times, *International Journal of Production Economics*, 43(2-3), 79-89.

Gendreau, M., Hertz, A. and Laporte, G. (1992), New insertion and post-optimization procedures for the traveling salesman problem, *Operations Research*, 40(6), 1086-1094.

Glover, F. (1989), Tabu search-Part I, *ORSA Journal on Computing*, 1, 190-206.

Glover, F. (1990), Tabu search-Part II, *ORSA Journal on Computing*, 2, 4-32.

Guinet, A. (1993), Scheduling sequence-dependent jobs on identical parallel machines to minimize completion time criteria, *International Journal of Production Research*, 31(7), 1579-1594.

Kim, S. Y. and Bobrowski, P. M. (1994), Impact on sequence-dependent setup times on job shop scheduling performance, *International Journal of Production Research*, 32(7), 1503-1520.

Laguna, M. and Velarde, J. L. G. (1991), A search heuristic for just-in-time scheduling in parallel machines, *Journal of Intelligent Manufacturing*, 2(4), 253-260.

Laguna, M., Barnes, J. W. and Glover, F. (1991), Tabu search methods for single machine scheduling problem, *Journal of Intelligent Manufacturing*, 2(2), 63-74.

Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G. and Shmoys, D. B. (1993), Sequencing and scheduling: algorithms and complexity, in Handbooks in

Operations Research and Management Science: Logistics of Production and Inventory, North Holland, 445-522.

Lee, Y. H., Bhaskran, K. and Pinedo, M. (1997), A heuristic to minimize the total weighted tardiness with sequence-dependent setups, *IIE Transactions*, 29(1), 45-52.

Lee, Y. H. and Pinedo, M. (1997), Scheduling jobs on parallel machines with sequence-dependent setup times, *European Journal of Operational Research*, 100(3), 464-474.

Matsuo, H., Suh, C. J. and Sullivan, R. S. (1989), A controlled search simulated annealing method for the general jobshop scheduling problem, *Annals of Operations Research*, 21(85-108).

Parker, R. G., Deane, R. H. and Holmes, R .A. (1977), On the use of a vehicle routing algorithm for the parallel processor problems with sequence dependent changeover costs, *AIIE Transactions*, 9(2), 155-160.

Ovacik, I. M. and Uzsoy, R. (1995), Rolling horizon procedures for dynamic parallel machine scheduling with sequence-dependent setup times, *International Journal of Production Research*, 33(11), 3173-3192.

Uzsoy, R., http://palette.ecn.purdue.edu/~uzsoy2/Problems/parallel/parameters.html

Vakharia, A. J. and Chang, Y. (1990), A simulated annealing approach to scheduling a manufacturing cell, 40, 559-577.

신 현 준

고려대학교 산업공학과 학사
고려대학교 산업공학과 석사
고려대학교 산업공학과 박사
현재: Texas A&M University 산업공학과 Post-doc
관심분야: 휴리스틱 최적화, SCM, 정보이론, 생산계획 및 통제 시스템

김 성 식

고려대학교 기계공학과 학사
고려대학교 산업공학과 학사
S.M.U. 산업공학과 석사
S.M.U. 산업공학과 박사
현재: 고려대학교 산업시스템정보공학과 정교수
관심분야: Queueing Theory, 생산 및 재고관리 시스템, CIM, ERP, SCM 등의 기업운용 시스템 분석 및 운용 소프트웨어 개발