Journal of the Society of Korea Industrial and Systems Engineering
Vol, 27, No. 4, pp.42－48, December 2004.

선행관계를 가진 다중프로세서 작업들의 Makespan 최소화를 위한 변형타부검색

이 동 주

공주대학교 산업시스템공학과

Applying tabu search to multiprocessor task scheduling problem with precedence relations

Dong-Ju Lee

Industrial Systems Engineering, Kongju National University

This paper concerns on a multiprocessor task scheduling problem with precedence relation, in which each task requires several processors simultaneously. Meta-heuristic generally finds a good solution if it starts from a good solution. In this paper, a tabu search is presented to find a schedule of minimal time to complete all tasks. A modified tabu search is also presented which uses a new initial solution based on the best solution during the previous run as the new starting solution for the next iteration. Numerical results show that a tabu search and a modified tabu search yield a better performance than the previous studies.

Keywords : Multiprocessor tasks; Scheduling; Tabu Search

1. 서 론

전통적인 일정계획문제에서는 각 작업들이 하나의 프로세서에 의해 처리된다고 가정해 왔다. 그렇지만, 병렬처리 시스템들이 발달함에 따라, 각 작업들이 다중프로세서를 필요로 하는, 즉 하나의 작업이 하나이상의 프로세서에 의해 동시에 처리되어야 하는 문제가 대두되게 되었다. 본 논문에서는 각 작업이 다중 프로세서를 필요로 하고, 또한 각 작업들이 선행관계를 가질 때 makespan을 최소화하는 일정계획을 찾는 문제에 대해 연구해 보았다. **선행관계**란 어떤 작업들이 다른 작업들보다 먼저 혹은 늦게 처리되어야만 하는 것을 의미한다. 각 작업들을 가공할 프로세서들이 이미 정해져 있을 때 Veltman et al.[17]은 본 문제를 $P|fix_i; prec|C_{\max}$ 라고 정의하였다.

이러한 다중프로세서 작업의 일정계획문제들은 여러 학자들에 의해 널리 연구되어 왔는데, Oguz et al.[16]은 혼합흐름생산에서 makespan을 최소화하는 경우 선행관계가 없고 모든 작업시간이 상계(bounded above)되어 있을 때, 이 문제는 P문제임을 증명하고, 만약 선행관계가 있다면 NP-hard임을 증명하였다. Confessore et al.[9]은 프로세서들이 환형(ring)으로 서로 연결되어 있고 각 작업들이 인접해 있는 동수(同數)의 프로세서에 의해 처리될 때 makespan을 최소화하는 문제는 NP-hard임을 밝히고, 이런 경우 최적해값을 2배 이상 초과하지 않는 휴리스틱을 개발하였다. Baptiste[5]는 작업들이 시작시기(release date)가 있고, 동일한 작업시간이 드는 경우에 작업의 완성시간들을 최소화하는 문제($P_m|r_i, p_i=p, size_i|\sum C_i$)는 P문제임을 증명하고 이 문제를 푸는 동적계획법을 제안하였다. Amoura et al.[4]은 세대의 프로세서가 있고 각 작업들을 가공할 프로세서들이 이미 정해져 있을 때 makespan을 최소화하는 문제($P_3|fix_j|C_{\max}$)에 대한 새

로운 휴리스틱을 제안하고, 실험을 통해 Goemans[12] 과 Blazewicz et al. [6] 가 제시한 휴리스틱들과 비교하였다. Hoogeveen et al.[13]은 다중프로세서 작업 문제의 복잡도(Complexity)에 대해 연구하였다. Chen & Lee[8]는 일반적인 다중프로세서 문제에 있어 makespan의 최소화뿐만 아니라 작업할당 결정을 연구하였는데, 특히 기계가 2대인 경우와 기계가 3대중 특별한 경우에는 동적계획법으로 최적해를 찾아내었다. Blazewicz et al.[7]는 다중프로세서 문제들에 대해 지금까지 해온 연구들을 조사했다.

Kramer[15]는 분지한계법(Branch and Bound)를 이용하여 선행관계를 가진 다중 프로세서 일정계획문제의 최적해를 찾았는데, 해의 탐색시간이 너무 오래 걸린다는 단점이 있었다. 본 연구에서는 짧은 시간 내에 우수한 근사해를 구하기 위해 타부검색(TS)과 본 논문에서 제시한 변형 타부검색을 이용하였다. 타부검색은 Glover [10][11]에 의해 제안되었으며, Simulated Annealing, Genetic Algorithm등과 함께 복잡한 문제에 있어 우수한 근사해를 제공해 주는 기법으로 널리 이용되어 왔다. TS는 해를 탐색해 나가는 과정을 기억하여(타부목록에 저장) 이전에 거쳐간 해로 이동해가는 것을 예방하고, 또한 해의 순환을 방지하여 지역 최적해(local optimum)에서 다른 지역최적해로의 이동을 가능케 하므로써 빠른 시간 내에 근사해를 탐색해 낸다.

현철주[3]는 한 생산라인에서 유사한 여러 모델의 제품을 생산하는 혼합모델 조립라인에서 컨베이어 정지위험 최소화와 부품사용의 일정율 유지의 두 목적을 동시에 고려하는 파레토 최적해를 효율적으로 찾는 타부검색을 제시하였다. 신현준 외[2]는 단일기계에서 납기지연시간의 최대값을 최소화하는 일정계획을 찾는 타부검색을 제시하였다. Kim et al.[14]은 비계층적인 주문형 영상 서비스(video-on-demand) 네트워크을 설계하는데 타부검색을 적용하였다.

n개의 작업들과 m개의 기계가 있다고 할 때, 본 연구에 쓰이는 기호들은 다음과 같다.

J_i : 작업 i, i=1, ···, n
M_j : 기계 j , j = 1, ··· , m
C_i : 작업 i (J_i)의 완성시간 (completion time)
$C_{\max}$: makespan = $\max\{C_i; i=1,\cdots,n\}$
t_i : 작업 i (J_i)의 소요시간
D_i : 작업 i (J_i)를 처리하기 위한 기계군
$prec_i$: 작업 i (J_i)의 선행작업들

예를 들면, 6개의 작업들과 5개의 기계들(n=6, m=5)이 있다고 하자. 여기서 작업 i (J_i)를 처리하기 위한 기계군은 $D_1=\{M_3, M_4, M_5\}$, $D_2=\{M_1, M_4\}$, $D_3=\{M_2, M_3, M_4\}$, $D_4=\{M_1, M_5\}$, $D_5=\{M_2, M_3\}$, $D_6=\{M_5\}$ 이고, 각 작업의 소요시간은 $t_1=t_5=1$, $t_2=t_3=t_4=3$, $t_6=4$ 이다. J_1은 기계들 M_3, M_4와 M_5에 의해 동시에 1단위시간에 작업되어야 하고, J_2는 M_1과 M_4에 의해 동시에 3 단위시간에 작업되어야 한다. 또한, J_4는 J_5가 처리되기 이전에 작업되어야 하며 J_5는 J_6가 처리되기 이전에 처리되어야 한다. 자세한 사항은 <표 1>과 같다.

<표 1> 예 제

Jobs	*ti*	D*i*	prec*i*
J1	1	{M3,M4,M5}	
J2	3	{M1,M4}	
J3	3	{M2,M3,M4}	
J4	3	{M1,M5}	
J5	1	{M2,M3}	{J4}
J6	4	{M5}	{J4,J5}

만약 J_i가 J_j 이전에 처리되어야 한다면, J_i는 J_j와 **선행관계**(precedence relationship)를 가진다. 이러한 관계는 $J_i \rightarrow J_j$로 표시하기로 한다. 만약 작업들 J_i와 J_j가 동시에 처리될 수 있다면 이러한 관계는 **평행관계**(parallel relationship)로 정의하고 $J_i \| J_j$로 표시하기로 한다. 예를 들면, 만약 J_i가 M_1, M_2, M_3에 의해 처리되고, J_j가 M_4, M_5에 의해 처리되고, J_i와 J_j가 선행관계가 없다면, J_i와 J_j는 평행관계를 가졌다고 할 수 있다. 이러한 평행관계만이 해의 값을 개선할 수 있으므로, 본 연구에서는 평행관계를 가진 작업들만 고려하여 인접해들을 탐색하도록 한다.

이 논문의 구성은 다음과 같다. 2절에서는 작업 순서와 각 기계들로의 작업 할당에 대한 정보를 보관하면서, makespan의 계산을 돕는 중간행렬을 도입하고, 또한 초기해 생성 앨고리듬을 소개하였다. 3절에서는 타부검색의 실제 적용을 위한 여러 가지 패러미터와 변형 타부검색 앨고리듬을 제안하였다. 4절에서는 적용한 타부검색의 성능을 측정하기 위하여, 타부검색의 CPU 처리시간과 근사해를 20개의 예제 문제를 이용하여 Kramer의 분지한계법과 문기주[1]의 유전자 앨고리듬과 비교 분석해 보았다. 마지막으로 5절에서는 결론과 앞으로의 연구과제에 대해 간략하게 논의해 보았다.

2. 중간행렬과 초기해

작업 순서와 각 기계들로의 작업 할당에 대한 정보를 보관하면서, makespan의 계산과 이웃 해를 쉽게 검색하기 위해 "중간행렬"이 소개되었다. 중간행렬의 각 열은 하나의 작업으로 각 행은 하나의 기계에 해당된다. 중간행렬을 이용하여 초기해를 생성하는 앨고리듬은 다음과 같다.

초기해 생성 앨고리듬

단계 0. $j = 0$.

단계 1. $j = j + 1$. 선행관계를 유지하는 작업을 중간행렬의 열 j에 배치한다. 중간행렬에서 열 j에 위치한 작업을 a_j라고 하자. 중간행렬에서 a_j와 평행관계를 가진 작업 a_k를($k < j$) 찾는다. 여기서 a_j는 a_k와 평행관계를 가진 작업들 중 가장 가까운 열에 위치한 작업이다. a_k가 없다면 단계 1로 되돌아가고, 아니면 단계 2로 계속한다.

단계 2. 중간행렬에서 열 j 와 k사이에 위치하고 a_k와 선행관계를 가진 작업들을 찾는다.

단계 3. a_j 앞에 a_k와 단계 2에서 찾은 a_k와 선행관계를 가진 작업들을 배치한다.

단계 4. $j \leq n$이면 단계 1로 되돌아가고 아니면 종료한다. 여기서 n은 총 작업 수이다.

기본적으로 각 작업은 작업의 번호 순서대로 하나씩 왼쪽에서 오른쪽으로 할당되는데, 현재 할당할 작업 앞에 최소한 하나이상의 평행관계를 가진 작업이 있다면 현재 할당할 작업과 선행관계를 가진 작업들(현재 할당할 작업을 포함)을 가장 가까이에 위치한 평행관계를 가진 작업직전에 할당한다. 즉, 중간행렬에서 열 k에 위치한 작업을 a_k라고 하고, 현재 k번째 작업을 할당하려고 한다고 하자. 또한, a_k와 평행관계를 가진 작업들 중 가장 가까이에 위치한 작업을 $a_j (j < k)$라고 하자. a_j와 a_k 사이에 위치한 작업들 중 a_k와 선행관계를 가진 작업들 모두를 a_k와 함께 a_j 직전에 할당한다. 그 다음에는 k+1번째 작업을 같은 방법으로 할당한다. 이런 식으로 모든 작업을 할당할 때까지 계속한다. 2절에서 보여 준 예제를 통하여 중간행렬을 이용한 초기해 생성을 <그림 1>에, 생성된 초기해의 간트 차트(Gantt chart)를 <그림 2>에 나타내었다.

	열1
M1	
M2	
M3	J1
M4	J1
M5	J1

	열1	열2
M1		J2
M2		
M3	J1	
M4	J1	J2
M5	J1	

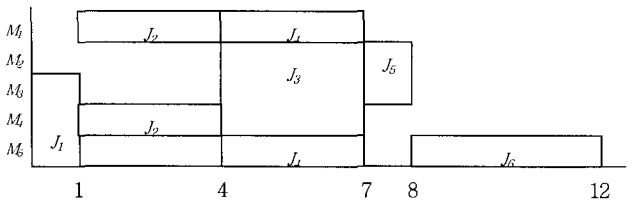
	열1	열2	열3
M1		J2	
M2			J3
M3	J1		J3
M4	J1	J2	J3
M5	J1		

	열1	열2	열3	열4
M1		J2	J4	
M2				J3
M3	J1			J3
M4	J1	J2		J3
M5	J1		J4	

	열1	열2	열3	열4	열5
M1		J2	J4		
M2				J3	J5
M3	J1			J3	J5
M4	J1	J2		J3	
M5	J1		J4		

	열1	열2	열3	열4	열5	열6
M1		J2	J4			
M2				J3	J5	
M3	J1			J3	J5	
M4	J1	J2		J3		
M5	J1		J4			J6

<그림 1> 중간행렬로 나타낸 초기해 생성 앨고리듬의 예.



makespan 12

<그림 2> 초기해에 대한 Gantt chart

3. 타부검색(Tabu Search, TS)에 의한 해의 개선

타부검색은 초기해의 우수함의 정도, 현재 해에서 이웃해로 옮겨가게 하는 이동(Move)의 효율성, 탐색한 해를 기록(history)하는 타부목록(tabu list)의 크기에 많은 영향을 받는 것으로 알려져 있다. 본 연구에서는 좋은 초기해에서 시작하면 우수한 근사해를 찾는다는 데 착안하여 우선 일반적인 타부검색을 통하여 지역최적해들을 찾고 그들의 좋은 특성을 보존하는 새로운 초기해를 발생시켜 타부검색의 초기해로 다시 대입하여 해를 개선시키고 이것을 반복하므로 써 일반적인 타부검색보다 나은 해를 찾았다. 또한, 본 연구에서 다루는 한 작업-다중기계에서 makespan을 최소화하는 문제들은 평행관계를 가진 작업들을 인접하게 둘 때에만 makespan이 감소하므로 평행관계를 가진 작업들만 이동(move)에 고려하므로 써 이동의 효율을 향상시켰다.

3.1 이동(move)과 이웃해(Neighborhood)

이동(move)이란 어떤 해에서 다른 해로 이동해가는 규칙 혹은 함수이다. 본 연구에 쓰인 이동을 다음과 같은 예를 통해 설명해 보겠다. 먼저 *X, R, P*를 각각 작업순서(일정계획), 선행관계, 평행관계라고 하자. 현재의 작업순서를 $J_t < J_i < J_u < J_j < J_v < J_k < J_l \in X$ 이라고 하고 여기서 $J_t \rightarrow J_u \rightarrow J_v \rightarrow J_k \in R$이고 $J_i \| J_j \| J_k \| J_l \in P$ 이라고 하자.

임의로 작업 J_k가 선택되었다고 할 때, J_k와 평행관계를 가진 작업들을 현재 작업순서에서 찾는다면 J_i, J_j와 J_l이다. 임의로 선택된 J_k와 평행관계를 가진 작업들 중 J_k보다 이전에 할당된 작업들만 본 연구에서는 이동으로 고려하므로 J_i와 J_j만 고려한다. 작업 J_k 를 작업 J_i 혹은 J_j 직전에 삽입하기 위해서는 J_i 혹은 J_j와 선행관계를 가진 다른 작업들도 같이 삽입하여야 한다. 그러므로, 다음과 같은 두 가지의 가능한 경우가 있다. :

$$X_1 = J_t < J_u < J_v < J_k < J_i < J_j < J_l \quad \text{or}$$
$$X_2 = J_t < J_i < J_u < J_v < J_k < J_j < J_l$$

이 두 가지의 가능한 작업순서(일정계획), X_1과 X_2 중 더 작은 makespan을 가지는 작업순서를 다음의 해로 선택하고, 이렇게 다음 해로 옮겨가는 규칙을 이동이라고 한다.

3.2 타부목록(Tabu List)

$T = (T_1, \cdots, T_{maxleng})$를 길이 *maxleng*인 타부목록이고, T_j 는 현재의 타부목록에 있는 이동들이라고 하자. 만약 현재의 이동 *s*가 현재의 타부목록에 없는 이동이기에 타부목록 T에 추가하려고 한다면, 타부목록에 있는 이동들을 왼쪽으로 옮기고, 제일 마지막, 즉 *maxleng*에 현재의 이동 s를 추가한다. 순환 현상을 예방하고 근사해를 찾기 위해 탐색기록은 이 타부목록에 저장된다.

3.3 열망함수(The aspiration function)

열망 함수는 현재해로부터 생성된 특정 이웃해가 타부에 의해 금지되고 있는 해일지라도 그때까지의 최상해보다 더 나은 해를 제공한다면 허용하게 하는 것이다. 본 문제($P|fix_i; prec|C_{\max}$)는 makespan을 최소화하는 것을 목표로 한다. 열망함수는 아래와 같다.

$$A = \{s \in S(x) \cap T : C_{\max}(X(s)) < C^*\}$$

여기서 $S(x)$는 가공순서 $x \in X$에 적용된 모든 가능한 이동(move)들의 집합이고, $X(s)$는 이동 *s*를 적용하므로써 얻어진 가공순서 $x \in X$를, $C_{\max}(X(s))$는 가공순서 $X(s)$의 makespan을, 마지막으로 *T*는 타부목록(tabu list)을 나타낸다.

3.4 새로운 초기해 생성

다양한 초기해를 생성하여 타부검색을 한다면 서로 다른 영역에 있는 지역최적해를 찾아내어 그 중 나은 지역최적해를 탐색해 낼 수 있겠지만, 다양한 초기해를 생성해 내는데 드는 시간을 무시할 수 없으며 또한 타부검색에 의해 개선된 지역최적해를 사용하지 않으므로

써 정보의 손실을 초래하게 된다. 그러므로 본 절에서는 타부검색에 의해 탐색된 지역최적해의 좋은 구조(정보)를 유지한 새로운 해를 생성하는 방법에 대해 논의해 보기로 한다.

즉, 타부검색에 의해 탐색된 근사해의 중간행렬에서 어떤 작업이 좌우의 인접한 작업들 중 평행관계를 가지는 작업이 없다면 제일 뒷 열로 옮기고 그렇지 않다면 그대로 두면서 새로운 해를 생성해 내는 것이다. 이렇게 생성해 낸 해는 다시 타부검색의 초기해로 대입해서 개선해 나가는 것이다. 예를 들어 <그림 1>에 보여진 해가 타부검색에 의해 탐색된 해라고 가정하자. 첫 열의 J_1은 인접한 J_2와 평행관계를 가지지 않기에 뒤로 옮긴다. 다시 J_2는 인접한 J_4와 평행관계를 가지지 않기에 뒤로 옮긴다. J_4는 J_3와 평행관계를 가지고 있으므로 그대로 둔다. J_3는 또한 인접한 J_4와 J_5중 J_4와 평행관계를 가지므로 그대로 둔다. J_5는 인접한 J_3와 J_6중 J_6와 평행관계를 가지므로 그대로 둔다. 이렇게 해서 완성된 새로운 초기해는 <그림 3>과 같다.

	열 1	열 2	열 3	열 4	열 5	열 6
M1	*J4*					*J2*
M2		*J3*	*J5*			
M3		*J3*	*J5*		*J1*	
M4		*J3*			*J1*	*J2*
M5	*J4*			*J6*	*J1*	

<그림 3> 변형에 의해 생성된 초기해

변형 타부검색 알고리듬은 다음과 같다.

단계 1. 초기해 생성 알고리듬에 의해 초기해를 생성한다.

단계 2. a_k를 중간행렬에서 열 k에 위치한 작업이라고 하자($k=1, \cdots, n$). $k=1$로 *iteration* =1로 *phase2* =1로 정하자.

단계 3. 중간행렬에서 열 k 이전에 있는 작업들 중 작업 a_k와 평행관계를 가진 작업들을 찾고, 이러한 작업들, 즉 이동(move)중에서 타부목록에 없거나 있더라도 열망함수를 만족하는 이동들을 찾는다.

단계 4. 3단계에서 찾은 이동들에 의해 생성된 이웃해 중에서 가장 최소의 makespan을 가지는 이웃해를 찾고, 그 이웃해($X(s')$)와 관련된 이동(s')이 타부목록에 없다면 타부목록에 기록한다. $k=k+1$로 정하고 열 k가 중간행렬의 마지막 행렬이라면 단계 5로, 아니면 단계 3으로 되돌아간다.

단계 5. C^*를 현재까지의 최소의 makespan이라고 할 때, 만약 $C^* > C_{\max}(X(s'))$, C^*에 $C_{\max}(X(s'))$를 대입하고, *phase2* = 1로 하고, 현재의 작업순서 $X(s')$을 X^*에 저장한다. 최대의 허용가능한 *iteration*수를 *maxiter*이라고 할 때, *iteration* < *maxiter* 이면 *iteration* = *iteration* + 1로 하고 단계 2로, 아니면 단계 6으로 간다.

단계 6. 3.4절에 제시된 새로운 초기해 생성 최대횟수를 *maxphase2* 라고 하자. *phase2* < *maxphase2* 라면, 3.4절에 제시된 기법에 의해 X^*로 부터 새로운 해를 생성하고, 타부목록을 비우고, *iteration* = 1로 하고, 단계2로 되돌아가고, 아니면 종료한다.

본 연구에 사용된 타부검색 알고리듬은 변형 타부검색 알고리듬에서 새로운 초기해 생성을 제외하면 동일하므로 생략한다.

4. 결 과

타부검색과 변형 타부검색의 결과는, 문기주 외[1]의 유전자 알고리듬에 의한 결과, Kramer[15]의 분지한계법(branch and bound)의 결과와 비교되었다. Kramer의 분지한계법은 최적해를 찾지만 한 문제를 푸는 데 평균 178.3분이란 오랜 시간이 걸린다. Kramer의 분지한계법은 'C 언어'로 구현되어 Sun 4/20 Workstation에 의해 실행되었고, 나머지 변형 타부검색, 타부검색, 유전자 알고리듬은 역시 'C 언어'로 구현되어 800MHz p/c로 실행되었다. 각 기법들의 성능을 비교하기 위해 rdata 와 vdata로 불리는 두 가지의 데이터 종류가 사용되었는데, 각 종류의 데이터는 10개의 예제들이 포함되어 있다. 각 예제들은 5대의 기계로 구성되어 있고, rdata에 대해서는 30개의 작업들이, vdata에 대해서는 40개의 작업들로 구성되어 있다. 선행관계는 각 5개의 작업들이 사슬관계로 연결되어 있다. rdata에서는 각 작업이 요구하는 평균 기계수는 2대이고 vdata에서는 2.5대이다.

타부검색과 변형 타부검색에 영향을 미치는 많은 패러미터들이 있는데, 본 실험에 쓰인 요소들은 다음과 같다 :

maxleng : 타부목록의 총 길이.

maxiter : 총 반복횟수.

no_phase2 : 변형 타부검색에서 새로운 해를 생성해내는 총 횟수.

유전자 앨고리듬에 쓰인 패러미터들은 다음과 같다.
Popsize : population의 크기.
maxiter : 총 반복횟수.
PM : Mutation을 할 확률.
PC : Crossover를 할 확률.

rdata에 대해 사용된 타부검색(TS)의 패러미터들은 maxleng=22, maxiter=3000 이며, 유전자 앨고리듬(GA) popsize=26, maxiter=6000, PM=0.6, PC=0.2이다. 또한, vdata에 대해서 사용된 타부검색의 패러미터들은 maxleng=24, maxiter=3000이며, 유전자 앨고리듬은 popsize=32, maxiter=4000, PM=0.6, PC=0.2이다. 변형타부검색의 유전자 앨고리듬, 타부검색, 변형 타부검색 각각에 의한 근사해와 최적해와의 평균 차이를 아래와 같이 계산하였다.

$$\text{평균오차율} = \frac{\text{근사해} - \text{최적해}}{\text{최적해}} \times 100(\%)$$

위 식에 의거해 타부검색(TS)과 유전자 앨고리듬(GA)의 계산결과를 <표 2>에 나타내었다.

<표 2> 데이터 종류와 기법에 따른 평균오차율

		rdata	vdata
GA	평균오차율(%)	5.23	11.08
	평균시간 (초)	6.7	7.2
TS	평균오차율(%)	1.85	1.87
	평균시간 (초)	7.7	10.7

maxiter과 no_phase2의 값이 변화하고, 다른 패러미터들의 값은 타부검색과 동일하게 할 때의 변형 타부검색의 결과를 <표 3>에 나타내었는데, 모든 결과는 유전자 앨고리듬보다 낮은 평균오차율을 보였다. vdata의 결과는 maxiter이 200이상이고, no_phase2가 11이하일 때 타부검색보다 낮은 평균오차율을 나타내었다. rdata의 경우에는 maxiter이 300이상이고 500이하이며, no_phase2가 5이상이고 8이하일 때 타부검색보다 낮은 평균오차율을 나타내었다. 결과로 보아 변형타부검색은 no_phase2가 5에서 8사이 정도이고 maxiter은 여러 지역 최적해를 탐색해 볼 수 있을 정도로 충분하도록 할 때 가장 좋은 결과를 보여주는 것으로 예상된다.
유전자 앨고리듬과 타부검색의 차이는 커 보이고, 상대적으로 타부검색과 변형 타부검색의 차이는 적어 보이지만, 최적해에 다가갈수록 해를 향상시키기가 점점 더 어려워진다는 사실을 염두에 둔다면 타부검색과 변형 타부검색의 차이도 적다고 할 수는 없을 것이다.

5. 결 론

선행관계를 가진 다중 프로세서 일정계획문제 ($P|fix_i; prec|C_{\max}$)를 타부검색과 변형타부검색을 적용하여 보았고, 또한 그 결과를 문기주외[1]의 유전자 앨고리듬과 Kramer[15]의 분지한계법과 비교하여 보았다. Kramer의 분지한계법은 최적 해를 찾을 수 있지만, 너무 오랜 컴퓨터 계산시간 (최악의 경우 5시간동안 최적 해를 못 찾을 수도 있다.)을 필요로 한다는 단점이 있다. 본 연구에서 적용한 타부검색은 훨씬 짧은 시간내에(평

<표 3>max_iter과 no_phase2의 변화에 따른 변형타부검색의 결과

rdata				vdata			
maxiter	no_phase2	평균오차율(%)	평균시간(초)	maxiter	no_phase2	평균오차율(%)	평균시간(초)
100	24	4.43	6.4	50	42	2.67	7.7
200	12	2.21	7.4	100	21	2.50	7.8
300	8	1.76	7.6	200	11	1.82	9.1
350	7	1.20	8.6	300	7	1.09	10.4
400	6	1.92	7.8	350	6	1.06	10.7
450	5	1.59	8.5	400	5	0.97	9.8
500	5	1.55	8.5	450	4	1.34	8.7
600	4	1.99	8.8	500	4	1.31	9.7
700	3	1.87	7.5	600	3	1.41	8.5
800	3	1.48	7.5	700	3	1.32	10.3
1200	2	1.95	7.5	1000	2	1.37	10.0

균 11초 이내에) 최적해에 가까운 근사해를 제공하였으며, 비슷한 시간 내에 문기주외[1]의 유전자 앨고리듬보다 우수한 근사해를 제공하였다. 또한, 본 연구에서는 우수한 초기해에서 시작하면 우수한 근사해를 찾는다는데 착안하여 일반적인 타부검색을 통해 찾은 근사해의 좋은 특성을 보존하는 새로운 초기해를 발생시켜 타부검색에 다시 대입하여 해를 개선시키는 방법을 반복하므로 써 일반적인 타부검색보다 나은 해를 찾았다.

미래의 연구과제로는 본 논문에서는 평행관계에 기초한 이동(move)을 적용하였는데, 이웃해의 탐색범위를 줄이는 좀 더 우수한 이동에 대한 연구가 필요하다고 하겠다. 또한, 국내에는 다중 프로세서 일정계획문제에 대한 연구가 많지 않은데, 다중 프로세서 시스템이 계속 발달해 감에 따라 그 중요성은 점점 더 커진다하겠다. 그러므로, 좀 더 다양한 다중프로세서 일정계획문제에 대한 연구가 시급하다고 하겠다.

참고문헌

[1] 문기주, 오현승, 이동주; "A Genetic Algorithm for Scheduling Multiprocessor Tasks on Dedicated Processor", 대한설비관리학회지, 8(2) : 5-15, 2003.

[2] 신현준, 김성식, 고경석; "작업투입시점과 순서 의존적인 작업준비시간이 존재하는 단일 기계 일정계획 수립을 위한 Tabu Search", 대한산업공학회지, 27(2) : 158-168, 2001.

[3] 현철주; "혼합모델 조립라인의 생산순서 결정을 위한 다목적 타부검색", 대한설비관리학회지, 7(3) : 45-57, 2002.

[4] Amoura, A.K., Bampis, E., Manoussakis, Y., and Tuza, Z.; "A comparison of heurisitics for scheduling multiprocessor ta나s on three dedicated processors", Parallel Computing, 25; 49-61, 1999.

[5] Baptiste, P.; "A note on scheduling multiprocessor tasks with identical processing times", Computers & Operations Research, 30; 2071-2078, 2003.

[6] Blazewicz, J., Dell'Olmo, P., Drozdowski, M., and Speranza, M..G.,; "Scheduling multiprocessor tasks on three dedicated processors", *Information Processing Letters*; 41; 275-280, 1992.

[7] Blazewicz, J., Drozdowski, M., and Weglarz, J.; "Scheduling Multiprocessor Tasks-A Survey", *Microcomputer Applications*, 13(2); 89-97, 1994.

[8] Chen, J., and Lee, C.-Y.; "General Multiprocessor Task Scheduling", *Naval Research Logistics* 46; 57-74, 1999.

[9] Confessore, G., Dell'Olmo, P., and Giordani, S.; "Complexity and approroximation results for scheduling multiprocessor tasks on a ring", Discrete Applied Mathematics, 133; 29-44, 2004.

[10] Glover, F., " Tabu search, Part 1", ORSA journal on Computing, Vol. 1, No. 3, pp.190-206, 1989.

[11] Glover, F., "Tabu search,Part 2", ORSA journal on Computing, Vol. 2, No. 1, pp.4-32, 1990.

[12] Goemans, M. X.; "An approximation algorithm for scheduling on three dedicated machines", Discrete Applied Mathematics, 61; 49-59, 1995.

[13] Hoogeveen, J.A., van de Velde, S.L., and Veltman, B.; "Complexity of sheduling multiprocessor tasks with prespecified processor allocations", Discrete Applied Mathematics, 55; 259-272, 1994.

[14] Kim, Y.K., Kim, J.Y., and Kang, S.S.; "A Tabu Search Approach for Designing a Non-Hierarchical Video-on-Demand Network Architecture", *Computers ind. Engng.*, 33(3-4); 837-840, 1997.

[15] Kramer, A.; "Branch and Bound method for scheduling problems of multiprocessor tasks on Dedicated Processors", Ph.D. dissertation, Department of Mathematics/Informatics, Universitat Osnabruck. 1995.

[16] Oguz, C., Y. Zinder, V.H. Do, A. Janiak, and M. Lichtenstein; "Hybrid flow-shop scheduling problems with multiprocessor task systems", European Journal of Operational Research, 152; 115-131, 2004.

[17] Veltmann, B., B.J. Lageweg, and J.K. Lenstra; "Multiprocessor Scheduling with Communication Delays". Parallel Computing, 16; 173-182, 1990.