工業經營學會誌 第18卷 第35輯 1995年 9月 107

동일한 병렬기계 일정계획에서 평균지연시간의 최소화를 위한 Tabu Search 방법
- Applying Tabu Search to Minimize Mean Tardiness in the Parallel Machine Scheduling -

전 태 웅*
Chun, Tae Woong
강 맹 규**
Kang, Maing Kyu

Abstract

This paper proposes the Tabu Search algorithm to minimize mean tardiness in the parallel machine scheduling problem. The algorithm reduces the computation time by employing restricted neighborhood and produces an efficient solution in this problem.

1. 서 론

본 연구에서 다루는 문제는 n 개의 작업과 m 대의 기계가 있는 동일한 병렬기계 시스템(identical parallel machine system)에 있어서 최적 일정계획(scheduling)을 구하는 것이다. 동일한 병렬기계 시스템이란, 시스템 내에 m 대의 기계가 있고 처리될 한 작업의 가공시간이 m 대의 기계에서 동일하며, 작업이 완료되면 시스템을 떠난다. 예를 들면, 컴퓨터 시스템에서 병렬처리기로 프로그램을 동시에 처리하고, 제조공정에서 병렬기계로 부분품을 동시에 가공한다. 이러한 병렬처리는 최근에 시스템의 효율과 생산성을 위해 많이 사용되고 있다[5, 8].

본 연구에서 사용하는 일정계획에 대한 수행도는 평균 지연시간(mean tardiness; MT)의 최소화이다. 지연시간이란 작업의 납기시간보다 완성시간이 초과되는 시간으로 정의된다[2]. 지연시간 최소화 문제의 계산량은 병렬기계 일정계획 문제뿐 아니라 단일기계 일정계획 문제에서도 NP-hard 이다[5].

Wilkerson 과 Irwin [11]은 단일기계 일정계획 문제에서 EDD (earliest due date) 방법에 이웃교환 방법을 이용하는 휴리스틱 방법을 개발하고 이를 병렬기계 일정계획에 적용하였다. Ho 와 Chang [7]은 단일기계 일정계획 문제에서 EDD 와 SPT (shortest processing time) 방법을 결합시킨 TPI(traffic priority index) 방법을 개발하고 이를 병렬기계 일정계획에 적용하였다.

* 조선대학교 산업공학과
** 한양대학교 산업공학과

최근에 범용적인 알고리듬인 Tabu Search(TS) 알고리듬을 일정계획 문제와 같은 조합 최적화 문제에 적용하여 우수한 해를 찾고 있다. Languna 등 [9]의 단일기계 일정계획 문제, Barnes와 Languna[3]의 복수기계 일정계획 문제, Adenso-Diaz[1]과 Taillard[10]의 흐름생산 일정계획 문제, Dell'Amico 와 Trubian[6] 및 Barnes 등[4]의 개별생산 일정계획 문제 등이 있다.

본 연구의 목적은 병렬기계 일정계획 문제에서 평균 지연시간을 최소화하는 TS 알고리듬을 개발하고자 한다. 2장에서는 본 문제의 수리적 모형을다루고 3장에서는 본 연구와 관련된 TS 알고리듬을 간단히 소개한다. 4장과 5장에서는 본 연구에서 개발한 TS 알고리듬의 설계 및 제한된 이웃해집단의 구성방법을 보이고 6장에서는 실험결과를 요약한다.

2. 수리적 모형

본 연구의 일정계획 문제에 대해 가정한 사항은 다음과 같다. 기계 i 에서 수행될 j 번째 작업 J_{ij} (i=1, ... , m, j=1, ... , n_i)의 가공시간 P_{ij} 와 납기시간 D_{ij} 는 주어진다(여기서 n_i 는 기계 i에 할당된 작업수를 나타내며 $\sum^{m}_{i=1} n_i = n$ 이다.). 그리고 하나의 작업이 하나의 기계에 할당되면 그 작업은 그 기계에서 작업이 완료된다. 본 연구의 수리적 모형은 식 (1)-(4)로 표현된다.

최소화 $$MT = (1/n)\sum^{m}_{i=1} \sum^{n_i}_{j=1} T_{ij} \quad X_{ij} \tag{1}$$

제약조건 $$T_{i,j} = \max(0, C_{ij} - D_{ij}) \quad (i=1, \dots , m), (j=1, \dots , n_i) \tag{2}$$

$$\sum^{n_i}_{j=1} n_i = n \tag{3}$$

$$X_{ij} = \begin{cases} 1, & \text{if 작업 j가 기계 i 에 할당되면} \\ 0, & \text{그렇지 않으면} \end{cases} \tag{4}$$

여기서, $C_{i,j}$ = 작업 J_{ij} 의 완료시간

식 (1)에서 T_{ij} 는 작업 J_{ij}의 지연시간으로 식 (2)로 표현되며, 작업 완료시간에 대해서 비선형으로 나타나므로 MT를 최소화시키는 최적해를 구한다는 것이 매우 어렵다. 이 문제는 NP-hard 이기 때문에 최적해를 얻기 위해서는 동적계획법이나 분지한계법을 이용할 수 있지만 계산시간과 컴퓨터 기억장소가 많이 필요하다. 그러므로 이 문제를 해결하기 위해서는 좋은 해를 제공하는 효율적인 휴리스틱 방법이 바람직하다[7].

3. TS 알고리듬 탐색절차

TS 알고리듬은 이웃해 탐색방법(neighbour search method)과 탐색절차가 유사하므로 먼저 이웃해 탐색방법의 탐색절차를 간단히 설명한다..

3.1 이웃해 탐색방법

다음과 같은 최적화 문제가 있다고 하자.

최소화 $F(s)$ (5)

제약조건 $s \in X$

여기서 X는 해의 공간이며, F(s)는 목적함수이다. 초기해 $s_0 \in X$ 를 현재해 s로 하여 탐색을 시작한다. 탐색의 매 단계마다 이 s를 이동(move)하여 이웃해집단 N(s)를 생성하고, 이웃

해 $s_i \in N(s)$ 중에서 $F(s_i)$ 가 최소가 되는 s' 를 선택한다. 이때 만약 $F(s') < F(s)$ 이면 s' 를 현재해로 대체하여 탐색을 계속하고, 그렇지 않으면 탐색을 끝낸다. 이웃해 탐색방법에서 찾아진 해는 최적해가 보장되지 않는다. 여기서 이동이란 s에서 s_i를 생성하는 절차로 일정계획 문제에서는 일반적으로 교환이동(swap move)이 많이 사용된다. 예를 들어, 단일 기계 일정계획 문제에서 작업 J_i (i=1, ... , n)가 나열되어 있을 때 두 작업 J_i 와 J_j (j=1, ... , n, $j \neq i$) 를 교환하여 이웃해를 생성하는 방법이다..

3.2 TS 알고리듬의 탐색절차

TS 알고리듬의 기본적인 탐색절차는 이웃해 탐색방법과 유사하다. 즉, 초기해 $s_0 \in X$ 를 현재해로 탐색을 시작하여 이웃해집단 N(s)를 생성하고 그 중에서 $F(s_i)$ 가 최소가 되는 s' 를 선택한다. 이때 $F(s') < F(s)$ 이면 s' 을 현재해로 하여 탐색을 진행하는 절차는 같다. 그러나 TS 알고리듬은 선택한 s'가 $F(s') > F(s)$ 일지라도 탐색공간을 넓히기 위하여 s'를 현재해로 하여 탐색을 계속한다. 그리고 해의 반복이나 사이클링을 방지하기 위하여 s'를 일정기간 동안 기억장소(tabu list) T에 저장하여 새로 찾아진 s'가 T안의 해가 되지않도록 한다. 또한 더 나은 해를 찾기 위하여 새로 찾아진 s'가 T안의 해일지라도 s' 를 선택할 수 있는 기준(이를 희망기준(aspiration criteria)이라고 함) 을 사용하여 s'가 이 기준을 만족하면 이 s' 를 선택한다. TS 알고리듬은 이상과 같은 탐색절차를 일정 횟수 동안 반복하여 해를 찾는 방법이다.

4. 제안하는 TS 알고리듬

4.1 이동방법

그림 1은 본 연구에서 제안하는 TS 알고리듬의 이동방법을 설명하기 위한 예로서, TS 알고리듬을 m=3, n=10 인 병렬기계 일정계획 문제에 적용하여 해를 탐색할 때 찾아진 하나의 현재해 s를 나타낸다.

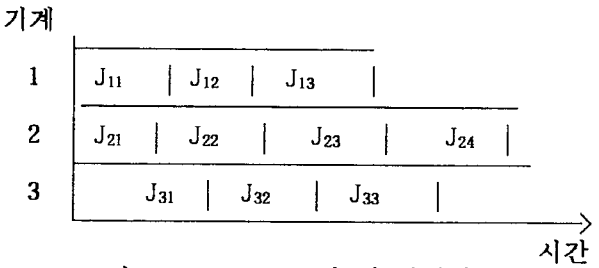

그림 1. m=3, n=10 일 때 현재해

본 연구는 그림 1과 같은 s에서 이웃해 s_i를 생성하는 이동방법으로 다른 두 기계에 있는 작업을 교환시키는 교환이동을 사용한다. 예를들어, s에서 임의로 J_{11}을 택하여 이동하면 교환되는 작업들의 쌍은 (J_{11}, J_{21}), (J_{11}, J_{22}), (J_{11}, J_{23}), (J_{11}, J_{24}), (J_{11}, J_{31}), (J_{11},J_{32}) 및 (J_{11}, J_{33}) 가 되며 이 경우 생성되는 이웃해집단의 크기는 66 개가 된다. 이를 일반화하면 n 개의 작업에 대한 이웃해집단의 크기는 $n^2 - \sum^m_{i=1} n_i^2$ 이 된다. 현실 문제에 TS 알고리듬을 적용하면 이웃해의 크기가 매우 커지므로 계산시간이 많이 소요된다.

4.2 단일기계 일정계획 방법

그림 2 는 현재해를 교환이동할 때 기계 k의 J_{ki} 과 기계 x의 J_{xj} 가 교환 대상이 된 현재해를 나타낸다.

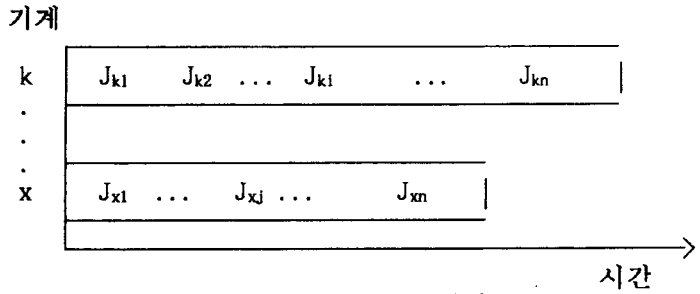

그림 2. J_{ki} 와 J_{xj} swap 전 현재해

이 두 작업을 교환하면 J_{ki}는 기계 x 의 J_{xj} 의 위치, J_{xj} 는 기계 k 의 J_{ki}의 위치로 와서 하나의 이웃해를 생성한다. 이때 두 작업은 MT의 최소화가 보장된 교환이 아니고 단순히 새로운 해를 얻기 위한 것이므로 MT가 최소화되도록 각각의 기계에 대한 작업순서를 다시 정해야 한다. 따라서, 현재해를 교환이동하여 작업들이 교환되더라도 교환된 기계에 대해서 작업순서를 최적화하는 단일기계 일정계획 알고리듬을 적용해야 하므로 매우 많은 계산시간이 소요된다. 이 단계에서 해의 효율을 위해 EDD 방법이나 Ho 와 Chang 알고리듬[7]과 같은 기존의 휴리스틱 알고리듬을 사용할 수 있다.

그러나 본 연구는 더 범용적인 알고리듬의 개발하고 해의 값을 개선하기 위하여 J_{xj}를 그림 3과 같이 기계 k에 나열되어 있는 작업들(J_{k1}, J_{k2}, J_{k3}, ... , J_{kL}) 앞에 삽입시키는 방법을 사용하여 기계 k에서 발생가능한 n_k개의 해를 얻고, n_k 개의 해 중 MT를 최소화하는 해를 찾아 그 기계에서의 작업순서로 한다. 본 연구에서 제안하는 TS 알고리듬에서는 단일기계 일정계획을 위해 위와 같은 삽입방법을 사용한다.

해						
1	J_{xj}	J_{k1}	J_{k2}	J_{k3}	....	J_{kn_k}
2	J_{k1}	J_{kj}	J_{k2}	J_{k3}	....	J_{kn_k}
.						
.						
.						
n_k-1	J_{k1}	J_{k2}	J_{k3}	....	J_{xj}	J_{kn_k}
n_k	J_{k1}	J_{k2}	J_{k3}	....	J_{kn_k}	J_{xj}

그림 3. 기계 k에 J_{xj} 가 삽입되어 발생 가능한 n_k 개의 해

4.3 제안하는 TS 알고리듬의 구성요소

본 연구에서는 tabu list 의 크기를 일반적으로 사용하는 7로 하고 희망기준으로서 새로 찾아진 해의 값이 현재의 최선해보다 좋으면 이 해를 현재해로 선택한다. 초기해로 본 연구는 병렬기계 일정계획에 일반적으로 사용하는 최소 납기규칙에 의한 최소 작업부하 할당방법 (smallest-load machine rule)을 적용한다. 이 방법은 임의로 나열되어 있는 n 개의 J_i (i=1, ..., n) 를 m 대의 기계에 할당할 때 i=1 부터 n 까지 가장 작업부하가 적은 기계에 J_i를 할당하는 방법이다[7, 11]. 본 연구에서 제안하는 TS 알고리듬은 그림 4 와 같다.

Notations: X: set of feasible solution.
f(): MT = $(1/n)\sum_{i=1}^{m} \sum_{j=1}^{n_i} T_{ij}$ where $T_{ij} = \max(0, C_{ij}-D_{ij})$.
nb_max: m.
nb_iter: the number of current iteration.
best_iter: the number of the iteration which has given the last improvement.
best_sol : the best solution in current search process.
Problem: Find x (x ∈ X) such that f(x) is *minimum*.
Algorithm:
Initialisation:
- generate the initial solution s (s ∈ X) by EDD sequence and smallest-load machine rule;
- best_sol:= f(s);
- nb_iter := 0;
- best_iter := 0;
- T :={ }

while (nb_iter - best_iter< nb_max)
- generate neighbours s_i of current solution s with move (s -> s_i) ∉ T or with $f(s_i) \le$ best_sol;
- $f(s_i)$ is caculated by insert method for one machine scheduling;
- let s' be the best neighbour generated;
- update tabu list;
- if f(s')< best_sol then
  - best_iter:=nb_iter;
  - best_sol:=f(s');
- s:=s';
- nb_iter:=nb_iter+1;

end while

그림 4. 제안하는 TS 알고리듬

5. 제한된 이웃해의 구성

일반적인 TS 알고리듬을 적용하여 본 문제의 해를 탐색할 때 생성가능한 이웃해집단의 크기가 매우 크므로 많은 계산시간이 소요된다. 본 연구는 현재해에서 생성가능한 이웃해집단의 크기를 제한하는 제한된 이웃해집단을 구성하고 해의 값과 계산시간의 관계를 규명하여 타당한 제한된 이웃해집단을 결정하여 계산시간을 절약한다. 이를 위하여 본 연구 교환될 k 기계
의 작업과 교환하려는 x 기계의 작업순서와 작업개수를 고려하여 이웃해를 만드는 이웃해집단을 구성한다(그림 5 참조). 이때 교환하려는 x기계의 작업개수를 R 이라고 하자. 그림 5는 R를 3으로 할 경우에 본 연구에서 구성한 제한된 이웃해 집단의 구성 형태이다. 그림 5에서 J_{kj}의 교환 대상의 작업은 기계 i (i=2, ... , m) 의 J_{xj-1}, J_{kj}, J_{kj-2} 가 된다.

기계	작 업 순 서								
번호	1	. . .	j - 2	j - 1	j	j + 1	j + 2	. . .	n_i
.		.							
k	J_{k1}	. . .	J_{kj-2}	J_{kj-1}	J_{kj}	J_{kj+1}	J_{kj+2}	. . .	
.									
x	J_{x1}	. . .	J_{xj-2}	J_{xj-1}	J_{xj}	J_{xj+1}	J_{xj+2}	. . .	
:									
m	J_{m1}	. . .	J_{mj-2}	J_{mj-1}	J_m	J_{mj+1}	J_{mj+2}	. . .	

그림 5. R=3 인 경우의 제한된 이웃 해집단의 구성

6. 알고리듬 수행결과 및 분석

본 연구에서 제안한 TS알고리듬을 Ho 와 Chang 연구[7]와 비교하기 위해 그들이 사용한 n=50 일 때 m=2, 3, 4, n=100 일 때 m=5, 6, 7 로 하여 각각의 문제를 20 회 반복하여 얻은 해의 값과 계산시간을 구하였으며, 실험에 사용된 사용한 문제는 가공시간 P_{ij} =Uniform[1, 25], 납기시간 D_{ij}=Uniform [1, $2n\overline{P}/4.5m$] 에 의하여 발생시켰다.

또한 본 연구는 제한된 이웃해집단의 크기를 R=1, 3, 5, 7, n_i 로 변화시켜 각각의 문제에 적용하여 실험하였다. 여기서 n_i 는 기계 i에 배정한 작업의 수이므로 제한된 이웃해를 적용하지 않고 하나의 작업에 대한 총 이웃해를 발생하여 이웃해집단으로 구성함을 의미한다. 본 연구에서 알고리듬 정지여부를 판단하는 기준은 일정 반복횟수동안 해의 값이 개선되지 않는 횟수인 최대 반복횟수이며, 최대 반복횟수는 본 실험에서 수행한 문제의 해의 값이 m 보다 크게하여 알고리듬을 수행하여도 해가 개선되지 않고 계산시간만 증가하였으므로 기계대수 m 으로 하였다.

실험에 사용된 컴퓨터 시스템은 486-SX 이며 사용한 언어는 FORTRAN-77 이다. 본 연구의 알고리듬의 평가는 Ho 와 Chang 연구[7]보다 향상된 해의 값을 나타내기 위해 효율 =(Ho - TS)/H0 × 100 을 사용하였다.

표 1 은 여러 R의 값에 따른 해의 값과 계산시간을 보여준다.

표 1 에서 나타난 바와 같이 R=1 에 비하여 R=n_i 일 경우 최대 20 배, 최소 8 배, 평균 12 배의 계산시간이 더 소요됨을 알 수 있다. 그러나 해의 효율에서는 R=1 과 R= n_i 일 때의 차이는 최대 1.2%, 최소 0.3%, 평균 0.6% 로 차이가 적다. 따라서 본 연구의 제한된 이웃해집단의 구성방법으로 간단하게 R=1 로 하여 제안하는 TS 알고리듬을 사용하는 것이 해의 값과 계산시간면에서 바람직하다.

표 1. 여러 R의 값에 따른 해의 값과 계산시간

cpu=계산시간(초)

n	m	R=1		R=3		R=5		R=7		R=n_k	
		value	cpu	value	cpu	value	cpu	value	cpu	value	cpu
50	2	43.90	1.0	43.69	2.0	43.57	3.2	43.50	4.7	43.49	17.0
	3	40.98	1.1	40.72	2.7	40.65	5.0	40.58	6.8	40.58	16.0
	4	33.48	1.3	33.34	4.2	33.36	6.5	33.50	8.2	33.30	19.1
100	5	44.15	2.3	44.11	4.9	44.03	5.7	44.00	8.7	43.93	27.2
	6	30.95	13.6	30.63	39.0	30.32	59.5	30.12	62.6	30.07	181.3
	7	25.03	15.5	24.99	39.6	24.95	66.4	24.97	70.4	24.90	218.4

표 2는 Ho 와 Chang 알고리듬[7]과 본 연구의 TS 알고리듬을 비교항 것으로 평균 3.5% 해의 효율을 높힐 수 있다. 더욱이, 본 연구의 알고리듬은 실험에 사용된 모든 문제에 대해서 Ho 와 Chang 알고리듬[7] 보다 좋은해를 구하였다.

표 2. Ho 와 Chang 알고리듬[7] 과 제안된 TS 알고리듬의 비교

n	m	Ho와 Chang	제안된 TS	효율(%)
50	2	45.90	43.90	4.6
	3	42.70	40.98	4.0
	4	34.07	33.18	2.7
100	5	45.82	44.15	3.6
	6	32.13	30.95	4.1
	7	25.46	25.03	2.6

7. 결 론

평균 지연시간을 최소화하는 병렬기계 일정계획 문제는 최근 생산시스템에서 나타나고 있는 납기의 중요성과 생산성의 효율을 위하여 많이 사용되고 있다. 본 연구는 이 문제에 대해 우수한 해를 제공함과 동시에 범용성을 갖는 TS 알고리듬을 개발하였다. 제안한 TS 알고리듬에서는 이웃해를 생성하는 이동방법으로 다른 두 기계에 있는 두 작업들을 교환시키는 교환이동을 사용하였으며, 이때 두 기계의 작업순서가 MT를 최소화한다는 보장이 없으므로 각각의 기계에서 단일기계 일정계획을 적용해야 한다. 이를 위하여 본 연구는 각각의 기계에서 작업순서를 최적화하키 위한 방법으로 삽입방법을 사용하여 일반적인 TS 알고리듬의 특성인 범용적인 알고리듬이 되도록 하였다.

문제의 크기가 큰 현실 일정계획 문제에 TS 알고리듬을 적용할 때 생성가능한 이웃해에 따른 많은 계산시간이 필요하다. 계산시간을 줄이기 위하여 본 연구는 교환될 기계의 작업과 교환하려는 기계의 작업순서와 작업개수를 고려하여 이웃해를 만드는 이웃해집단을 구성하고 제안된 TS 알고리듬에 적용하여 실험하였다.

실험결과 R=1 에 비하여 R= n_i 일 경우 최대 20 배, 최소 8 배, 평균 12 배의 계산시간이 더 소요됨을 알 수 있었다. 그러나 해의 효율에서는 R=1 과 R= n_i 일 때의 차이는 최대 1.2%, 최소 0.3%, 평균 0.6% 로 차이가 적다. 따라서 본 연구의 제한된 이웃해집단의 구성방법으로 간단하게 R=1 로 하여 제안하는 TS 알고리듬을 사용하는 것이 해의 값과 계산시간면에서 바람직하다. 또한 해의 값은 Ho 와 Chang 알고리듬[7]을 비교해 볼때 평균 3.5% 해의 효율을 높힐 수 있었다. 더욱이, 본 연구의 알고리듬은 실험에 사용된 모든 문제에 대해서 Ho 와 Chang 알고리듬[7] 보다 좋은 해를 구하였다. 따라서, 본 연구에서 개발한 TS 알고리듬은 병렬기계 일정계획 문제에 있어서 평균 지연시간의 감소에 따른 비용절약과 생산성 향상에 도움을 줄 것이다.

참 고 문 헌

1. Adenso-Diaz, B. "Restricted Neighborhood in the Tabu Search for the Flowshop Problem," *European Journal of Operational Research*, Vol. 62, pp. 27-37, 1992.
2. Baker, K. R. and G. Merten, "Scheduling with Parallel Processors and Linear Delay Costs ," *Naval Research Logistics Quarterly*, Vol. 20, pp. 793 -804, 1973.

3. Barnes, J. W. and M. Laguna, "Solving the Multipul-Machine Weighted Flow Time Problem Using Tabu Search," *IIE Transactions,* Vol. 25, No. 2, pp. 121-127, 1993.
4. Barnes, J. W. and J. B. Chambers, "Solving the Job Shop Scheduling Problem with Tabu Search," *IIE Transactions,* Vol. 27, pp. 257-263, 1995.
5. Cheng, T. C. E. and C. C. S. Sin, "A State of the Art Review of Parallel Machine Scheduling Research," *European Journal of Operational Research,* Vol. 47, pp. 271-292, 1990.
6. Dell'Amico, M. and M. Trubian, "Applying Tabu Search to the Job-Shop Scheduling Problem," *Annals of Operations Research,* Vol. 41, pp. 231-252, 1992.
7. Ho, J. C. and Yih-L/ong Chang, "Heuristic for Minimizing Mean Tardness for m Parallel Machines,"*Naval Research Logistics Quarterly,* Vol. 38, pp. 367-381, 1991.
8. Ho, J. C. and Yih-Long Chang, "Minimizing the Number of Tardy Jobs for m Parallel Machines," *European Journal of Operational Research,* Vol. 84, pp. 343-355, 1995.
9. Laguna, M., J. Barnes, and F. Glover, "Tabu Search Methods for a Single Machine Scheduling Problem," *Journal of Intelligent Manufacturing,* Vol. 2, pp. 63-74, 1991.
10. Taillard, E., "Some Efficient Heuristic Methods for the Flowshop Sequencing Problem," *European Journal of Operational Research,* Vol. 47, pp. 65-74, 1990.
11. Wilkerson, J. and J. D. Irwin, "An Improved Algorithm for Scheduling Independent Tasks," *AIIE Transactions,* Vol. 3, pp. 239-245, 1971.