한국산학기술학회논문지
Vol. 5, No. 2, pp. 137-140, 2004

Simulated Annealing을 이용한 한 작업-다중 기계문제에서의 Makespan 최소화

이동주 · 황인극 · 김진호

Makespan Minimization Problem for A Job - Multiple Machines Using Simulated Annealing

Dong-Ju Lee, Inkeuk Hwang and Jin Ho Kim

요 약 다중 프로세서 시스템이 개발됨에 따라, 새로운 일정계획문제, 하나의 작업이 하나이상의 기계에 의해 동시에 처리되어야 하는 문제가 대두되었다. 이 연구에서는 선행관계를 가진 이러한 다중 프로세서 일정계획문제에 대해 다루어 보았다. 이 연구의 목적은 makespan을 최소화하는 일정계획을 찾는 것이다. 일반적으로 Branch and Bound 기법을 이용하여 선행관계를 가진 다중 프로세서 일정계획문제의 최적해를 찾았는데, 해의 탐색시간이 너무 오래 걸린다는 단점이 있었다. 본 연구에서는 짧은 시간 내에 최적해와 가까운 근사해를 simulated annealing(SA)방법을 이용하여 구해보았다. SA의 성능을 측정하기 위하여, SA의 CPU 처리시간과 구한 근사해를 40개의 예제문제를 통하여 Kramer의 방법의 CPU 처리시간과 최적해와 비교해 보았다.

Key Words : Scheduling; Multiprocessor tasks; Simulated Annealing

1. 서 론

고전적인 일정계획문제에서, 일반적인 가정 중 하나는 각 작업은 하나의 기계에 의해 처리되어야만 한다는 것이다. 그러나, 다중프로세서 시스템이 발달함에 따라, 이러한 가정은 사실과 멀어지게 되었다. 최근에 여러 학자들은 하나의 작업이 여러 대의 기계에 의해 동시에 처리되어야 하는 문제를 분석해오고 있다. 이러한 문제는 한 작업-다중 기계문제(one job on multiple-machine problem) 라고 불리운다.

예를 들면, 여러 대의 작업들과 여러 대의 기계가 있다고 할 때, 몇몇 작업들은 다른 작업들 이전에 처리되어야 하는데 이러한 관계를 선행관계라고 한다. 또한, 각 작업은 하나 이상의 기계들에 의해 동시에 처리되고, 각 작업들이 처리될 기계들은 이미 정해져 있다. 이 문제의 목적은 총 작업이 마치는 시간(makespan)을 최소화하는 것이다. Veltman et. al.[8]은 이 문제를 Plfixi;precICmax.라고 정의했다.

이러한 문제들은 여러 학자들에 의해 널리 연구되어 왔고, Blazewicz et al.[4]은 우선적인(preemptive) 작업이 있는 경우와 없는 경우(non-preemptive)에 대해 makespan을 최소화하는 문제를 연구했다. Goemans[7] and Blazewicz et al.[3]은 3 대의 전용기계들에 있어서 makespan을 최소화하는 문제를 연구했다 (P3lfixilCmax). Blazewicz et al[4]는 연쇄 제약식을 가지는 경우에 있어서 다중 프로세서 일정계획문제를 분석했다. Chen과 Lee [6]는 작업의 기계에 대한 할당과 일정계획을 모두 고려한 일반적인 다중 프로세서문제를 연구했다. 특히, 기계가 두 대일 때와 세 대일 때의 특수한 경우에는 최적해를 찾는 방법을 동적계획법을 이용해 개발했다. Blazewicz et al. [5]는 다중 프로세서 문제들에 대해 지금까지 해온 연구들을 조사했다.

최근에는 일정계획문제등에도 폭 넓게 사용되고 있다. 우[2]는 순열 flowshop에서 makespan을 최소화하는 문제를 SA를 이용해 풀어보았다. 또한, 김[1]은 TSP(Traveling salesman problem)를 엔트로피 개념을 이용해 개선한 SA를 적용하여 좋은 해를 구했다.

2. Simulated Annealing(SA) 알고리듬

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

J_i: 작업 i, $i = 1,..,n$

공주대학교 산업시스템공학과
ikhwang@kongju.ac.kr

M_j: 기계 j $j = 1,...,m$
C_i: 작업 i (J_i)의 완성시간 (completion time)
C_{max}: makespan = max $\{C_i : i = 1,...,n\}$
t_i: 작업 i (J_i)의 소요시간
D_i 작업 i (J_i)를 처리하기 위한 기계군
$preci$: 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. 작업들의 선후관계는 $J_4 \rightarrow J_5 \rightarrow J_6$ 이다. 즉, 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 [Table 1.]과 같다.

Table 1. 예제

Jobs	ti	Di	preci
J_1	1	{M3,M4,M5}	
J_2	3	{M1,M4}	
J_3	3	{M2,M3,M4}	
J_4	3	{M1,M5}	
J_5	1	{M2,M3}	{J4}
J_6	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_j동시에 처리될 수 있다면 이러한 관계는 평행관계(parallel relationship)로 정의하고 $J_i \| J_j$로 표시하기로 한다. 본 연구에서는 평행관계를 가진 작업들만 고려하여 인접해들을 탐색하도록 한다.

SA의 초기온도는 초기에 이루어지는 대부분의 이동이 수락되도록 충분히 높게 설정되어야 한다. 또한, 온도제어 패러미터를 탐색과정에 따라 감소시키는 방법을 쿨링스케쥴이라고 하는데 본 연구에서는 일반적으로 자주 쓰이는 다음 방법을 쓰기로 한다: $T_{k+1} = \alpha T_k$, 여기서 k는 현재의 반복횟수이고 T_k는 쿨링스케쥴에서의 현재 온도이다. 패러미터 α는 일반적으로 0.5~0.99값이 설정되는데, 크게 설정될수록 수렴속도는 늦어지지만 해의 질은 좋아지게 된다. 본 연구에서는 좀 더 좋은 해를 구하기 위해 0.999를 쓰기로 한다. 현재의 온도(T_k)가 최종온도(T_f)보다 낮은 경우나 반복횟수(k)가 미리 정해둔 최대 반복횟수(max_iteration)보다 큰 경우에는 종료한다. 이러한 패러미터들과 함께 이동(move)도 또한 SA의 효율에 많은 영향을 미친다.

기본적으로 각 작업은 작업의 번호 순서대로 하나씩 왼쪽에서 오른쪽으로 할당되는데, 현재 할당할 작업 앞에 최소한 하나이상의 평행관계를 가진 작업이 있다면 현재 할당할 작업을 선행관계를 가진 작업들과 함께 가장 가까이에 위치한 평행관계를 가진 작업직전에 할당한다. 즉, 작업순서에서 k번째에 있는 작업을 a_k라고 하고, 현재 k번째 작업을 할당하려고 한다고 하자. 또한, a_j가 a_k와 가장 가까이에 위치한 평행관계를 가진 작업이라고 하자. 여기서 $j < k$. j번째 작업과 k번째 작업 사이에 위치한 작업들 중 a_k와 선행관계를 가진 작업들 모두를 a_k와 함께 a_j직전에 할당한다. 그 다음에는 k+1번째 작업을 같은 방법으로 할당한다. 이런 식으로 모든 작업을 할당할 때까지 계속한다. 위의 여러가지를 고려한 본 연구에 적용된 SA의 앨고리듬은 다음과 같다.

단계 1. 초기온도 T_0와 최종온도 T_f를 정한다. 초기해 x_0와 초기해의 makespan ($C(x_0)$)을 구한다. 반복횟수 k = 0, $best = x_0$, $C_{best} = C(x_0)$으로 정한다.

단계 2. 현재의 작업순서에서 임의로 j번째 있는 작업을 선택한다. a_j를 j번째에 있는 작업이라고 하면, a_j와 평행관계를 가진 모든 작업들(p_j)을 찾는다. 이러한 p_j중 a_j를 p_j 직전에 삽입했을 때 최소의 makespan을 가지는 p_j를 선택하고 이러한 이동에 의해 얻어진 새로운 해를 y로 정한다.

단계 3. $k = k+1$로 정하고, $\Delta C = C(y)-C(x_k)$를 계산한다.

(1) $\Delta C \leq 0$이고 $C(y) < C_{best}$ 이면 $best = y$, $C_{best} = C(y)$ 으로 두고, 또한 $x_{k+1} = y$, $C(x_{k+1}) = C(y)$로 둔다.

(2) $\Delta C > 0$이고 $C(y) < C_{best}$ 이면 $x_{k+1} = y$, $C(x_{k+1}) = C(y)$으로 둔다.

(3) $\Delta C > 0$이면, 확률 $exp(-\Delta C/T_k)$으로 $x_{k+1} = y$, $C(x_{k+1}) = C(y)$로 둔다(uphill move). 확률 $1-exp(-\Delta C/T_k)$으로 $x_{k+1} = x_k$로 두고, $T_{k+1} = \alpha T_k$로 둔다.

단계 4. 만약 $k \leq$ max_iteration이고 $T_f < T_{k+1}$이면 단계 2로 돌아가고, 그렇지 않으면 종료한다.

3. 결 과

SA 기법의 결과는 Kramer의 branch and bound 기법의 결과와 비교되었다. SA는 'C 언어'로 구현되어 800MHz p/c로 실행되었고, Kramer의 방법은 역시 'C 언어'로 구현되어 Sun 4/20 Workstation에 의해 실행되었다. 이 두 방법의 성능을 비교하기 위해 rdata 와 vdata로 불리는 두 가지의 데이터 종류가 사용 되었는

데, 각 종류의 데이터는 10개의 예제들이 포함되어 있다. 각 예제들은 5개의 기계로 구성되어 있고, rdata에 대해서는 20개 혹은 30개의 작업들이, vdata에 대해서는 20개 혹은 40개의 작업들로 구성되어 있다.

* SA : Simulated Annealing에 의해 찾은 근사해
* Time2 (sec) : SA에 의해 근사해를 찾는데 걸린 컴퓨터CPU 시간(단위:초)
* Error (%) : SA에 의한 근사해와 최적해와의 평균 차이, 즉,

$$\frac{\text{SA} - \text{Optimum}}{\text{Optimum}} \times 100(\%)$$

20개의 작업들이 있는 예제들[Table 2]와 [Table 4]인 경우에는 거의 대부분 SA에 의해 최적해를 찾을 수 있었고, 작업수가 30개 혹은 40개인 경우에는 SA에 의한 해는 대부분의 경우 최적해와 차이가 있었다. 그렇지만, 이러한 차이, 즉, error율은 크지 않아서 최악인 경우 7.05%에 지나지 않았다. 특히, 40개의 예제들 중 세가지 예제들에서는 Kramer의 알고리듬이 5시간동안 최적해를 찾지 못하고 그 시간동안의 최선해가 괄호안에 기록되었는데, SA에 의해 제공된 해가 최선해보다 나았다.

선행관계는 각 5개의 작업들이 사슬관계로 연결되어 있다. rdata에서는 각 작업이 요구하는 평균 기계수는 2대이고 vdata에서는 2.5대이다. 계산결과는 Table. 2, 3, 4, 5에 주어져 있다.

Kramer의 branch and bound 앨고리듬은 몇몇 문제에 대해서는5시간 (CPU-time)동안에도 최적해를 구하

Table 2. 20개의 작업들을 가진 Rdata

Rdata with 20 jobs

Prob	Time1 (sec)	Optimum	Time2 (sec)	SA	Error (%)
Rla01	44.4	543	1	548	0.92
Rla02	292.1	612	1	612	0.00
Rla03	61.0	588	1	588	0.00
Rla04	35.6	553	1	553	0.00
Rla05	63.6	552	1	552	0.00
Rla06	817.5	570	1	570	0.00
Rla07	27.0	548	1	556	1.46
Rla08	67.0	510	1	510	0.00
Rla09	27.5	573	1	573	0.00
Rla10	16.8	656	1	656	0.00

Table 3. 30개의 작업들을 가진 Rdata

Rdata with 30 jobs

Prob.	Time1 (sec)	Optimum	Time2 (sec)	SA	Error (%)
rla01	991	795	2	825	3.77
rla02	815.5	772	2	805	4.27
rla03	18000	(916)	2	915	
rla04	5727.5	779	2	792	1.67
rla05	9961.5	684	2	701	2.49
rla06	2236	772	2	799	3.50
rla07	2017.5	733	2	749	2.18
rla08	3092.5	733	2	752	2.59
rla09	461	737	2	787	6.78
rla10	178.5	803	2	832	3.61

Table 4. 20개의 작업들을 가진 vdata

Vdata with 20 jobs

Prob.	Time1 (sec)	Optimum	Time2 (sec)	SA	Error (%)
vla01	218.9	753	1	753	0.00
vla02	18000	(813)	1	813	0.00
vla03	2628.	657	1	657	0.00
vla04	988	792	1	792	0.00
vla05	22.9	809	1	809	0.00
vla06	983.5	694	1	694	0.00
vla07	18000	(593)	1	593	0.00
vla08	18000	(734)	1	723	
vla9	70.4	965	1	965	0.00
vla10	18000	(574)	1	564	

Table 5. 40개의 작업들을 가진 vdata

Vdata with 40 jobs

Prob.	Time1 (sec)	Optimum	Time2 (sec)	SA	Error (%)
vla01	22.7	1651	2	1683	1.94
vla02	713.2	1461	2	1499	2.60
vla03	13.9	1365	2	1381	1.17
vla04	18000	(1157)	2	1189	2.77
vla05	268	1282	2	1282	0.00
vla06	76.5	1393	2	1403	0.72
vla07	18000	(1121)	2	1170	4.37
vla08	14865	1291	2	1339	3.72
vla09	884.2	1320	2	1413	7.05
vla10	10649	1448	2	1499	3.52

지 못하는 경우가 있었는데, 이러한 경우에는 5시간동안 찾은 해중 최상의 해를 괄호안에 기록하였다. 표 2,3,4,5의 각 열이 의미하는 바는 다음과 같다.

* Optimum : Kramer의 branch and bound 앨고리듬에 의해 찾은 최적해
* Time1 (sec) : 최적해를 찾는데 걸린 컴퓨터CPU 시간 (단위:초)

모든 경우에 있어 SA는 1,2초내에 근사해를 제공한 반면, Kramer의 기법은 최소 13.9초, 최대 5시간에 최적해를 제공했다. 그러므로, SA는 좋은 해를 짧은 시간내에 제공한다고 할 수가 있겠다.

SA에 영향을 미치는 많은 패러미터들이 있는데, 이 실험에 쓰인 요소들은 다음과 같다

* *maxiter* : 총 반복횟수. 작업 수가 20개일 경우 5,000번이고 30개 혹은 40개인 경우 10,000번.
* T_0 : 초기온도. 모든 예제에 대해 200.
* T_f : 최종온도 모든 예제에 대해 2.

총 평균 SA에 의한 해의 평균error율은 1.53%이다. 이러한 총 평균 error율과 각 작업 수별 데이터종류별 평균 error율은 [Table 6.] 에 주어져 있다.

Table 6. 데이터와 작업수별평균 Error 비율

평균 Error 비율(%)	
20개의 작업을 가진rdata	0.24
30개의 작업을 가진 rdata	3.09
20개의 작업을 가진 vdata	0
40개의 작업을 가진vdata	2.79
계	1.53

4. 결 론

Kramer의 branch and bound 앨고리듬은 최적해를 찾을 수 있지만, 너무 오랜 컴퓨터 계산시간이 걸린다는 단점이 있다. 그러므로, 본 연구에서는 Simulated Annealing(SA)을 이용하여 빠른 시간 내에 최적해와 별로 차이가 없는 근사해를 찾아 보았다. 결과에서 보는 대로 SA에 의해 구한 해는 많은 경우에 최적해를 찾을 수 있었고, 모든 경우에 있어 좋은 해를 짧은 컴퓨터 CPU시간에 제공해 주었다. 특히, 작업들의 수가 20개인 경우에는 대부분 최적해를 찾은 반면, 작업의 수가 증가할수록(30개 혹은 40개) 최적해와의 차이가 발생하게 되었다. 그러므로, 작업의 수가 많은 경우에는 좀 더 나은 해를 제공할 수 있는 새로운 기법의 연구가 필요하다.

참고문헌

[1] 김정자, 최규탁, 정진욱 2단계 엔트로피를 이용한 Simulated Annealing의 개선에 관한 연구, 대한설비관리학회지 제 4권, 제 4호, pp.159-165.

[2] 우훈식 1997. Simulated Annealing 을 이용한 순열 Flowshop에서의 Makespan 최소화, 한국경영과학회/대한산업공학회, '97 춘계공동학술대회. pp. 42-45.

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

[4] J. Blazewicz, M. Drabowski, and J. Weglarz 1986. Scheduling Multiprocessor Tasks to Minimize Schedule Length, *IEEE Transactions on Computers*, Vol. C-35, No. 5, pp. 389-392.

[5] J. Blazewicz, M. Drozdowski, and J. Weglarz 1994. Scheduling Multiprocessor Tasks-A Survey, *Microcomputer Applications*, Vol. 13, No. 2, pp. 89-97

[6] J. Chen & C.-Y. Lee 1999. General Multiprocessor Task Scheduling, *Naval Research Logistics* 46, pp. 57-74.

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

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