Journal of Korea Multimedia Society Vol. 9, No. 1, January 2006(pp. 119-126)

유전 알고리즘을 이용한 멀티프로세서 시스템에서의 태스크 스케쥴링 알고리즘

김 현 철†

요 약

멀티 프로세서 시스템에서 스케쥴링은 매우 중요한 부분이지만, 최적의 해를 구하는 것이 복잡하여 최근 다양한 휴리스틱 방법들에 의한 스케쥴링 알고리즘들이 제안되고 있다. 본 논문에서는 유전 알고리즘을 이용한 새로운 스케쥴링 알고리즘을 제시한다. 또한, 해를 구하는 과정에서 시뮬레이티드 어닐링 (simulated annealing)의 확률을 이용하여 유전 알고리즘의 성능을 개선시킨다. 제시된 알고리즘은 태스크들의 최종 수행 완료 시간 (makespan)을 최소화하는 것을 목표로 한다. 모의 실험을 통하여 제시된 알고리즘이 다른 알고리즘보다 최종 수행 완료 시간이 작음을 확인할 수 있었다.

Task Scheduling Algorithm in Multiprocessor System Using Genetic Algorithm

Hyunchul Kim†

ABSTRACT

The task scheduling in multiprocessor system is one of the key elements in the effective utilization of multiprocessor systems. The optimal assignment of tasks to multiprocessor is, in almost practical cases, an NP-hard problem. Consequently algorithms based on various modern heuristics have been proposed for practical reason. This paper proposes a new task scheduling algorithm using Genetic Algorithm which combines simulated annealing (GA+SA) in multiprocessor environment. In solution algorithms, the Genetic Algorithm (GA) and the simulated annealing (SA) are cooperatively used. In this method, the convergence of GA is improved by introducing the probability of SA as the criterion for acceptance of new trial solution. The objective of proposed scheduling algorithm is to minimize makespan. The effectiveness of the proposed algorithm is shown through simulation studies. In simulation studies, the result of proposed algorithm is better than that of any other algorithms.

Key words: Genetic Algorithm(유전 알고리즘), Multiprocessor Scheduling(멀티프로세서 스케쥴링), Simulated Annealing(시뮬레이티드 어닐링)

1. 서 론

최근 멀티 프로세서 시스템에서의 응용프로그램이 급속도로 발전되고 있음에 따라 많은 태스크들을 보다 빨리 처리하기 위해 스케쥴링에 대한 관심이 높아지고 있다. 멀티 프로세서 시스템에서의 태스크

※ 교신저자(Corresponding Author) : 김현철, 주소 : 경북 경주시 효현동 산 42-1 경주대학교 (780-712), 전화 : 054) 770-5274, FAX : 054)770-5274, E-mail : kimhc@kyongju.ac.kr

접수일 : 2005년 4월 1일, 완료일 : 2005년 8월 24일
† 정회원, 경주대학교 컴퓨터멀티미디어공학부
※ 본 연구는 2006년도 경주대학교 학술연구비 지원에 의하여 이루어 졌음

스케쥴링은 매우 복잡하기 때문에 다양한 휴리스틱(heuristic) 방법을 이용한 알고리즘들이 제시되고 있다[1].

최근, 유전 알고리즘 (genetic algorithm; GA)을 이용한 몇몇의 스케쥴링 알고리즘들이 제시되었다. 유전 알고리즘은 생물의 적자 생존 과정을 이용한 최적화 수법으로 전통 수리적인 방법으로 해결하기 어려운 많은 문제들에 적용되어 좋은 결과들을 보여왔다[2,3]. Mitra와 Ramanatha는 유전 알고리즘을 이용하여 태스크간의 실행순서와 마감시간을 가진 비선점 태스크들을 위한 스케쥴링 알고리즘을 제시하였다[4]. Lin과 Yang은 하이브리드 (Hybrid) 유전 알고리즘을 사용하여 실행 순서별로 정렬된 태스크들을 위한 스케쥴링 알고리즘을 제시하였다[5]. Monnier는 유전 알고리즘을 사용하여 실시간 태스크를 위한 스케쥴링 알고리즘을 제시하였다[6]. Oh & Wu는 유전 알고리즘을 사용하여 다목적을 가지는 스케쥴링 알고리즘을 제시하였다[7]. 그러나 이러한 알고리즘들은 매우 간단한 유전 알고리즘 (simple GA)을 그대로 사용하고 있으므로 결과 값들은 보다 더 나은 결과 값으로의 개선의 여지를 가지고 있다.

본 논문에서는 유전 알고리즘을 기본으로 하는 새로운 스케쥴링 알고리즘을 제시한다. 해를 구하는 과정에서 시뮬레이티드 어닐링 (simulated annealing)[8]의 확률을 이용하여 유전 알고리즘의 성능을 개선시킨다. 제시된 알고리즘은 태스크들의 최종 수행 완료 시간 (makespan)을 최소화하는 것을 목표로 한다.

본 논문의 구성은 다음과 같다.

2절에서는 유전 알고리즘을 간략하게 소개하고, 3절에서는 멀티 프로세서 스케쥴링 문제를 보다 자세히 설명하고, 멀티 프로세서 스케쥴링 문제를 수학 모델화한다. 4절에서는 유전 알고리즘과 시뮬레이티드 어닐링의 확률을 설명하고 제시된 알고리즘을 소개한다. 5절에서는 제안한 알고리즘의 성능을 측정하기 위해 수행한 모의 실험 결과를 보이고, 6절에서 결론을 맺는다.

2. 유전 알고리즘

유전 알고리즘은 생물의 적자 생존 과정을 이용한 최적화 수법이다[9]. 자연계에 있는 생물의 적자 생존 과정에 있어서, 어떤 세대(generation)를 형성하는 개체(individual)들의 집합, 즉 개체군(population) 중에서 환경에 대한 적합도(fitness)가 높은 개체가 높은 확률로 살아남을 수 있게 되며, 이때 교배(crossover) 및 변이(mutation)로서 다음 세대의 개체군, 즉 자식 세대를 형성하게 된다.

GA에서 개체의 수를 개체군의 크기(population size)라고 한다. 각각의 개체는 염색체(chromosome)를 가지고 있으며 염색체는 복수개의 유전자(gene)의 집합으로 구성된다. 유전자에 의해 결정되는 개체의 형질을 표현형(phenotype)이라고 하고 이에 대응되는 염색체의 구조를 유전형(genotype)이라 하며, 표현형을 유전형으로 바꾸는 것을 엔코딩(encoding) 그 역을 디코딩 (decoding)이라고 한다.

기본적인 단순 유전 알고리즘(Simple Genetic Algorithm)은 다음 알고리즘 1과 같다[9].

```
procedure: Simple GA
input: data set of problem, parameter set of GA
output: best solution
begin
   t ←0;
   initialize P(t) by encoding routine;
   fitness eval(P) by decoding routine;
   while (not termination condition) do
       crossover P(t) to yield C(t);
       mutation P(t) to yield C(t);
       fitness eval(P, C) by decoding routine;
       select P(t +1) from P(t) and C(t);
       t ← t +1;
   end
      output best solution;
end
```

알고리즘 1. 단순 유전 알고리즘

알고리즘 1에서 t 는 현 세대를 나타내는 변수이고, $P(t)$는 부모, $C(t)$는 자식 세대를 나타낸다.

3. 멀티 프로세서 스케쥴링 문제와 수학 모델

본 논문에서 제시하는 스케쥴링 알고리즘은 모든 태스크들의 최종 수행 완료 시간을 최소화하는 것을 목적으로 하고, 스케쥴링 문제를 보다 명확히 정의하기 위해 다음과 같은 몇 가지 사항들을 가정한다.

1. 모든 태스크들은 비선점이다.
2. 모든 프로세서는 어떤 한 시점에서는 오로지 한 태스크만을 수행할 수 있다.
3. 모든 태스크는 동시에 두 개 이상의 프로세서에서 수행될 수 없다.
4. 프로세서들의 성능은 모두 같다.
5. 프로세서간의 메시지 전송시간은 무시한다.

이러한 멀티 프로세서 스케줄링 문제는 다음과 같이 수학적인 모델로 나타내어 질 수 있다. 모델에서 모든 태스크들의 수행 시간은 미리 알려져 있음을 가정한다.

$$\min F(\boldsymbol{x}, \boldsymbol{t}^S) = \max_i \{(t_i^S + c_i)x_{im}\} \tag{1}$$

$$\text{s.t. } t_i^S \geq t_j^S + c_j, \quad e_{ij} \in E \tag{2}$$

$$\sum_{m=1}^{M} x_{im} = 1, \ \forall i \tag{3}$$

$$x_{im} \in \{0,1\}, \ \forall i, m \tag{4}$$

위 수식에서 사용된 기호들을 다음과 같이 정의한다.

- 인덱스
  - i, j : task index, i, j=1,2,⋯,N
  - m : processor index, m=1,2,⋯,M
- 파라메타
  - $G=(T, E)$: task graph
  - $T=\{t_1, t_2, \cdots, t_N\}$: a set of N tasks
  - $E=\{e_{ij}\}$, i, j=1,2,⋯,N, $i \neq j$: a set of directed edges among the tasks representing precedence relationship
  - t_i : the ith task, i=1,2,⋯, N
  - e_{ij} : precedence relationship from task t_i to task t_j
  - p_m : the mth processor, m=1,2,⋯,M
  - c_i : computation time of task t_i
  - pre*(t_i) : set of all predecessors of task t_i
  - suc*(t_i) : set of all successors of task t_i
  - pre(t_i) : set of immediate predecessors of task t_i
  - suc(t_i) : set of immediate successors of task t_i
- 결정변수
  - t_i^S : start time of task t_i

$$x_{im} = \begin{cases} 1, & \text{if processor } p_m \text{ is selected for task } \tau_i \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

수식 (1)은 제시하는 알고리즘의 목적함수이며 모든 태스크들의 최종 수행 완료 시간의 최소화하는 것을 의미한다. 수식 (2), (3), (4)는 제시된 수학모델의 제약 조건들이다.

4. 시뮬레이티드 어닐링과 결합된 유전 알고리즘을 이용한 스케줄링

본 논문에서 제시하는 멀티 프로세서 시스템에서의 태스크 스케줄링 알고리즘에 적용된 엔코딩과 디코딩을 포함한 유전 알고리즘과 시뮬레이티드 어닐링 방법을 이 절에서 설명한다.

4.1 엔코딩과 디코딩

염색체 (chromosome) V_k, k=1,2,⋯,*pop-size* 는 프로세서에 할당된 태스크들을 나타낸다. 여기에서 *pop-size*는 한 세대에서의 전체 염색체의 수를 의미한다. 염색체 V_k는 $u(\cdot)$와$v(\cdot)$ 두 부분으로 나뉘어진다. $u(\cdot)$ 는 태스크들의 수행순서를 나타내고, $v(\cdot)$는 태스크들의 프로세서 할당 정보를 나타낸다. 염색체 각 부분의 길이는 태스크의 총 수와 같다.

엔코딩 프로시쥬어는 다음 알고리즘 2와 같이 나타내어 질 수 있다.

```
procedure: Encoding for multiprocessor scheduling
    begin
        l←1, W←φ ;
        while (T ≠ φ)
            W ← WUarg{τ_i |pre*(τ_i) = φ ,∀i };
            T ← T- {τ_i}, i∈W;
            while (W ≠ φ)
            j ← random(W);
            u(l) ← j ;
            W ← W - {j };
            pre*(τ_j) ← pre*(τ_j) - {τ_j},∀i;
            m← random[1:M];
            v(l) ← m ;
            l ← l+1;
            end
        end
        output u(·), v(·) ;
end
```

알고리즘 2. 엔코딩 프로시쥬어

엔코딩 프로시쥬어에서 W는 선행 태스크들을 가지지 않는 태스크의 집합이다.

$u(\cdot)$는W에서 랜덤하게 태스크를 선택함으로써 구성된다. 선택된 태스크는W에서 삭제되고, 또한 선택된 이 태스크를 선행 태스크로 가지는 모든 태스크들의 선행 태스크집합에서 삭제된다. $v(\cdot)$는$u(\cdot)$보다 구성이 간단한다. $v(\cdot)$는 프로세서를 랜덤하게 선택함으로써 구성되어진다. 제시된 엔코딩 프로시쥬어는 항상 태스크들의 주어진 수행 순서를 만족하여 적법한 염색체를 생성한다.

생성된 염색체는 디코딩 단계에서 해독되어 스케쥴링 결과를 생성하다. 디코딩 단계에서는 최종 스케쥴링 결과를 생성할 뿐만 아니라, 태스크들의 수행 시작 시간 및 종료 시간을 계산하여 목적함수 $F(\boldsymbol{x}, t^S)$ 값을 계산한다.

4.2 평가함수와 선택

유전 알고리즘에서 염색체는 평가함수에 의해서 평가되어진다. 좋은 평가를 받은 염색체는 그렇지 못한 염색체에 비해 다음 세대에 선택될 보다 높은 기회를 가지게 된다. 본 논문에서의 평가함수는 2절에서 제시된 목적함수를 기본으로 하지만, 선택과정에서 사용하게 될 룰렛 휠 (roulette wheel)[10]을 위해 최소화의 형태에서 최대화의 형태로 바뀌게 된다. 평가함수는 다음과 같이 정의한다.

$$eval(V_k) = 1 / F(\boldsymbol{x}, t^S), \ \forall k \qquad (6)$$

식 (6)은 목적함수의 역수이다. 본 논문에서 제시하는 알고리즘의 목적은 목적함수 $F(\boldsymbol{x}, t^S)$의 값을 최소화하는 것이다. 또한 룰렛 휠 방법에는 평가함수의 값이 큰 염색체가 좋은 평가를 받으므로 본 논문의 평가함수는 목적함수의 역수를 취함으로 인해 최소화의 형태에서 최대화의 형태로 바쑤었다. 선택 과정에서는 앞서 언급했듯이 룰렛 휠 방법을 사용한다.

4.3 교배와 변이

교배 (crossover)를 위하여 본 논문에서는 한 점 교배방법 (one-cut point crossover)을 이용한다. 태스크들의 수행 순서가 교배에 의하여 바뀌면 스케쥴링의 조건으로 주어진 수행 순서를 위배하는 적합하지 않은 염색체가 생성될 위험이 있으므로 염색체의 두 부분 중 프로세서의 할당을 나타내는 $v(\cdot)$ 부분만을 교배하도록 한다. 그림 1은 사용된 교배 방법의 예를 보여준다.

그림 1에서 교배점은 랜덤하게 선택한다. 스케쥴링의 조건으로 주어진 수행 순서를 위배하지 않도록 하기 위해$u(\cdot)$부분은 교배하지 않고, $v(\cdot)$부분만을 교배한다. 자식 1의$v(\cdot)$부분은 부모1의 교배점 전의 부분과 부모2의 교배점 후의 부분을 결합하여 형성된다. 마찬가지로 자식 2의$v(\cdot)$부분은 부모2의 교배점 전의 부분과 부모1의 교배점 후의 부분을 결합하여 형성된다.

염색체의 변이를 위해서는 가장 광범위하게 사용되고 있는 한 점 변이 방법 (one-bit altering mutation)을 사용하였다[11,12].

4.4 시뮬레이티드 어닐링의 확률을 이용한 유전 알고리즘의 성능 개선

시뮬레이티드 어닐링(simulated annealing)[8,9]은 커다란 탐색 공간에서 주어진 함수의 전역 최적점 (global optimum)에 대한 훌륭한 근사치를 찾으려고 하는 전역 최적화 (global optimization) 문제에 대한 일반적인 확률적 휴리스틱 (probabilistic heuristic) 접근 방식으로 그 명칭과 정신은 야금학 (metallurgy)의 담금질 (annealing)에서 따온 것이다. 즉 결정체 (crystals) 의 크기를 크게 하고 결함을 작게 하려고

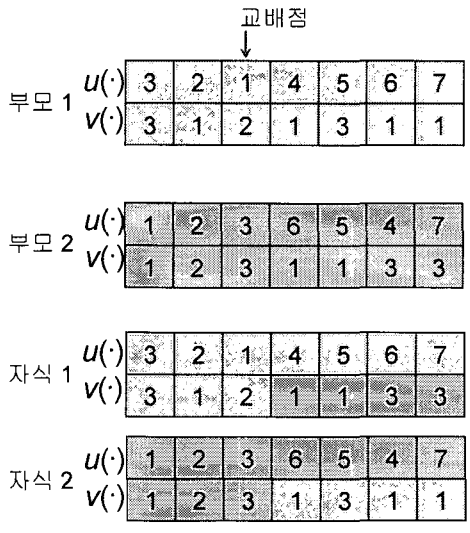

그림 1. 염색체의 교배

금속에 열을 가하고 냉각시키는 속도를 조절하는 (controlled cooling) 기술에서 따온 것이다.

유전 알고리즘과 시뮬레이티드 어닐링은 유사하지만, 몇 가지 중요한 차이점이 있다. 시뮬레이티드 어닐링은 반드시 최적해로 수렴하지만 유전 알고리즘은 그렇지 않다. 시뮬레이티드 어닐링은 목적 함수값에 의하여 생성된 후보 해를 확률적으로 선택하므로(Metropolis criterion), 열등한 해를 확률적으로 받아들이지만, 유전 알고리즘은 새로 생성된 자식의 해가 부모의 해보다 상당히 열등할지라도 모두 선택한다. 이러한 유전 알고리즘과 시뮬레이티드 어닐링이 결합함으로써 각각의 단점이 서로 보완될 수 있다. 즉, 유전 알고리즘에 시뮬레이티드 어닐링의 후보 해를 확률적으로 선택하는 과정을 도입함으로써 유전 알고리즘의 해의 수렴은 보다 안정적으로 이루어 질 수 있다.

시뮬레이티드 어닐링의 확률에 의한 유전 알고리즘의 개선에 대한 프로시쥬어는 다음 알고리즘 3와 같이 나타내어 질 수 있다.

```
procedure: Improving of GA by the probability of SA
  begin
      r ← random[0,1];
      ΔE ← eval(V')-eval(V);
      if (ΔE >0 || r <Exp(ΔE/Tem) )
              V'' ← V';
      else
              V'' ← V;
      Tem ← Tem × ρ;
      output offspring chromosomes V''
end
```

알고리즘 3. SA에 의한 GA의 성능 개선

제시된 프로시쥬어에서 V 와 V'는 각각 부모 염색체와 자식 염색체를 나타낸다. V''는 제시된 프로시쥬어에 의해 최종적으로 선택될 염색체를 의미한다. Tem와ρ는 시뮬레이티드 어닐링에서의 온도와 냉각률을 나타낸다.

프로시쥬어는 0에서 1까지에서의 난수를 발생하여 이 난수와 염색체의 개선된 정도를 비교하여 다음 세대에서 채택할 것인지를 결정하는 것을 나타낸다. 채택될 확률은 세대가 거듭할 수록 Tem의 값에 의하여 변하게 된다.

5. 모의 실험

본 논문에서 제시된 알고리즘의 성능을 검증하기 위해 모의 실험을 하였다. 모의 실험을 통해 본 논문에서 제시된 알고리즘 (GA+SA)을 GA를 사용하지 않은 휴리스틱 방법인 List 알고리즘[13,14]과 Monnier가 제시한 알고리즘 (Monnier's GA)과 성능을 비교하였다. 또한 시뮬레이티드 어닐링과 결합하지 않은 단순 유전 알고리즘 (simple GA)과 시뮬레이티드 어닐링 (SA)과도 성능을 비교하였다. List 알고리즘은 현재까지 사용되어 온 가장 널리 알려진 휴리스틱 방법이나 최근 제시되고 있는 GA를 이용한 스케쥴링 알고리즘보다 더 나은 결과를 보이지 못하고 있으며, 해를 구하는 시간에서도 태스크의 수가 늘어날수록GA보다 많은 시간이 걸린다. Monnier's GA는 단순한 유전 알고리즘에 기본을 두고 있으며, 선택에 있어서 선형 정규화 방법을 사용하고 있다. 선형 정규화 방법은 염색체들이 유사한 경우에 효과적으로 이용될 수 있으나 이러한 경우는 매우 제한적이다. 시뮬레이티드 어닐링은 항상 최적의 해를 구하는 알고리즘으로 알려져있으나 해를 구하는 과정에서 많은 시간을 요구한다.

모의 실험에 사용된 태스크들은 랜덤하게 발생하였다. 태스크들의 수행 순서를 정하기 위하여 P-Method[15]를 사용하였다. P-Method는 인접행렬의 확률적 구성을 기본으로 한다. 태스크t_i에서 태스크t_j로의 수행 순서관계가 존재할 경우 행렬의 요소 a_{ij}는 1이 된다. 그렇지 않고 독립적일 경우 행렬의 요소 a_{ij}는 0이 된다. 행렬의 하삼각 요소들의 값은 항상 0이 된다. 행렬의 상삼각 요소들은 파라메타ε를 가지는 Bernoulli의 성공 확률 법칙을 바탕으로 하여 1과 0중 랜덤한 값을 가지게 된다. 파라메타ε가 0일 경우 모든 태스크들은 수행 선행관계를 가지지 않고, 파라메타ε가 1일 경우는 모든 태스크들이 그 수행 선행관계에 의해 서로 연결되어 있다.

태스크들의 수행 시간은 평균값 5의 지수분포에 의해 랜덤하게 발생시켰다.

모의 실험은 총 태스크의 수 10, 50, 100의 세 가지 경우에 의해 실시되었고, 프로세서의 수는 2, 3의 두 가지의 경우로 실험하였다. 또한 태스크 수를 랜덤하게 발생시켜 2개의 프로세서에서 최종 수행 완료 시간을 비교한다.

5.1 태스크의 총 수 10의 경우

모의 실험에 사용된 태스크들의 수행 선행관계는 그림 2와 같다.

그림 2에서의 태스크들의 수행시간은 다음의 표 1과 같다.

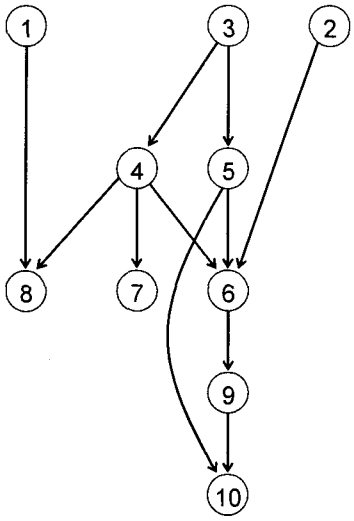

그림 2. 태스크 총 수 10의 수행 선행관계

표 1. 태스크 총 수 10의 경우 태스크들의 수행 시간

i	suc(τ_i)	c_i	i	suc(τ_i)	c_i
1	8	3	6	9	4
2	6	5	7	-	5
3	4,5	7	8	-	4
4	6,7,8	9	9	10	8
5	6,10	10	10	-	11

표 2는 실험의 결과를 보여준다. 표 2에서 본 논문에서 제시하는 스케쥴링 알고리즘 GA+SA가 다른 두 개의 알고리즘보다 최종 수행 완료시간이 현저히 작음을 알 수 있다. 비록 스케쥴링을 결과를 생산하는 계산시간이 조금 길긴 하지만 약간의 계산시간이 길어지는 것은 충분히 받아들여 질 수 있다.

5.2 태스크의 총 수 50의 경우

모의 실험에 사용된 태스크들의 수행 선행관계를 나타내는 그래프 그림과 수행시간을 나타내는 표는 생략하도록 한다.

표 3은 모의 실험 결과를 나타낸다. 표 3에서 또한 본 논문에서 제시하는 스케쥴링 알고리즘 GA+SA가 다른 두 개의 알고리즘보다 최종 수행 완료시간이 작다.

표 2. 태스크 총 수 10의 모의 실험 결과

프로세서의 수	비교 항목	List algorithm	Monnier's GA	Simple GA	SA	GA+SA
2	최종 수행 완료 시간	27	36	31	26	26
	계산시간 (*msec*)	23	21	21	32	25
'3	최종 수행 완료 시간	23	27	25	22	22
	계산시간 (*msec*)	25	22	23	33	26

표 3. 태스크 총 수 50의 모의 실험 결과

프로세서의 수	비교 항목	List algorithm	Monnier's GA	Simple GA	SA	GA+SA
2	최종 수행 완료 시간	128	134	131	122	123
	계산시간 (*msec*)	99	88	89	135	95
3	최종 수행 완료 시간	101	116	107	91	93
	계산시간 (*msec*)	102	89	91	136	97

표 4. 태스크 총 수 100의 모의 실험 결과

프로세서의 수	비교 항목	List algorithm	Monnier's GA	Simple GA	SA	GA+SA
2	최종 수행 완료 시간	345	369	352	337	339
	계산시간 (*msec*)	231	207	208	256	220
3	최종 수행 완료 시간	289	301	292	273	273
	계산시간 (*msec*)	239	208	208	257	221

5.3 태스크의 총 수 100의 경우

모의 실험에 사용된 태스크들의 수행 선행관계를 나타내는 그래프 그림과 수행시간을 나타내는 표는 생략하도록 한다.

표 4은 모의 실험 결과를 나타낸다. 표 4에서 또한 본 논문에서 제시하는 스케쥴링 알고리즘 GA+SA가 다른 두 개의 알고리즘보다 최종 수행 완료시간이 작다.

5.4 랜덤하게 발생된 태스크의 수

이 절에서는 태스크의 수를 랜덤하게 발생하여 2개의 프로세서에서GA를 이용한 스케쥴링 알고리즘들의 최종 수행 완료 시간을 비교한다. 태스크들의 수행 선행관계를 나타내는 그래프 그림과 수행시간을 나타내는 표는 생략한다.

그림 3은 모의 실험 결과를 나타낸다. 그림 3에서 또한 본 논문에서 제시하는 스케쥴링 알고리즘 GA+SA가 다른 두 개의 알고리즘보다 최종 수행 완료시간이 작음을 알 수 있다.

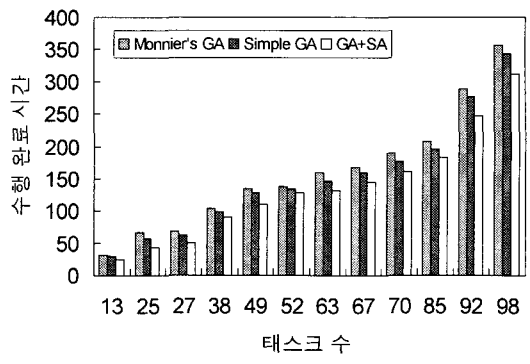

그림 3. 태스크 수 변화에 대한 수행 완료시간의 비교

6. 결 론

멀티 프로세서 시스템에서의 태스크 스케쥴링은 매우 복잡하기 때문에 다양한 휴리스틱 방법을 이용한 알고리즘들이 제시되고 있다. 본 논문에서는 유전 알고리즘을 이용한 멀티 프로세서 스케쥴링 알고리즘을 제시하였다. 또한 해를 구하는 과정에서 시뮬레이티드 어닐링의 확률을 이용하여 유전 알고리즘의 성능을 보다 개선시켰다. 제시된 스케쥴링 알고리즘은 태스크들의 최종 수행 완료 시간 (makespan)을 최소화하는 것을 목표로 한다.

제시된 알고리즘은 모의 실험을 통하여 유전 알고리즘을 이용한 다른 스케쥴링 알고리즘과 성능을 비교하였다. 모의 실험 결과에서 다른 스케쥴링 알고리즘보다 제시된 스케쥴링 알고리즘의 최종 수행완료 시간이 현저히 작음을 확인할 수 있었다.

제시된 스케쥴링 알고리즘은 기타 유사한 환경에서의 스케쥴링 알고리즘, 예를 들어, 프로세서 간의 메시지 전송시간을 고려한 환경에서의 스케쥴링 알고리즘, 다목적을 가지는 스케쥴링 알고리즘 등으로 확장될 수 있을 것이다.

참 고 문 헌

[1] Yalaoui, F., and C. Chu, "Parallel Machine Scheduling to Minimize Total Tardiness," *International Journal of Production Economics*, Vol. 76, No. 3, pp. 265-279, 2002.

[2] Bernat G., A. Burns, and A. Liamosi, "Weakly

Hard Real-Time Systems," *IEEE Transactions on Computer Systems*, Vol. 50, No. 4, pp. 308-321, 2001.

[3] Diaz J. L., D. F. Garcia, and J. M. Lopez, "Minimum and Maximum Utilization Bounds for Multiprocessor Rate Monotonic Scheduling," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 15, No. 7, pp. 642-653, 2004.

[4] Mitra, H. and P. Ramanathan, "A Genetic Approach for Scheduling Non-preemptive Tasks with Precedence and Deadline Constraints," *Proc. of the 26th Hawaii International Conference on System Sciences*, pp. 556-564, 1993.

[5] Lin, M. and L. Yang, "Hybrid Genetic Algorithms for Scheduling Partially Ordered Tasks in A Multi-processor Environment," *Proc. of the 6th International Conference on Real-Time Computer Systems and Applications*, pp. 382-387, 1999.

[6] Monnier, Y., J. P. Beauvais, and A. M. Deplanche, "A Genetic Algorithm for Scheduling Tasks in a Real-Time Distributed System," *Proc. of 24th Euromicro Conference*, pp. 708-714, 1998.

[7] Oh, J. and C. Wu, "Genetic-algorithm-based Real-time Task Scheduling with Multiple Goals," *Journal of Systems and Software*, Vol. 71, No. 3, pp. 245-258, 2004.

[8] Kirkpatrick, S., C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, No. 4598, pp. 671-680, 1983.

[9] Kim, H. C, Y. Hayashi, and K. Nara, "An Algorithm for Thermal Unit Maintenance Scheduling through combined use of GA, SA and TS," *IEEE Transactions on Power Systems*, Vol. 12, No. 1, pp. 329-335, 1997.

[10] Hou E. S. H., R. Hong, and N, Ansari, "Efficient multiprocessor scheduling based on genetic algorithms," *IEEE Conference on Industrial Electronics Society*, Vol. 2, pp. 1239-1243, 1990.

[11] Jackson L. E. and G. N. Rouskas, "Optimal Quantization of Periodic Task Requests on Multiple Identical Processors," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 14, No. 8, pp. 795-806, 2003.

[12] Deb, K., *Multi-objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[13] Adam, T., K. Chandy, and J. Dickson, "A comparison of list schedules for parallel processing system," *ACM*, Vol. 7, No. 12, pp. 685-690, 1974.

[14] Hwang, J., Y. Chow, F. Anger, and C. Lee, "Scheduling precedence graphs in systems with interprocessor communication times," *SIAM J. Computing*, Vol. 8, No. 2, pp. 244-257, 1989.

[15] Al-Sharaeh, S. and B. E. Wells, "A Comparison of Heuristics for List Schedules using The Box-method and P-method for Random Digraph Generation," *Proc. of the 28th Southeastern Symposium on System Theory*, pp. 467-471, 1996.

김 현 철

1984 숭실대학교 전자계산학과(학사)

1985~1991 국민건강보험공단 전산실

1993 숭실대학교 정보과학대학원(석사)

1996 일본 이바라키대학교 대학원 시스템공학과(박사)

2000~현재 경주대학교 컴퓨터 멀티미디어학부 조교수

관심분야 : 시스템공학, 모던휴리스틱의 NP-hard문제의 응용