韓國經營科學會誌
第21卷 第1號
1996年 4月

Genetic Algorithm을 이용한 다중 프로세서 일정계획문제의 효율적 해법*

朴承憲** · 吳勇周**

An Efficient Method for Multiprocessor Scheduling Problem Using Genetic Algorithm*

Seung-Hun Park** · Yong-Ju Oh**

Abstract

Generally the Multiprocessor Scheduling(MPS) problem is difficult to solve because of the precedence of the tasks, and it takes a lot of time to obtain its optimal solution. Though Genetic Algorithm(GA) does not guarantee the optimal solution, it is practical and effective to solve the MPS problem in a reasonable time.

The algorithm developed in this research consists of a improved GA and CP/MISF(Critical Path/Most Immediate Successors First). An efficient genetic operator is derived to make GA more efficient. It runs parallel CP/MISF with GA to complement the faults of GA. The solution by the developed algorithm is compared with that of CP/MISF, and the better is taken as a final solution.

As a result of comparative analysis by using numerical examples, although this algorithm does not guarantee the optimal solution, it can obtain an approximate solution that is much closer to the optimal solution than the existing GA's.

1. 서 론

다중 프로세서(Multiprocessor) 시스템은 동일한 능력을 가진 여러 대의 프로세서(processor)로 이루어진 시스템으로 처리효율의 이점 때문에 정보처리 시스템뿐만 아니라 로봇트 제어와 동적 시스템의 고속 시뮬레이션 등 여러

* 본 연구는 인하대학교 1995년도 연구비 지원에 의해 수행되었음
** 인하대학교 산업공학과

등 여러 분야에서 사용되고 있다. 이것은 병렬처리의 효율면에서 최대완료시간을 최소화하고 경제성을 고려하여 가능한 한 프로세서의 수를 적게 하는 것이 바람직하다. 이러한 시스템에서 과업(task)들의 처리시간이 다르고 선점(preemption)이 허락되지 않으며 과업들간에 선행관계가 존재한다면 최적 일정계획(scheduling)은 매우 어렵다. 이러한 문제는 멀티 프로세서 스케쥴링(Multiprocessor Scheduling : MPS)으로 정식화할 수 있다.

예로서 MPS문제는 컴퓨터 시스템에 있어서 복수의 프로세서로 복수의 과업을 처리하여 최대완료시간을 최소화하는 문제이다. 이러한 문제의 최적 일정계획은 복수의 프로세서 상에서 과업들간의 선행관계를 유지하면서 가능한 한 모든 과업을 최단시간에 완료시키는 것이다. 지금까지는 CPM(Critical Path Method)과 HLFET(Highest Levels First with Estimated Times) 등의 해법을 사용하여 왔으나 최근에는 Genetic Algorithm(GA : 유전해법)을 이용한 해법이 제안되었다.

특히 문제의 규모가 커지는 경우 최적해를 구하기 위해서는 계산량이 대폭 증가하여 많은 시간이 소요되므로 실용적이지 못하다. GA는 대규모 문제에 있어서 효율적이며 실용적인 근사해법으로 알려져 있다.

GA는 자연계의 진화과정을 지배하는 적자생존의 원리 및 유전정보의 교환에 의한 세대교체의 원리를 문제해결을 위한 해법절차로서 모사하려는 것이다. GA에서는 단일해가 아닌 해집단을 다루며, 해집단의 반복적인 세대교체 과정을 통해 최적해를 탐색한다.

본 연구는 MPS 문제를 대상으로 한 기존의 Gen[7], Hou[9] 등의 GA방법에 효율적인 유전적 조작을 도입한 GA를 이용하여 문제를 해결한다. GA는 대체로 우수한 해를 구하지만 탐색절차에 따라서는 그렇지 못한 경우도 발생한다. 이러한 GA의 결점을 보완하기 위해 비교적 우수한 단일해를 구하는 발견적 알고리듬인 CP/MISF(Critical Path/Most Immediate Successors First)방법[11]을 병행한다. 그리고, 두 가지 방법을 통해 얻은 해를 비교하여 더 우수한 해를 최종해로 제시하는 알고리듬을 개발한다. 제시된 알고리듬은 수치예를 통하여 그 유효성을 검토한다.

2. 정식화 및 유전적 조작

여기에서는 GA를 이용하여 MPS문제를 정식화하기 위한 개념들을 정의하고 유전적 조작(Genetic Operation)방법들의 의미와 역할을 살펴본다. 또한 CP/MISF방법의 개념 및 절차를 기술한다.

2. 1 고도함수와 일정계획의 표현

일정계획은 [그림 1]의 과업도표(task graph)와 같이 과업들간에 선행관계를 만족해야 하며 각 과업을 단 한번만 포함해야 한다. 각 과업간의 선행관계를 고도함수(height' function)로 표현하며 이것에 의해 실행가능한 일정계획을 표현한다.

고도함수는 각 과업의 선행관계를 표현함으로써 GA를 적용해 MPS문제를 해결하는 데 핵심적인 역할을 수행한다. 즉, 고도함수값이 작은 과업을 고도함수값이 큰 과업보다 선행하도록 일정계획함으로써 과업간의 선행관계가 유지될 수 있다. [그림 1]의 과업 도표의 각각

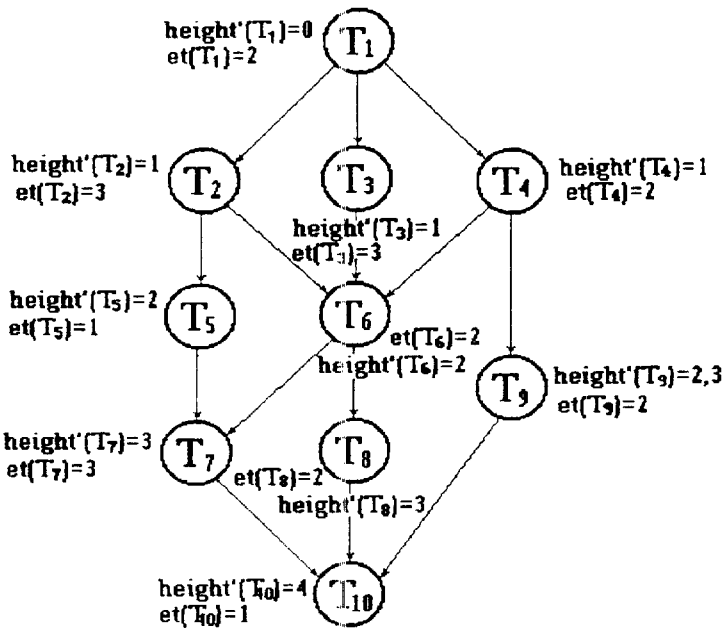
의 과업에 대해 식 (1)과 같은 고도함수 $height'(T_j)$를 정의한다.

$$height'(T_j) = \text{rand} \in [\max_{\text{over all } T_i \in PRED(T_j) \text{ and } T_k \in SUCC(T_j)} \{height'(T_i)\} + 1,\ \min\{height'(T_j)\} - 1] \quad \cdots\cdots(1)$$

$$height(T_j) = \begin{cases} 0, & PRED(T_j) = \varphi \quad \cdots\cdots(2) \\ 1 + \max_{T_i \in PRED(T_j)} \{height(T_i)\}, & otherwise \end{cases}$$

T_i : i번의 과업(i = 1, 2, ···, n)
$PRED(T_j)$: T_i의 선행공정 집합
$SUCC(T_j)$: T_j의 후속공정 집합
$rand$: 整數의 亂數

식 (2)의 $height(T_j)$는 각 과업 선행공정의 집합 ($PRED(T_j)$)에 속한 과업들의 $height(T_i)$ 값 중에서 가장 큰 값에 1을 더한 값을 취한다. 그러나 식 (2)의 $height(T_j)$에 의해 고도함수값이 고정되면 GA의 해공간 탐색이 제한되는 단점이 있다. 예로서 $height(T_j)$에 의해 [그림 1]의 9번 과업의 고도함수값을 2로 고정하면 실행가능한 일정계획임에도 불구하고 고도함수값이 3인 경우를 배제하게 되어 GA의 해공간 탐색이 제한된다. 따라서 과업간의 선행관계가 유지되는 범위내에서 GA의 해공간 확대를 위해 식 (1)의 $height'(T_j)$를 도입할 필요가 있다.



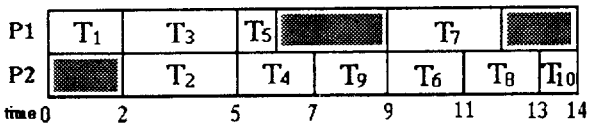
[그림 1] 선행 관계를 갖는 10개 과업의 과업 도표

일정계획을 표현하려면 n(처리할 과업의 수)개의 과업을 m(프로세서수)개의 프로세서에 할당하여야 하며 그 방법은 다음과 같다.

모든 과업들은 [1, m]의 구간내에서 발생된 난수값에 따라 각각의 프로세서에 할당된다. 이때 각 과업은 고도함수값에 따라 오름차순으로 정렬되도록 자신의 고도함수값보다 큰 값을 가지는 과업들 중 가장 작은 고도함수값을 가지는 과업앞에 삽입된다. 이 과정을 통해 [그림 1]의 예를 두 대의 프로세서 P_1, P_2에 할당한 결과는 다음과 같다.

P_1 : $height'(T_1) \leq height'(T_3)$
$\leq height'(T_5) \leq height'(T_7)$
P_2 : $height'(T_2) \leq height'(T_4)$
$\leq height'(T_9) \leq height'(T_6)$
$\leq height'(T_8) \leq height'(T_{10})$

이것을 간트 차트로 표현하면 [그림 2]와 같다.



[그림 2] 간트 차트

위의 간트 차트를 일정계획으로 표현하면 다음과 같다. 이후 실행가능한 일정계획을 GA의 특성상 개체 또는 개체 S라 칭한다.

$$S = \begin{cases} P_1\ (T_1\ T_3\ T_5\ T_7) \\ P_2\ (T_2\ T_4\ T_9\ T_6\ T_8\ T_{10}) \end{cases}$$

2. 2 초기 해집단과 평가함수

초기 해집단은 임의로 부여하는 해집단 크기(population size : popsize)만큼의 개체들로 구성된다. 이 개체들은 2. 1에서 전술한 과정을 해집단 크기만큼 반복함으로써 생성된다.

평가함수는 각 개체의 우열을 판단하는 척도이다. MPS문제의 평가함수는 모든 과업이 완료되는 시점으로 정의할 수 있다. 즉, 평가함수는 개체의 최대완료시간이고 최대완료시간은 각각의 프로세서의 완료시간 중 가장 큰 값이다. 이를 표현하면 식(3)과 같다.

$$f(S) = \max_{1 \leq k \leq m} \{t_k\ (S)\} \quad \cdots\cdots (3)$$

$f(S)$: 개체 S의 평가함수
$t_k(S)$: 개체 S 중 프로세서 k의 완료 시간
m : 프로세서수

한 세대내에서 해집단의 개체 중 최소평가함수값을 갖는 개체를 S^*_{gen}라 하고 다음과 같이 정의한다.

$$S^*_{gen} = \underset{S_i}{argmin}\ \{f(S_i)\} \quad \cdots\cdots (4)$$

S_i : i번째 개체 ($i = 1,2,\cdots,popsize.$)
gen : 세대수 (gen = 1,2,3$\cdots$,$maxgen.$)
$maxgen$: 최대세대수 (Maximum generation, GA과정을 반복할 세대교체의 횟수)

여기서 $\underset{S_i}{argmin}$는 최소평가함수값 $f(S_i)$의 S_i를 채택함을 의미한다.

한편 세대교체가 진행된 세대까지 발견된 S^*_{gen}들 중 가장 우수한 개체를 S^{**}라 하고 다음과

같이 정의한다.

$$S^{**} = \underset{S^{*}_{gen}}{argmin}\ \{f(S^{*}_{gen})\} \quad \cdots\cdots(5)$$

gen : 세대수 (gen = 1,2,…,현재 세대수)

세대교체가 거듭되는 동안에 만약 S^{*}가 S^{**}보다 최대완료시간이 작다면 S^{**}를 S^{*}로 교체한다. S^{**}는 유전적 조작방법 중 복제에서 사용되며 GA가 종료된 후에는 GA의 최종해로 제시된다.

2. 3 유전적 조작

유전적 조작은 현재 해집단의 개체들로부터 유전정보를 교환하여 다음 세대의 새로운 개체를 생성하는 것이다. 즉, 자손해는 부모해를 재배열하거나 결합함으로써 만들어 진다.

다음 세대를 창조하기 위한 부모해의 선택과정에서는 부모해가 지닌 고유한 특성이 자손에게 전달될 수 있도록 한다. 부모의 복합적인 특성을 물려받은 자손해는 열등한 부모해를 대체하면서 다음 해집단을 재구성한다. 이러한 유전적 조작을 거치는 과정에서 치사유전자(실행 불가능한 일정계획)가 발생하지 않도록 고려하여야 한다.

일반적으로 GA의 유전 조작은 두 개체간의 유전정보의 일부분을 교환하고 재배열하여 이루어지지만 본 연구의 유전 조작은 한 개체 내에서 임의(random)로 선택된 두 프로세서에 할당된 과업들의 일부분을 교환함으로써 이루어진다. 여기에서는 교배(crossover), 돌연변이(mutation), 고도함수 변환(height function transformation), 복제(reproduction) 등 4가지 유전적 조작방법(genetic operators)을 수행하며 그 의미와 역할은 다음과 같다.

(1) 교배(crossover)

교배는 두 부모해의 유전정보를 임의의 위치에서 부분적으로 교환함으로써 새로운 자손해를 생성하도록 하는 조작방법이다. 이것은 현재까지 탐색되지 않은 새로운 해공간을 탐색하는 것을 의미한다.

교배는 GA에서 중요한 역할을 수행하지만 모든 개체들을 항상 교배시키는 것은 바람직하지 않다. 왜냐하면 본질적으로 교배는 난수에 의해 지정된 위치의 과업의 후속부분을 교환하므로 최적해를 제거할 수도 있기 때문이다. 따라서 각 개체마다 난수를 부여하고 p_c(교배확률)보다 작은 경우에만 교배를 실시한다. 교배방법은 다음과 같다.

프로세서수가 2보다 클 경우에는 난수에 의해 개체를 선발한 다음 각각의 프로세서에 대해 [1, m]의 구간내에서 정수난수를 중복되지 않도록 발생시킨다. 이러한 정수난수에 의해 프로세서들간에 문자열 교환이 이루어진다. 즉, 각각의 프로세서의 문자열 중 임의로 선택된 고도함수값을 갖는 과업 뒷부분은 정수난수에 의해 지정된 프로세서의 같은 고도함수값을 갖는 과업 뒷부분으로 대체된다.

프로세서수가 2일 때에는 난수에 의해 선발된 임의의 개체에 대하여 두 프로세서의 문자열 중 임의로 선택된 고도함수값을 갖는 과업 뒷부분(윗줄이 있는 이탤릭체 부분)을 서로 교환한다. [그림 2]의 일정계획을 예로 하면 고도함수값으로 1이 선택되었다고 가정할 때 개체 S는 개체 S'로 변환된다.

$$S = \begin{cases} P_1\ (T_1\ T_3\ \overline{T}_5\ \overline{T}_7) \\ P_2\ (T_2\ T_4\ \overline{T}_9\ \overline{T}_6\ \overline{T}_8\ \overline{T}_{10}) \end{cases}$$

$$S' = \begin{cases} P_1\ (T_1\ T_3\ \overline{T_9}\ \overline{T_6}\ \overline{T_8}\ \overline{T_{10}}) \\ P_2\ (T_2\ T_4\ \overline{T_5}\ \overline{T_7}) \end{cases}$$

(2) 돌연변이(mutation)

돌연변이는 부모해로부터 자손해에게로 전달되는 특정한 유전정보에 대하여 무작위적인 변형을 시도함으로서 전체 해집단에서 배제된 새로운 개체, 혹은 진화과정에서 상실된 특정 유전정보의 부분적인 재현을 시도하는 조작방법이다.

돌연변이 역시 교배와 같은 이유로 항상 돌연변이를 수행하는 것은 바람직하지 않다. 따라서 부여된 난수가 p_b(돌연변이확률)보다 작은 경우에만 돌연변이를 실시한다. 돌연변이 방법은 다음과 같다.

난수에 의해 임의로 개체를 선발한 다음 각각의 프로세서에 대해 [1, m]의 구간내에서 정수난수를 발생시킨다. 각각의 프로세서내의 문자열 중에서 임의로 선택된 고도함수값을 갖는 과업들을 정수난수에 해당하는 프로세서의 문자열 내에 같은 고도함수값을 갖는 과업 앞에 삽입한다. 다음 예에서는 고도함수값이 2인 과업들(윗줄이 있는 이탤릭체부분)을 서로 교환한다.

$$S = \begin{cases} P_1\ (T_1\ T_3\ \overline{T_9}\ \overline{T_6}\ T_8\ T_{10}) \\ P_2\ (T_2\ T_4\ \overline{T_5}\ T_7) \end{cases}$$

$$S' = \begin{cases} P_1\ (T_1\ T_3\ \overline{T_5}\ T_8\ T_{10}) \\ P_2\ (T_2\ T_4\ \overline{T_9}\ \overline{T_6}\ T_7) \end{cases}$$

(3) 고도함수 변환(height function transformation)

고도함수를 변환하는 이유는 초기 해집단을 생성할 때에 고도함수를 고정시키면 탐색할 해공간이 제한되기 때문이다. 따라서 선행관계를 유지하는 범위 내에서 해공간의 확대를 위해 과업의 고도함수값을 변환할 필요가 있다. 즉, 과업간의 선행관계가 유지되는 범위내에서 고도함수값을 변환하여 해공간을 확대시킴으로써 최적해에 접근할 가능성을 크게 한다. 고도함수 변환방법은 다음과 같다.

고도함수 변환시에는 각 개체에 대해 난수를 발생시킨 다음 p_c (고도함수 변환시 개체선택확률)보다 작은 난수를 가진 개체를 선택하고 그 개체내에서 다시 고도함수 변환이 가능한 과업들에 대해 난수를 발생시킨 후 p_d(고도함수 변환시 과업선택 확률)보다 작은 난수를 가진 과업의 고도함수값을 과업간의 선행관계가 유지되는 범위내에서 임의로 변환한다. 다음 예에서는 T_9 의 고도함수값이 2에서 3으로 변환되었다.

$$S = \begin{cases} P_1\ (T_1\ T_3\ T_5\ T_8\ T_{10}) \\ P_2\ (T_2\ T_4\ \overline{T_9}\ \overline{T_6}\ T_7) \end{cases}$$

$$S' = \begin{cases} P_1\ (T_1\ T_3\ T_5\ T_8\ T_{10}) \\ P_2\ (T_2\ T_4\ \overline{T_6}\ \overline{T_9}\ T_7) \end{cases}$$

(4) 복제(reproduction)

룰렛트 휠(roulette wheel) 선택방법은 각 개체들의 평가함수값의 역수를 한 세대의 모든 개체들의 평가함수값의 역수값들의 합으로 나눔으로써 각각의 개체들의 선택확률을 계산하고, 이를 누적시켜 룰렛트 휠을 구성하는 방법이다. 이 방법에서 각각의 개체들이 선택될 확률은 평가함수값에 반비례(개체의 우수한 정도에 비례)하여 결정된다. (3절의 단계 4 참조) 이러한 선택방법은 주어진 문제의 최적화목표

에 잘 부합하는 우수한 해에 대하여 유전적 조작 과정에 참여할 수 있는 기회를 더 많이 부여함으로써 우수한 해의 특징적인 정보가 다음 세대의 주도적 정보로 발현되도록 하는 것이다.

그러나, 룰렛트 휠 선택방법에 있어서 개체들은 임의로 발생시켜 얻은 난수를 포함하는 개체의 누적 활률구간에 의해 선택되므로 우수한 개체가 누락되는 경우도 있다. 이러한 경우는 우수하지 않은 개체들을 다음 세대로 계속하여 유전시킴으로써 최적해에서 멀리 떨어진 해공간들을 탐색하게 되어 바람직하지 않은 결과를 도출할 수도 있다. 그리고 이 방법에 의해 선택된 개체들도 비록 그 세대내에서는 우수한 개체들이라 하더라도 S^{**} 보다는 열등할 수도 있다.

본 연구는 이러한 약점을 보완하기 위하여 각 세대마다 S^{**}를 그 세대의 마지막 개체로서 복제하여 매우 우수한 개체(S^{**}) 주위를 계속적으로 탐색하게 병행함으로써 최적해 또는 최적해에 가까운 우수한 해주변의 탐색 공간을 확대시키는 것이다.

그러나 이러한 복제방법(reproduction operator)은 국부최적해(local optimum)일지도 모르는 S^{**}주변에 국한된 개체만을 탐색하는 결점이 있다. 따라서 S^{**}를 제외한 나머지 모든 개체들은 기존의 GA과정을 거치게 함으로써 탐색 범위가 국부최적해에 수렴하는 현상을 방지할 수 있다.

2. 4 CP/ MISF 방법

GA는 초기 해집단의 생성, 룰렛트 휠 선택방법, 유전적 조작 등에서 난수를 빈번하게 사용한다. 이렇게 난수를 많이 사용하는 것은 의도적으로 우연성을 활용하려는 것이다. 의도적인 우연성의 활용은 국부 최적해에 대한 탐색에 국한되지 않고 전체최적해(global optimum)를 탐색할 가능성을 확률적으로 배제하지 않기 위해서이다. 그러나 탐색 과정이 난수에 의해 조절되므로 경우에 따라서는 우수한 해주변보다는 열등한 해주변에 대한 탐색이 많이 시도될 수 있다. 이렇게 우수한 해주변보다 열등한 해주변을 탐색하는 경우가 자주 발생하게 되면 GA는 바람직하지 못한 해를 도출할 수도 있다. 이러한 GA의 결점을 보완하기 위해 본 연구에서는 CP/MISF를 병행한다.

CP/MISF는 Kasahara[11]가 고안한 발견적 알고리듬으로 기존의 CPM을 보완한 방법이다. 즉, CP/MISF는 과업의 레벨(level : l_i)과 후속공정의 수를 고려하여 과업의 우선순위 목록(priority list)을 만들고 이에 따라 일정계획을 수행하는 방법이다. 이 방법은 해집단을 동시적으로 진화시키는 GA와 달리 우수하고 유일한 해를 구하며 최적해는 보장하지 못하지만 최적해에 가까운 근사해를 구할 수 있다.

CP/MISF 방법은 다음과 같은 단계로 이루어진다.

단계 1: 각각의 과업에 대해 레벨을 결정한다. 레벨은 최종 노드로부터 각각의 노드에 이르는 가장 긴 경로의 길이로 정해진다. 이 경로의 길이는 과업을 완성하는 데 걸린 시간을 합계한 값들 중 가장 큰 값이다. 이를 식으로 나타내면 다음과 같다.

$$l_i = \max_{k} \sum_{j \in \pi_k} t_j \quad (k = 1,2,\cdots,h) \cdots\cdots\cdots\cdots (6)$$

π_k : 최종노드로부터 각각의 노드에 이르는 k번째 경로

t_j : 과업의 처리시간

i : 과업의 번호 ($i=1,2,\cdots,n$)

h : 최종노드로부터 각각의 노드까지 존재하는 경로의 가지수

단계 2: 레벨과 후속 공정의 수에 따라 내림차순으로 정렬하여 우선 순위 목록을 작성한다.

단계 3: 우선 순위 목록에 따라 목록일정계획(list scheduling)을 수행한다. 즉, 과업간의 선행관계를 유지하면서 목록에 정리된 순서대로 과업을 프로세서에 할당한다. 여기서 선행관계를 유지한다는 것은 선행공정의 집합에 속한 모든 과업이 완료된 후에 비로소 그 과업이 시작되어야 함을 의미한다.

3. 알고리듬의 절차

본 연구의 알고리듬은 [그림 3]과 같이 구성된다. 알고리듬의 전체적인 개요는 단계 1에서 단계 6까지는 복제를 추가한 GA로 해를 구하고, 단계 7에서는 CP/MISF로 해를 구한다. 그리고 두가지 방법으로 얻은 해를 비교하여 더 우수한 해를 최종해로 선택하게 되어 있다.

알고리듬의 구체적인 절차는 다음과 같다.

단계 1 : 초기값을 설정한다.

p_a, p_b, p_c, p_d, 해집단 크기, 최대 세대수를 설정한다.

단계 2: GA에 의해서 초기 해집단을 생성한다. (2절 참조)

단계 3: 각 개체의 평가함수를 계산한다. (2절 참조)

단계 3. 1: 각 프로세서의 완료 시간 중 가장 큰 값을 그 개체의 최대완료시간으로 정한다. 모든 개체들에 대해 다음과 같은 평가함수를 계산한다.

$$f(S)= \max_{1\le k\le m} \{t_k(S)\}$$

단계 3. 2: S^*_{gen}가 S^{**}보다 평가함수값이 작다면 S^{**}를 S^*_{gen}로 교체한다.

단계 4: 룰렛트 휠 법을 사용하여 개체를 선택한다.

룰렛트 휠 선택방법을 사용하면 평가함수값에 반비례하여 각각의 개체들이 선택된다. 그러므로 최대완료시간이 짧은 우수한 개체일수록 선택될 확률은 크다. 현재 개체를 S_1, …, S_n, 선택된 개체를 S_j, 각 개체의 평가함수의 역수의 합계를 F, 각각 개체의 선택 확률을 p_i, 누적 확률을 q_i라 하면 각각의 개체들을 선택하는 방법은 다음과 같다.

$$F = \sum_{i=1}^{popsize} \frac{1}{f(S_i)} \quad \cdots\cdots(7)$$

$$p_i = \frac{\frac{1}{f(S_i)}}{F},\ i=1,\ 2,\ \cdots,popsize \quad \cdots\cdots(8)$$

$$q_0 = 0,$$

$$q_i = q_{i-1}+p_i,\ i=1,2,\cdots,popsize \quad \cdots\cdots(9)$$

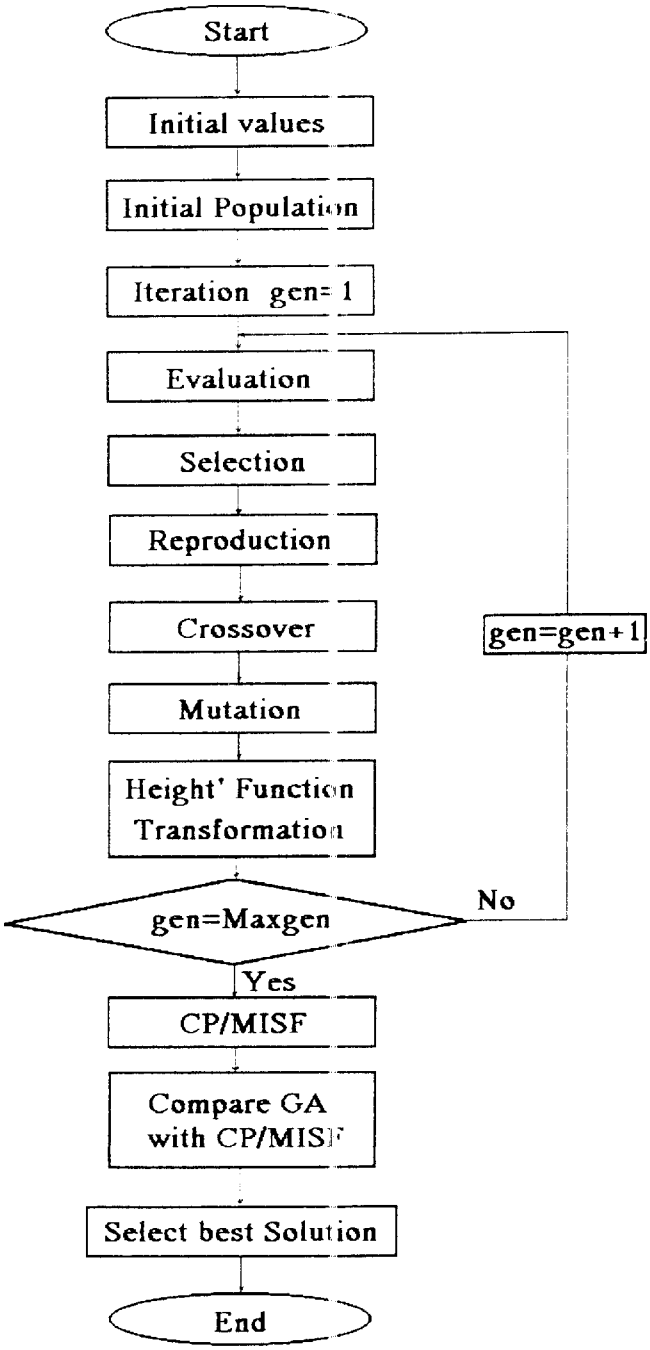
난수 r_j를 발생시켜 $r_j \le q_1$ 이면, $S_j=S_1$, $q_{i-1} < r_j \le q_i$이면 $S_j=S_i$

즉, 발생된 난수가 속한 누적확률 구간을 찾아 그에 해당하는 개체를 선택한다. 누적확률 구간의 크기는 선택확률에 비례한다.

단계 5: 유전적 조작을 수행한다.

단계 5. 1 복제 : S^{**}를 세대의 마지막 개체로서 복제한다. (2절 참조)

단계 5. 2 교배 : 각각의 개체에 대하여 발생시킨 난수가 p_a보다 작으면 그 개체


Start
Initial values
Initial Population
Iteration gen= 1
Evaluation
Selection
Reproduction
Crossover
Mutation
Height' Function Transformation
gen=Maxgen
No
Yes
gen=gen+1
CP/MISF
Compare GA with CP/MISF
Select best Solution
End

[그림 3] 알고리듬의 절차

를 선택하여 교배를 수행한다.

단계 5. 3 돌연변이 : 각각의 개체에 대하여 발생시킨 난수가 p_b보다 작으면 그 개체를 선택하여 돌연변이를 수행한다.

단계 5. 4 고도함수 변환 : 각각의 개체에 대하여 발생시킨 난수가 p_c보다 작으면 개체를 선택하고, 선택된 개체 내의 모든 과업들에 대해 난수를 발생시켜 p_d보다 작으면 그 과업의 고도함수값을 선행관계를 유지하는 범위내의 값으로 임의로 바꾼다.

단계 6: 세대수를 1만큼 증가시키고, 최대 세대수가 될 때까지 단계 3을 반복한다. 최대 세대수가 되면 GA를 종료시키고 단계 7을 수행한다.

단계 7: CP/MISF를 사용해서 해를 구한다.

단계 8: CP/MISF와 GA로 구한 해를 비교하여 우수한 해를 최종해로 한다.

4. 알고리듬의 수행 결과 및 비교 분석

여기에서는 [그림 1]과 [그림 4]의 과업 도표를 갖는 세가지 수치예제를 본 연구의 알고리듬으로 수행시키고 그 결과치를 기존연구와 비교 분석한다. 세가지 수치예의 초기값은 다음과 같이 설정해 주었다.

p_a=0.6, p_b=0.3, p_c=0.5, p_d=0.1, 해집단 크기=10, 최대 세대수=5000

4. 1 수치예 1

[그림 1]의 과업도표(10개 과업)를 대상으로 수치예제를 수행한 결과는 다음과 같다.

〈표 1〉 그림 1의 과업도표에 대한 결과 비교

프로세서수	최 적 해	Gen의 GA결과[7]	본 연구의 GA결과	CP/MISF	본 연구에서 선택된 최종해
2	13	13	13	14	13

4. 2 수치예 2

[그림 4]의 과업도표는 Stanford Manipulator의 동작을 제어하기 위한 88개의 연산에 시작 노드와 끝 노드를 추가한 90개의 과업을 선행관계를 고려하여 과업도표로 표현한 것이다. [그림 4](90개 과업)를 대상으로 수치예제를 수행한 결과와 기존 연구와의 비교는 〈표 2〉와 같다.

선택된 최종해의 ( )는 본 연구 알고리듬에서 최종적으로 선택한 방법을 의미한다. 이 예제에서 프로세서수 m≥7인 경우에는 주공정시간(critical path time)에 해당하는 570의 값을 구하였으므로 m=7이 최적해이다.

알고리듬은 C언어로 프로그래밍하였다.

〈표 2〉 Stanford Manipulator에 대한 기존 연구와 본 연구와의 결과 비교

프로세서수	최적해	Hou의 GA결과[9]	Gen의 GA결과[7]	본연구의 GA결과	CP/MISF	본 연구에서 선택된 최종해
2	1242	1246	1247	1247	1254	1247(GA)
3	843	938	903	856	861	856(GA)
4	659	774	765	693	694	693(GA)
5	586	679	671	612	610	610(CP/MISF)
7	570	603	611	570	570	570 (GA,CP/MISF)
10	570	590	570	570	570	570 (GA,CP/MISF)

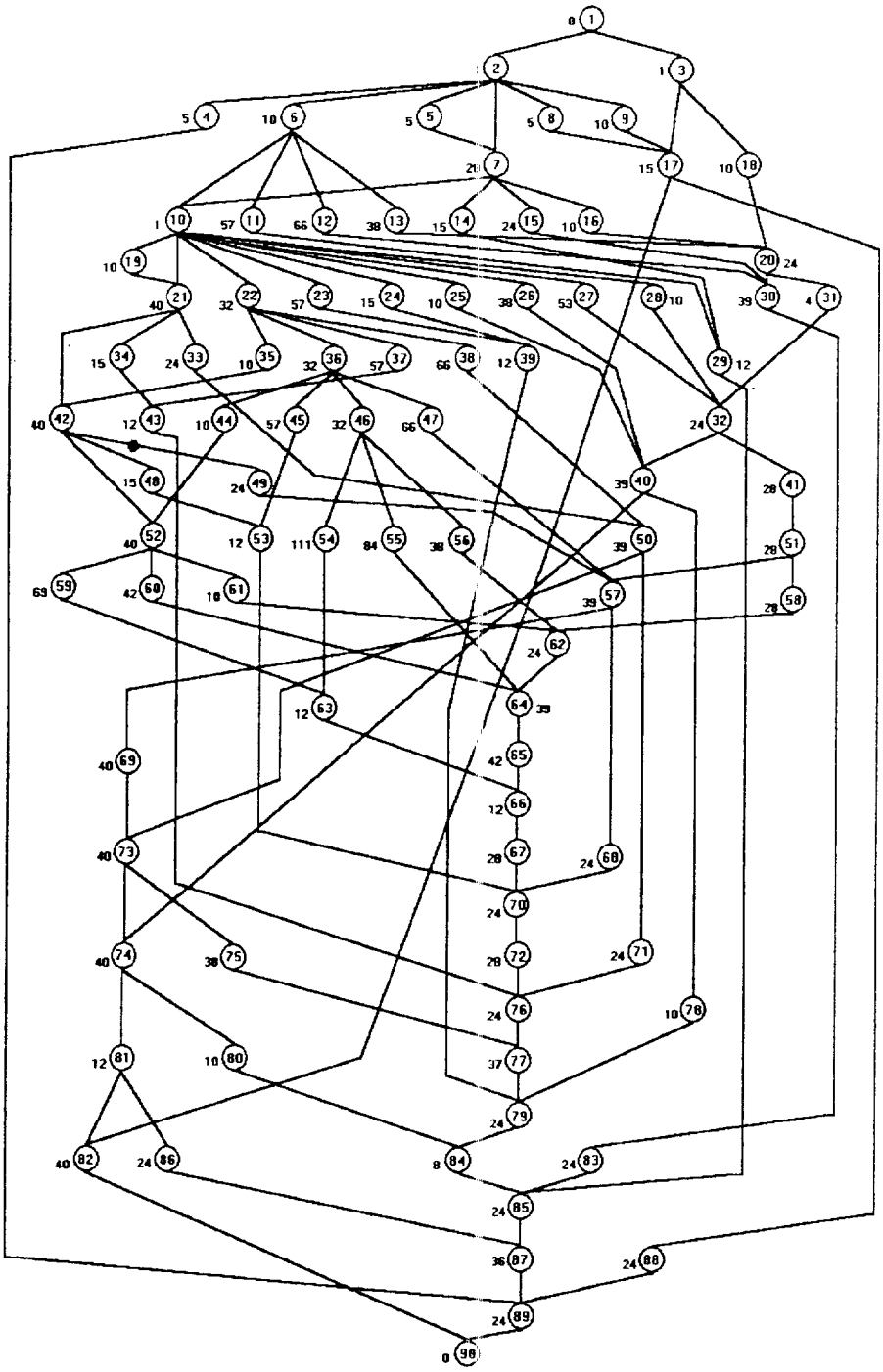
4. 3 수치예 3

이번 수치예는 [그림 4]의 과업들의 처리시간을 난수를 사용하여 〈표 3〉과 같이 변형하였다. 과업들간의 선행관계는 [그림 4]와 동일하다.

〈표 3〉 수치예 3의 과업들의 처리시간

과업번호	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
처리시간	0	28	43	82	23	99	91	28	11	67	23	69	11	42	36	13	57	63	18	78
과업번호	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
처리시간	34	52	74	27	48	68	26	22	63	25	74	7	96	58	69	28	90	15	82	29
과업번호	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
처리시간	97	17	85	32	28	48	13	65	28	38	27	60	98	4	26	92	88	79	70	91
과업번호	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
처리시간	62	7	65	50	37	41	68	82	67	51	27	71	80	20	16	72	48	39	12	32
과업번호	81	82	83	84	85	86	87	88	89	90										
처리시간	39	35	68	74	6	88	94	18	71	0										

수치예 3의 수행결과는 다음의 〈표 4〉와 같다.



[그림 4]Stanford manipulator의 과업도표

〈표 4〉 수치예 3의 수행결과

프로세서수	Hou의 결과	Gen의 결과	본 연구의 GA결과	CP/MISF	본 연구에서 선택된 최종해
2	2441	2329	2216	2269	2216(GA)
3	1932	1784	1563	1631	1563(GA)
4	1652	1566	1336	1337	1336(GA)
5	1529	1399	1253	1255	1253(GA)
6	1457	1339	1219	1219	1219 (GA, CP/MISF)
10	1311	1221	1219	1219	1219 (GA, CP/MISF)

위 예제에서 경우에는 프로세서수 m≥6 이상인 경우에는 주공정시간에 해당하는 값 1219의 값을 구하였으므로 m=6 이 최적해이다.

4. 4 수치예의 수행결과

전술한 수치예 1, 2, 3을 본 연구의 알고리듬으로 수행한 결과 프로세서 m이 많아(Stanford Manipulator의 경우 m≥7) 질수록 본 연구의 GA와 CP/MISF가 동시에 최종해로 선택되었다. 더구나 수치예 2에서는 m≥7, 수치예 3에서는 m≥6의 경우 주공정시간(critical path time)에 해당하는 최적해(최적일정계획)를 구하였다. 이것은 프로세서수가 많아질수록 수많은 대체적 일정계획이 가능하므로 과업들 사이의 유휴시간이 존재할 가능성이 없어지기 때문이다. 한편 프로세서수가 적은 (m≤5) 경우에는 거의 대부분 본 연구의 GA만이 최종해로 선택되었으며 CP/MISF보다 우수한 해를 구하였다. 그리고 지면상의 관계로 소개하지 못한 그 밖의 수치예에서도 동일한 경향을 보였다. 따라서 프로세서수의 전체적인 입장에서는 복제방법을 추가한 본 연구의 GA가 CP/MISF보다 우수한 해를 구하는 방법이라 할 수 있다. 더구나 현실적인 문제에서는 프로세서(예로서 로봇트, 병렬처리 컴퓨터 등)의 수가 많지 않은 것이 더욱 일반적이다.

수치예제는 486PC상에서 실행되었다. 90개의 과업이 있는 대단위 규모인 Stanford Manipulator의 경우에 branch and bound법으로 최적해를 얻기 위하여는 열흘 이상의 실행시간이 필요하며 현실적으로 거의 실용성이 없다. 그러나 본 연구의 알고리듬으로는 PC상에서 약 10분 내외의 실행시간으로 최적해 또는 최적해에 근접한 해를 구하였다.

5. 결 론

다중 프로세서 일정계획(MPS)은 동일한 능력을 가진 여러대의 프로세서에 복수의 과업들을 일정계획하는 것이다. 다중 프로세서 시스템에서 과업들의 처리시간이 다르고 선점(pre-

emption)이 허락되지 않으며 과업들 간에 선행관계가 존재한다면 일정계획은 매우 어렵다.

본 연구는 이러한 MPS문제에 대해 복제방법을 추가한 새로운 GA와 CP/MISF방법을 병행한 후 두 방법의 해를 비교하여 우수한 해를 최종해로 제시하는 알고리듬으로써 최적해 또는 최적해에 근접한 일정계획을 구할 수 있었다.

MPS문제에 대한 기존의 GA연구에 있어서 룰렛트 휠 선택방법은 각 개체(일정계획)의 누적확률구간의 크기(개체의 우수한 정도)에 비례하여 개체를 선택한다. 따라서 우수한 개체가 다른 개체들보다 선택될 확률이 높은 것은 사실이지만 이러한 경우 각 개체간의 상대적인 우열을 과소평가할 우려가 있다. 그러므로 세대교체가 진행되는 동안 우수한 개체를 누락시키는 경우도 발생하여 우수한 개체 주변에 대한 계속적인 탐색이 중단된다. 본 연구에서는 복제라는 유전적 조작을 추가함으로써 우수한 개체 주변을 계속적으로 탐색하도록 하였다. 그 결과 최적해 또는 최적해 주변의 해공간에 대한 탐색이 계속적으로 시도되어 〈표 2, 4〉에서와 같이 기존의 GA연구[7][9]보다 우수한 일정계획을 구할 수 있었다.

한편 GA는 초기 해집단의 생성, 룰렛트 휠 선택방법, 유전적 조작 등에서 빈번하게 난수를 사용한다. 이러한 의도적인 우연성의 활용은 국부최적해에 대한 탐색에 국한되지 않고 전체최적해를 탐색할 가능성을 확률적으로 배제하지 않기 위해서이다. 그러나 이러한 난수의 사용으로 인해 GA에서 해를 탐색하는 과정은 동일하지 않으며 GA를 통해 얻은 해의 평가함수값은 변동하는 것이 일반적이다. 만약 우수하지 못한 해주변의 탐색이 주로 시도되었다면 만족스럽지 못한 해를 얻을 수도 있다. 이러한 경우를 보완하기 위해 CP/MISF방법을 본 연구의 알고리듬에 병행시킴으로써 최적해에 더욱 근접한 일정계획을 구할 수 있었다.

최종해는 프로세서수 m이 많아(m≥7)질수록 복제를 추가한 GA와 CP/MISF가 동시에 선택되었다. 한편 프로세서수가 적은(m≤5) 경우에는 대부분 복제를 추가한 GA가 선택됨으로써 CP/MISF방법보다 우수한 해를 구하였다. 따라서 전체적 관점에서는 본 연구의 GA가 CP/MISF와 기존의 GA방법보다 가장 우수한 방법이라 하겠다. (4. 4 참조)

현재의 고도함수는 GA로 MPS문제를 해결하는 데 핵심적인 역할을 수행하지만 고도함수 자체가 문제의 새로운 제약조건으로 작용하여 GA의 탐색공간을 제한한다. 따라서 탐색공간을 더욱 확대시킬 수 있는 새로운 개념의 개발이 추후 연구과제로 절실히 요구된다.

참 고 문 헌

[1] 김진규, 윤덕균, 혼합형 유전해법을 이용한 비대칭 외판원문제의 발견적해법, 공업경영학회지 제 18권 제 33집, pp. 111-118, 1995.

[2] Chen, C., G. Lee and E. S. Hou, *Efficient Scheduling Algorithms for Robot Inverse Dynamics Computation on a Multiprocessor System*, IEEE Transactions on Systems, Man, and Cybernetics, Vol 18. No 5, pp. 729-743, September 1988.

[3] Davis, L., *Genetic Algorithms and Simulated Annealing*, A volume in the

Pitman Series of Research Notes on Artificial Intellingence, London, Pitman, 1987.

[4] Davis, L., *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, 1991.

[5] Davis, L., *Job shop scheduling with genetic algorithm*, Proc. 1st ICGA and their Applications (Pittsburgh, PA), pp 136-140, 1985.

[6] French, S., *Sequencing and Scheduling*, John Wiley, 1982.

[7] Gen, M., Y. Tsujimura and E. Kubota, *Genetic Algorithm for Multiprocessor Scheduling Problems*, 10th Fuzzy System Symposium, Japan Fuzzy Association, pp 43-46, June 1994.

[8] Goldberg, D. E., *Sizing populations for serial and parallel genetic algorithms*, Proceeding of the Third International Conference on Genetic Algorithms and Their Applocations, San Mateo, Calif., Morgan Kaufman.

[9] Hou, E. S. H., H. Ren and N. Ansari, *Efficient Multiprocessor Scheduling Based on Genetic Algorithms*, Dynamics, Genetic, and Chaotic Programming, Branko Soucek and the IRIS Group, John Wiley & Sons, Inc., 1992.

[10] Kasahara, H. and S. Narita, *Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing*, IEEE Transactions on Computers, Vol. C-33, No. 11, pp. 1023-1029, November 1984.

[11] Kasahara, H. and S. Narita, *Parallel Processing' of Robot-Arm Control Computation on a Multimicroprocessor System*, IEEE J. of Robotics and Automation, Vol. RA-1, No. 2, pp. 104-113, June 1985.

[12] Kasahara, H. and S. Narita, *Parallel processing scheme for multiprocessor continuous-time dynamical system simulator*, J. Japan Soc. simulation Technol., Vol. 2, pp. 142-151, 1983.

[13] Kinnear, K. E., *Advances in Genetic Programming*, A Bradford Book, 1994.

[14] Luh, J. Y. S. and C. S. Lin, *Scheduling of parallel computer for a computer-controlled mechanical manipulator*, IEEE Trans. Syst., Man, Cybern., Vol. 12, pp. 214-234, 1982.