Genetic Algorithm을 이용한 다중 프로세서 일정계획문제의 효율적 해법

(An Efficient Method for Multiprocessor Scheduling Problem Using Genetic Algorithm)

吳 勇 周　　朴 承 憲

仁荷大學校 産業工學科

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). A new 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. 서론

다중 프로세서(Multiprocessor) 시스템은 처리효율의 이점 때문에 정보처리 시스템뿐만 아니라 로봇트 제어와 동적 시스템의 고속 시뮬레이션 등 여러 분야에서 사용되고 있다. 이것은 병렬처리의 효율면에서 최대완료시간을 최소로 하고 또한 프로세서의 수를 적게 하는 것이 바람직하다. 이러한 문제는 멀티프로세서 스케쥴링(Multiprocessor Scheduling : MPS)으로 정식화 된다.

하나의 구체적인 예로서 MPS문제는 컴퓨터 시스템에 있어서 m개의 프로세서(processor)로 n개의 과업(task)을 처리하여 최대완료시간을 최소로 하는 문제이다. 이 문제의 최적 계획(Scheduling)은 복수의 프로세서 상에서 과업들간의 선행관계를 유지하면서 가능한 모든 과업을 단시에 완료시키는 것이다. 지금까지는 CPM(Critical Path Method)과 HLFET(Highest Levels First with Estimated Times) 등의 해법을 사용하여 왔으나 최근에는 Genetic Algorithm(GA)을 이용한 해법이 제안되었다.

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

본 연구는 MPS 문제를 대상으로 기존의 방법에 새로운 유전적 조작을 도입한 GA를 이용하나 GA가 좋지 못한 해를 얻을 경우를 대비해 발견적 알고리듬인 CP/MISF(Critical Path/Most Immediate Successors First) 방법을 병행함으로써 두 가지 방법을 통해 얻은 해를 비교하여 더 우수한 해를 최종해로 제시하는 알고리듬을 개발한다. 본 알고리듬은 최적해(계획)를 보장하지는 않으나 실용적이며 최적해에 가까운 근사해를 얻을 수 있다. 또한 수치예를 통하여 본 연구의 알고리듬에 의한 유효성을 검토한다.

2. 정식화 및 유전적 조작

본 연구는 Hou[8], Gen[6], Kasahara[10] 등이 제안한 알고리듬을 기초로 그 유효성을 검토하고 새로운 유전적 조작의 도입과 CP/MISF 방법을 병행함으로써 보다 효과적인 알고리듬을 제시한다.

2.1 개체의 표현과 고도함수

일정계획은 그림 1과 같이 과업들간에 선행관계를 만족해야 하며 각 과업을 단 한번만 포함한다. 각 과업간의 선행관계를 고도함수(height' function)로 표현하고 이것에 의해 개체(Individual, 이후 실행가능한 일정계획을 개체 또는 개체 S로 칭한다)를 표현한다. 그림 1의 과업 도표(task graph)의 각각의 과업에 대해 다음과 같은 고도함수 $height'(T_j)$를 정의한다.

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

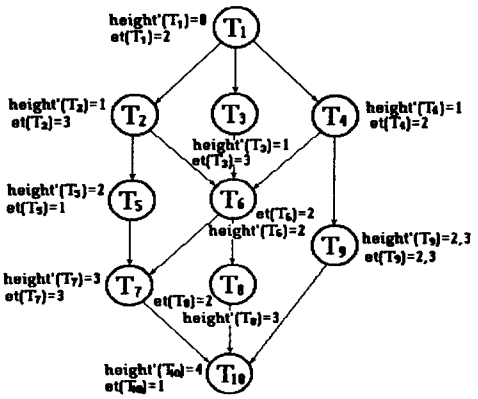
$$height(T_i) = \begin{cases} 0, & PRED(T_i) = \varphi \\ 1 + \max_{T_j \in PRED(T_i)} \{height(T_j)\}, & otherwise \end{cases} \tag{2}$$

T_i : i번째 작업

$PRED(T_i)$: T_i의 선행공정 집합

$SUCC(T_i)$: T_i의 후속공정 집합

$rand$: 整數의 亂數



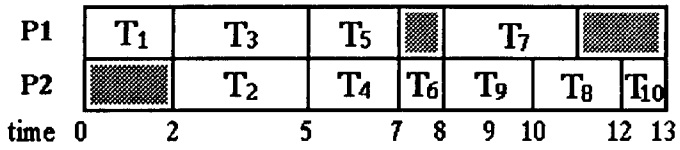
[그림 1] 선행 관계를 갖는 10개 과업의 과업 도표

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

P_1 : $height'(T_1) \le height'(T_3) \le height'(T_5) \le height'(T_7)$

P_2 : $height'(T_2) \le height'(T_4) \le height'(T_9) \le height'(T_6) \le height'(T_8) \le height'(T_{10})$

이것을 간트 차트로 옮기면 다음과 같다.



[그림 2] 간트 차트

위의 간트 차트로 표현된 개체 S(일정계획)는 다음과 같다.

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

2.2 초기 집단과 평가 함수

하나의 개체 S는 2.1에서 전술한 과정을 거쳐 만들어 진다. 초기집단의 개체들은 초기값으로 지정하는 모집단 크기(population size : 변수명은 popsize임)만큼 2.1에서 전술한 과정을 반복함으로써 생성된다.

개체의 각각의 프로세서의 완료시간 중 가장 큰 값을 그 개체의 최대완료시간으로 하고 이 값을 개체의 평가 함수로 다음과 같이 정의한다.

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

$f(S)$: 개체 S의 평가 함수

$t_k(S)$: 개체 S 중 프로세서 k의 완료 시간

p : 프로세서의 수

다음은 각 세대에 있어서 최대완료시간이 가장 작은 개체와 그 이전의 모든 세대에 있어서 나타난 가장 좋은 개체를 비교하여 우수한 개체를 다음 세대에 정보로 남김으로써 최종적으로 가장 좋은 개체 S^*가 얻어지도록 하며 S^*는 다음과 같이 정의한다.

$$S^* = \underset{S_i}{argmin}\{f(S_i)\}$$

S_i : i번째 개체. $i = 1, 2, \cdots, popsize$.

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

2.3 유전적 조작

유전적 조작은 현재 모집단의 개체들로부터 새로운 개체를 만드는 것이다. 새로운 개체는 현재 모집단의 개체들을 재배열하거나 결합함으로써 만들어 진다. 이렇게 조작하는 과정에서 치사 유전자(실행 불가능한 일정계획)가 발생하지 않도록 고려하여야 한다. 본 연구에서는 교배(crossover), 돌연변이(mutation), 고도함수 변환(height function transformation), 복사(replication) 등 4가지 유전적 조작(genetic operators)을 수행하며 그 의미와 역할은 다음과 같다.

교배(crossover)

난수에 의해 발생된 임의의 개체, 두 프로세서들 간에 임의로 선택된 고도함수값을 갖는 과업 뒷부분(이탤릭체 부분)을 서로 교환한다. 그림 2의 일정계획을 예로 하면 다음($S \rightarrow S'$)과 같이 변환된다.

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

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

돌연변이(mutation)

난수에 의해 발생된 임의의 개체, 두 프로세서들 간에 임의로 선택된 고도함수값을 갖는 과업들만을 서로 교환한다. 다음 예에서는 고도함수값이 2인 과업들(이탤릭체부분)을 서로 교환한다.

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

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

고도함수 변환(height function transformation)

초기집단을 생성할 때에 고도함수를 고정시키면 탐색공간이 제한되기 때문에 선행관계를 유지하는 범위 내에서 과업의 고도함수값을 변환할 필요가 있다. 즉, 과업의 고도함수값을 변환가능한 고도함수의

범위내에서 바꾸어 탐색공간을 확대시킴으로써 최적해에 접근할 가능성이 커지게 된다. 다음 예에서는 T_9의 고도함수값이 2에서 3으로 변환되었다.

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

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

복사(replication operator)

룰렛트 휠(roulette wheel) 선택방법은 각 개체들의 평가함수값을 한 세대의 모든 개체들의 평가함수값들의 합으로 나눔으로써 각각의 개체들의 선택확률을 계산하는 것이다. 즉, 각각의 개체들이 선택될 확률은 평가함수값의 우수한 정도에 비례하여 결정된다. 이러한 선택방법은 우수한 해를 다음 세대에 전달하여 우수한 해 주위를 계속적으로 탐색하도록 함으로써 더 나은 개체를 찾도록 하는 것이다. 그러나, 룰렛트 휠 선택방법에 있어서 개체들은 임의로 발생시켜 얻은 난수를 포함하는 개체의 누적확률구간에 의해 선택되므로 우수한 개체를 선택하지 못하는 경우가 생길 수도 있다. 이러한 경우는 우수하지 않은 개체들을 다음 세대로 계속하여 유전시킴으로써 최적해에서 멀리 떨어진 공간만을 탐색하게 되어 최종 결과가 나쁘게 된다. 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 복사함으로써 매우 우수한 개체(S^*) 주위를 계속적으로 탐색하게 병행함으로써 최적해 주변의 탐색 공간을 확대시키는 것이다.

그러나 이러한 복사 방법(replication operator)은 국지적인 최적해일지도 모르는 S^*주변에 국한된 개체만을 탐색하는 결점이 있다. 따라서 S^*를 제외한 나머지 모든 개체들은 기존의 GA과정을 거치게 함으로써 탐색 범위가 국지적인 최적해에 수렴하는 현상을 방지할 수 있다.

2.4 CP/MISF method

CP/MISF[10]는 Kasahara가 고안한 발견적 알고리듬으로 기존의 CPM을 보완한 방법이다. 즉, CP/MISF는 과업의 레벨(level)과 후속공정의 수를 고려하여 과업의 우선순위 목록을 만들고 이에 따라 일정계획을 수행하는 방법이다. 그러므로 이 방법의 해는 GA와 달리 유일하며 최적해는 보장하지 못하지만 최적해에 가까운 근사해를 구할 수 있다.

CP/MISF method는 다음과 같은 단계로 이루어진다.

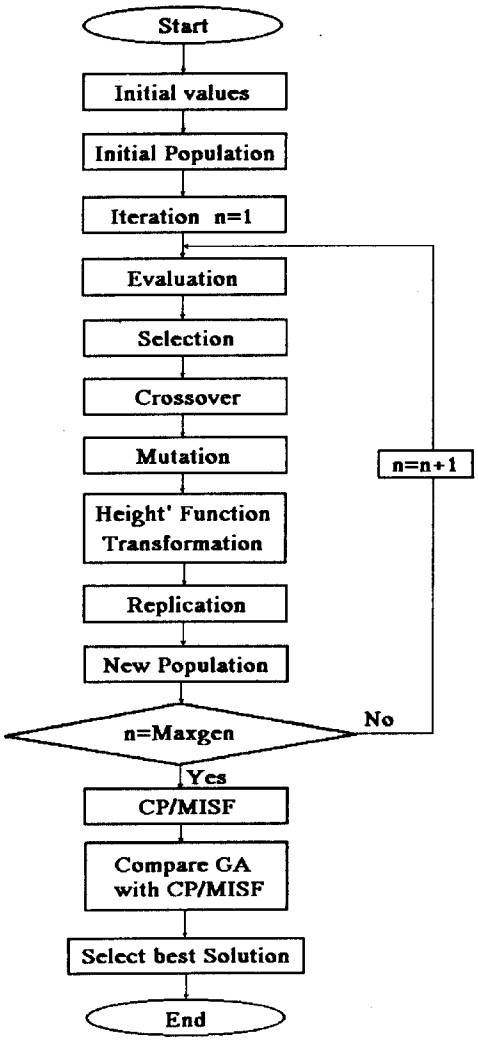
step 1: 각각의 작업에 대해 레벨을 결정한다. 레벨은 최종 노드로부터 각각의 노드에 이르는 가장 긴 경로의 길이로 정해진다.

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

step 3: 우선 순위 목록에 따라 목록 일정계획(list scheduling)을 수행한다.

3. 알고리듬의 절차

본 연구의 알고리듬은 그림 3과 같다. 즉, 첫번째 replication operator(복사방법)를 추가한 GA(단계 1 ~단계 6)에 의해 해를 구한 다음 두번째 CP/MISF 방법(단계 7)에 의해 해를 구한다. 그리고 두가지 방법으로 얻은 해를 비교하여 더 나은 해를 최종해로 선택한다.



[그림 3] 알고리듬의 절차

알고리듬의 구체적인 절차는 다음과 같다.

단계 1: 초기값을 설정한다.

p_1(교배 확률), p_2(돌연변이 확률), p_3(고도함수 변환시 개체선택 확률), p_4(고도함수 변환시 과업선택 확률), 모집단 크기, 최대 세대수(Maximum generation , GA과정을 반복할 세대수 : maxgen)을 설정한다.

단계 2: GA에 의해서 초기 집단을 생성한다. (2절 참조)

단계 3: 각 개체의 평가 함수를 계산한다. (2절 참조)

단계 3.1: 각 프로세서의 완료 시간 중 가장 큰 값을 그 개체의 최대완료시간으로 정한다.

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

단계 3.2: 그 개체의 최대완료시간이 이제까지 발견된 개체들 중 최대완료시간이 가장 짧은 개체보다 우수한 경우에만 S^*를 교체한다.

단계 4: 룰렛트 휠 법을 사용하여 개체를 선택한다. 현재 개체를 $S_1, \cdots, S_i$, 선택된 개체를 S_j, 각 개체의 평가 함수의 역수의 합계를 F, 각각 개체의 선택 확률을 p_i, 누적 확률을 q_i라하고 선택하는 방법은 다음과 같다.

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

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

$$q_0 = 0,$$

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

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

단계 5: 유전적 조작을 수행한다.

단계 5.1 교배 : 각각의 개체에 발생시킨 난수가 p_1보다 작으면 그 개체를 선택하여 교배를 수행한다.

단계 5.2 돌연변이 : 각각의 개체에 발생시킨 난수가 p_2보다 작으면 그 개체를 선택하여 돌연변이를 수행한다.

단계 5.3 고도함수 변환 : 각각의 개체에 발생시킨 난수가 p_3보다 작으면 개체를 선택하고, 선택된 개체 내의 모든 작업들에 대해 난수를 발생시켜 확률 p_4보다 작으면 그 과업의 고도함수값을 변환 가능한 값으로 임의로 바꾼다.

단계 5.4 복사 : S^*를 세대의 마지막 개체로서 복사한다. (2절 참조)

단계 6: 다음 세대의 새로운 집단을 생성한다.

세대수(Generation : gen)를 1만큼 증가시키고, 최대 세대수가 될 때까지 step 4를 반복한다. 최대 세대수가 되면 GA를 종료시키고 step 7을 수행한다.

단계 7: CP/MISF를 사용해서 해를 구한다.

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

4. 알고리듬의 수행 결과 및 비교 분석

여기에서는 그림 1과 그림 4의 과업 도표를 갖는 두가지 수치예제를 통해 본 연구의 알고리듬을 수행시키고 그 결과치를 기존연구와 비교 분석한다. 두가지 수치예의 초기값은 다음과 같이 설정해 주었다.

p_1=0.6, p_2=0.3, p_3=0.5, p_4=0.1, 모집단 크기=10, 최대 세대수=5000

4.1 수치예 1

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

프로세서수	최적해	Gen의 결과	CP/MISF	복사방법을 추가한 GA결과	선택된 최종해
2	12	12	14	12	12

4.2 수치예 2

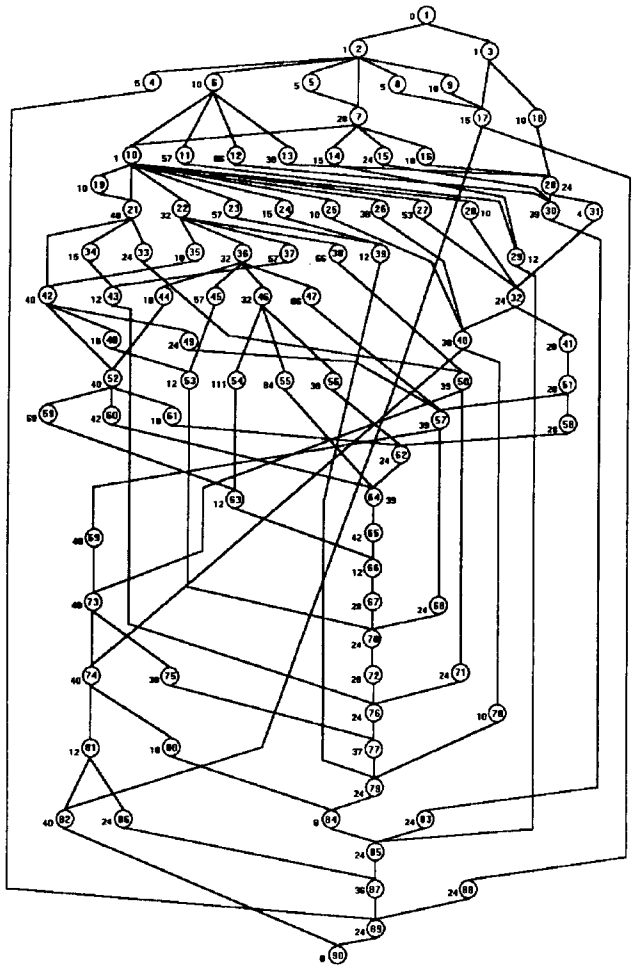
그림 4(90개 과업)를 대상으로 수치예제를 수행한 결과와 기존 연구와의 비교는 표 1과 같다.

<표 1> 본 연구와 기존 연구와의 결과 비교

프로세서수	최적해	Hou의 결과[8]	Gen의 결과[6]	CP/MISF	복사방법을 추가한 GA결과	선택된 최종해
2	1242	1246	1247	1254	1247	1247(GA)
3	843	938	903	861	856	856(GA)
4	659	774	765	694	693	693(GA)
5	586	679	671	610	612	610(CP/MISF)
6	570	627	628	570	578	570(CP/MISF)
10	570	590	570	570	570	570 (GA,CP/MISF)

선택된 최종해의 ( )는 본 연구 알고리듬에 의해 최종적으로 선택한 방법을 의미한다.

알고리듬은 C언어로 프로그래밍하였다.



[그림 4] Stanford manipulator의 과업도표

5. 결론

본 연구는 MPS문제의 효율적 해를 구하는 알고리듬을 검토하였다.

알고리듬의 일부는 GA의 유전적 조작에 복사방법(replication operator)을 추가시킴에 따라, 2.3에서 설명한 기존 연구의 룰렛트 휠 방법의 결점을 보완하고 최적해 주변의 탐색공간을 확대하였다. 그 결과 표 1에서와 같이 기존 연구 [8], [6]보다 최적해에 가까운 일정계획(개체)을 구할 수 있었다.

한편으로는 GA방법과는 달리 단 한번의 계산으로 근사해를 구하는 CP/MISF방법을 알고리듬에 병행시킴으로써 두 방법 중 좋은 해를 최종해로 선택하여 GA의 단점을 보완하는 동시에 최적해에 근접한 정도가 높은 일정계획(표 1참조)을 구할 수 있었다.

선택된 최종해는 프로세서수가 증가함에 따라 최적해와 일치하는 경향을 보였다.

6. 참고 문헌

[1] 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.

[2] Davis, L., *Genetic Algorithms and Simulated Annealing*, A volume in the Pitman Series of Research Notes on Artificial Intellingence, London, Pitman, 1987.

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

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

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

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

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

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

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