평균장 어닐링과 유전자 알고리즘을 결합한 부하균형기법

(A Load Balancing Technique Combined with Mean-Field Annealing and Genetic Algorithms)

홍 철 의 † 박 경 모 ††
(Chuleui Hong) (Kyeongmo Park)

요 약 본 논문에서는 병렬처리에서 중요한 이슈인 부하균형 문제에 대한 새로운 솔루션을 소개한다. 솔루션으로 제안하는 MGA 기법은 평균장 어닐링(MFA)과 유전자 알고리즘(GA)의 장점을 효과적으로 결합한 휴리스틱 부하균형기법이다. 제안된 MGA을 다른 매핑 알고리즘(MFA, GA-1, GA-2)들과의 성능 향상비를 측정하는 멀티프로세서 매핑 시뮬레이션을 개발하였다. 휴리스틱 매핑 기법의 합성을 통하여 기존의 방법보다 수행시간은 오래 걸리는 대신 솔루션 품질, 즉 최대종료시간 및 통신부하에서 개선된 실험결과를 얻을 수 있다는 것을 보였다.

키워드 : 매핑, 병렬처리, 시뮬레이티드 어닐링, 평균장 어닐링, 유전자 알고리즘

Abstract In this paper, we introduce a new solution for the load balancing problem, an important issue in parallel processing. Our heuristic load balancing technique called MGA effectively combines the benefit of both mean-field annealing (MFA) and genetic algorithms (GA). We compare the proposed MGA algorithm with other mapping algorithms (MFA, GA-1, and GA-2). A multiprocessor mapping algorithm simulation has been developed to measure performance improvement ratio of these algorithms. Our experimental results show that our new technique, the composition of heuristic mapping methods improves performance over the conventional ones, in terms of solution quality with a longer run time.

Key words : mapping, parallel processing, simulated annealing, mean-field annealing, genetic algorithms

1. 서 론

분산메모리 멀티프로세서 시스템에서 태스크들이 프로세서들에 적절하지 않게 배치되면 낮은 프로세서 이용률과 불균형 작업부하로 인하여 손실을 입는다. 이 문제는 매핑(mapping) 문제의 한 형태인 태스크-프로세서 매핑 문제로 연결된다[1-6,8]. 태스크 프로세서 할당이란 병렬 시스템의 여러 프로세싱 노드들에 병렬 프로그램의 통신 태스크 세트를 매핑하는 것이다. 태스크-프로세서 노드의 매핑의 목적은 솔루션의 품질 즉, 태스크 수행시간을 최소화하는 것이다. 이러한 매핑 목적 달성을 위해 프로세서들 간의 메시지 전달의 통신 비용을 최소화시키면서 태스크들을 노드들에 균등하게 배분해야한다. 즉, 부하균형(load balancing)을 통한 병렬 시스템 성능을 최대화시키는 것은 주요 목표이다.

병렬처리에서 부하균형을 통해 멀티프로세서 노드들의 처리 성능을 최대화시키는 것은 중요한 연구주제이다. 부하균형의 목적은 병렬시스템의 모든 프로세서 노드들의 작업부하를 동등하게 배분하고 노드간의 통신 트래픽을 적게 하는 것이다. 본 연구에서는 분산메모리 멀티프로세서 노드를 갖는 병렬처리 시스템에서의 부하균형 문제에 새로운 알고리즘을 소개한다. 제안 알고리즘은 평균장 어닐링(mean-field annealing)[1,4]과 유전자 알고리즘(genetic algorithms)[2,5]에 기반을 둔다.

제안 알고리즘은 휴리스틱스(heuristics) 최적화 기법으로 유전자 알고리즘, 평균장 어닐링, 시뮬레이티드 어

· 본 연구는 2006년도 가톨릭대학교 전공특성화 사업비 지원으로 이루어졌음

† 정 회 원 : 상명대학교 소프트웨어학부 교수
hongch@smu.ac.kr
†† 종신회원 : 가톨릭대학교 컴퓨터정보공학부 교수
kpark@catholic.ac.kr

논문접수 : 2004년 3월 29일
심사완료 : 2006년 5월 12일

널링(simulated annealing)[6,7] 등을 포함한다. 이런 기법들 간의 비교 연구가 부족한 실정에서 우리는 휴리스틱 알고리즘의 효과적인 결합에 의한 좋은 품질의 매핑 솔루션을 위한 선행연구로 부분적인 실험결과를 발표하였다[8]. 본 논문에서는 앞서 발표된 학술발표대회 논문[8] 보다 구체적인 설명과 함께 추가적인 시뮬레이션 성능결과를 보고하고 있다.

본 논문의 구성은 다음과 같다. 2장에서는 본 연구에서 다룰 멀티프로세서 매핑 문제를 서술한다. 3장에서는 주어진 문제를 해결하기 위한 알고리즘들을 제시하고 구현 관련 내용에 대해 기술한다. 4장에서는 시뮬레이션 연구를 통해 제시한 알고리즘들의 성능을 측정 평가한 실험결과를 설명한다. 마지막으로 5장에서 결론을 맺었다.

2. 멀티프로세서 매핑 문제

본 연구에서 다룰 태스크-프로세서 매핑 문제에 대하여 공식적으로 서술한다. 태스크 그래프 $G_T(V, E)$는 $|V| = N$, $(1, 2, \cdots, i, j, \cdots, N)$로 표시되고 공집합이 아닌 정점(vertex)의 집합이다. G_T의 정점들은 병렬프로그램의 태스크를 나타낸다. 정점 가중치 w_i는 태스크 i, $(1 \le i \le N)$에 따른 계산비용을 나타낸다. E는 V에 속하는 두 점 사이를 잇는 간선(edge)의 집합이다. 간선 가중치 e_{ij}는 간선 $(i, j) \in E$로 연결된 태스크 i와 j 간의 상호 데이타 교환량을 표시한다. 프로세서 그래프 $G_P(P, D)$는 $|P| = K$ 노드와 $|D| = \binom{K}{2}$ 간선를 갖는 모든 정점 간 간선가 존재하는 완전그래프(complete graph) 이다. G_P의 노드들은 $(1, 2, \cdots, p, q, \cdots, K)$로 목표로 하는 멀티프로세서를 표시한다. 간선 가중치 d_{pq}, $(1 \le p, q \le K)$, $p \ne q$는 프로세서 p와 q 사이의 통신비용을 나타낸다. 여기서 태스크 할당문제란 N-to-1 매핑 문제로 $M : V \rightarrow P$를 구하는 것이다. 즉 태스크 그래프 G_T의 각 정점을 프로세서 그래프 G_P의 유일한 노드에 할당하는 것이다. 각 프로세서에 균형을 맞춘 부하(*Load*)를 유지하면서 노드간 전체 통신비용(*Comm*)을 최소화하는 것이 문제다.

$$Comm = \sum_{(i,j) \in E,\, M(i) \ne M(j)} e_{ij} d_{M(i)M(j)} \qquad (1)$$

$$\mathrm{Load}_p = \sum_{i \in V,\, M(i) = p} w_i \,, \; 1 \le p \le K \qquad (2)$$

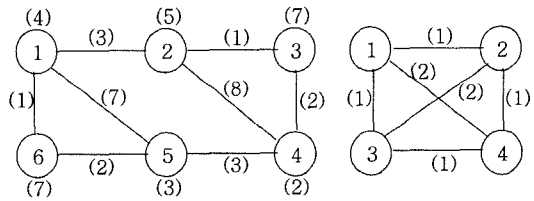
e_{ij} : 태스크 쌍 i와 j 사이의 간선 가중치

w_i : 태스크 그래프에서 태스크 i의 가중치

d_{pq} : 프로세서 그래프에서 프로세서 p와 q 사이의 간선 가중치(edge weight)

$M(i)$는 태스크 i가 매핑되는 프로세서를 표시한다. (1) 식에서 G_T의 각 간선(i, j)는 정점 i와 j가 G_P의 다른 2개 노드에 매핑이 되면, 즉 $M(i) \ne M(j)$이라면 통신비용에 부담을 준다. 그 분량은 2개 태스크 간의 상호작용한 양과 프로세서 p와 q 간의 단위 통신비용 d_{pq} $(p = M(i), q = M(j))$을 곱한 것과 같다. 프로세서 부하는 해당 프로세서 노드에 할당된 여러 개의 태스크 가중치를 합계한 것이다.

그림 1은 매핑 문제의 예를 나타낸다. 그림 1(a)는 N=6의 태스크 그래프를 나타내며, 그림 1(b)는 K=4 프로세서를 가지고 있는 2차원 메쉬(mesh)구조의 프로세서 그래프를 나타낸다. 그림 1에서 원안의 수는 태스크 또는 프로세서 명을 나타내며 괄호의 수는 각각의 가중치를 나타낸다. 즉, 태스크 그래프에서는 태스크 크기 및 두 태스크 사이의 통신량을 나타내며, 프로세서 그래프에서는 두 프로세서 사이의 hop의 수를 나타낸다. 그림 2는 주어진 매핑 문제(그림 1)에 대한 각 태스크의 프로세서 할당의 최적해를 나타낸다.



(a) 태스크 그래프 (b) 프로세서 그래프

그림 1 매핑 문제의 예 (N=6 태스크, K=4 프로세서)

i	1	2	3	4	5	6
$M(i)$	1	2	4	2	1	3

그림 2 주어진 매핑 문제(그림 1)에 대한 솔루션

3. 부하균형기법

3.1 평균장 어닐링 알고리즘

본래 평균장 어닐링(mean-field annealing, MFA)[1,4]은 열평형(thermal equilibrium) 상태에 있는 '스핀(spin)'이라 부르는 입자(particle)들 시스템의 상태를 추정하기 위해 물리학의 평균장 근사법(mean-field approximation)에서 나온 것이다. MFA 알고리즘은 시뮬레이티드 어닐링(simulated annealing, SA)과 홉필드 신경망(Hopfield neural network, HNN) 모델을 결합한 것이다. SA 기법은 최적화 문제에서 좋은 솔루션을 구하는데 많은 계산 시간이 소요되는 단점을 가지고 있다. 이 원인은 평형 상태에 도달하기 전에 개개의 온도에서 수많은 무작위 탐색을 수행하기 때문이다. MFA에서는 SA에 대한 결정적 근사법(deterministic approximation)을 이용하여, 즉 어닐링 프로세스를 통해 얻은 통

계자료를 평균함으로써 문제를 해결한다. 다시 말해 MFA는 SA[7] 기법에서 무작위로 상태를 변화시키는 것과 달리, 평균장 근사법을 사용하여 산출되는 평균값으로 대치시키는 방법으로 열(온도) 평형 상태에 빠르게 도달할 수 있다는 장점이 있다.

평균장 어닐링(MFA)은 홉필드 신경망(HNN)과 긴밀하게 관련되어 있다. HNN에서는 비동기적 갱신(asynchronous updating)에 기반을 둔 완전 그래프로 무작위로 선택된 뉴론(neuron)중 오직 하나의 단위만이 각 시간적 단계에 관여된다. 상호작용하는 뉴론들이 역동적 공동 집단 연산(dynamic collective computation) 특성으로 인하여 하드웨어 구현에 잠재적인 강점이 있다. HNN에서는 gradient descent 방법으로 엔코드된 비용함수를 최소화한다. HNN은 보통 작은 크기의 최적화 문제에는 효율적으로 적용이 가능하나, 문제 크기가 커질수록 적용이 곤란하여 확장성(scalability)이 좋지 않은 약점을 기지고 있다. MFA에서는 HNN과 같은 방법으로 상태를 표현한 다음 시스템을 열평형 상태에 도달시키기 위하여 SA에서와 같이 반복기법을 적용한다.

스핀 행렬(spin matrix)는 N 개의 태스크 행과 K 개의 프로세서 열로 구성되며 태스크-프로세서 할당의 표현 방안으로 사용된다. 즉, $N\times K$ 스핀 행렬은 솔루션을 인코드 하는데 사용된다. 스핀$(i,\ p)$의 값 s_{ip}는 태스크 i를 프로세서 p에 매핑할 확률을 표시한다. 여기에서 s_{ip}는 $0 \le s_{ip} \le 1$ 범위에서 연속 변수이다. MFA 알고리즘이 하나의 솔루션에 도달하면 스핀 값은 그 결과를 0 또는 1로 수렴한다. 만일 s_{ip}가 1로 수렴되면 태스크 i는 프로세서 p에 매핑되는 것을 의미한다. 그림 3은 2장에서 보여준 그림 2 솔루션에 대한 6개의 태스크와 4개의 프로세서로 구성된 6×4 스핀 행렬의 예를 보여준다.

	1	2	3	4
1	1	0	0	0
2	0	1	0	0
3	0	0	0	1
4	0	1	0	0
5	1	0	0	0
6	0	0	1	0

그림 3 6×4스핀 행렬의 예

태스크-프로세서 매핑 문제에 사용되는 목적(비용)함수[1]는 통신량을 줄이고 작업부하를 프로세서에 동등하게 배분하도록 다음과 같이 정한다.

$$C(s)=\frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}\sum_{p=1}^{K}\sum_{q\neq p}e_{ij}s_{ip}s_{jp}d_{pq}+\frac{r}{2}\sum_{i=1}^{N}\sum_{j\neq i}\sum_{p=1}^{K}s_{ip}s_{jp}w_iw_j \quad (3)$$

각 변수는 식 (1)(2)에서 정의된 바와 동일하고 $s_{ip}\times s_{jq}$는 태스크 i와 j가 서로 다른 프로세서 p와 q에 각각 매핑되는 확률이다. 식 (3)의 첫 번째 항에서 $e_{ij}\ s_{ip}\ s_{jq}\ d_{pq}$는 태스크-프로세서 매핑에 따른 가중치를 갖는 프로세서 간의 통신(inter-processor communication: IPC)량을 나타낸다. 따라서 비용식 (3)에서 4개의 시그마 합계 항은 태스크 그래프(G_T)의 각 간선 쌍에 대한 프로세서 그래프(G_P)의 모든 프로세서 쌍을 포함한다. 이것은 스핀 행렬의 값에 의해 표현되는 매핑을 위한 총 IPC 비용을 표시하며 최소화되어야 한다. 식 (3)두 번째 항은 프로세서에서 수행되는 부하량을 나타낸다. 즉 3개의 시그마 합계 항은 태스크들이 개개의 프로세서들에 매핑된 가중치들의 내부 곱에 대해 합산한다. 동등한 양의 태스크 가중치들이 모든 프로세서에 매핑될 때 전역적인 최소치가 나온다. 만일 불균형 매핑이면 3개의 합계 항을 불균형 매핑으로 인한 페널티를 주면서 불균형 양의 제곱으로 증가시킨다. 파라미터 r은 통신과 부하의 비를 조정하고 둘 사이의 균형을 유지하기 위한 변수이다. 따라서 목적함수의 최소값이 우리가 구하고자 하는 최적해가 된다. 비용식 (3)은 태스크를 사용할 병렬 컴퓨터 및 연결망 구조에 특성화 되지 않은 일반적인 비용식으로 실제 적용시에는 주어진 병렬처리 시스템의 환경을 반영하도록 가중치 r의 동적 조정 및 비용식 중 d_{pq}의 변화가 필요하다. 4장의 시뮬레이션 결과, r 계수를 온도에 따라 동적으로 변화시키는 것이 좋은 결과를 나타내는 것으로 나타났다.

평균장(mean-field) ϕ_{ip}는 다음과 같이 정의된다.

$$\phi_{ip}=-\frac{\partial C(s)}{\partial s_{ip}}=-\sum_{j\neq i}^{N}\sum_{q\neq p}^{K}e_{ij}s_{jq}d_{pq}-r\sum_{j\neq i}^{N}s_{jp}w_iw_j \quad (4)$$

- ϕ_{ip}는 태스크 i가 프로세서 p에 매핑될 때 얻어지는 비용함수의 감소를 나타낸다.

각각의 스핀 값 s_{ip}는 $e^{\phi_{ip}/T}$에 비례하므로 s_{ip}를 정규화하면 다음과 같다.

$$s_{ip}=\frac{e^{\phi_{ip}/T}}{\sum_{q=1}^{K}e^{\phi_{iq}/T}} \quad (5)$$

식 (3)에 의해서 C(s)는 s_{ip}에 선형 비례하므로, 비용함수 변화 ΔC는 스핀$(i,\ p)$의 변화 값 Δs_{ip}에 비례한다.

$$\Delta C=\sum_{p=1}^{K}\Delta C_{ip}=\sum_{p=1}^{K}\phi_{ip}\Delta s_{ip} \quad (6)$$

여기서 $\Delta s_{ip}=s_{ip}^{new}-s_{ip}^{old}$로 정의된다.

그림 4는 멀티프로세서 매핑 문제에 대한 MFA 알고리즘의 의사 코드로 기술한 것이다.

MFA구현 관련해 냉각스케줄(cooling scheduling)은

[1] 초기 온도($T = T_0$)를 세팅한다.
[2] 스핀 평균치 $s=\{s_{11},..., s_{ip},..., s_{NK}\}$를 초기화한다.
[3] while (T가 냉각 범위 내에 있다) do {
[4] while (not 평형 상태) do {
[5] 하나의 스핀/태스크 i를 무작위로 선택한다.
[6] 평균장을 계산한다.
i-행의 스핀의 평균장 계산한다.

$$\Phi_{ip} = -\sum_{j\neq i}^{N}\sum_{q\neq p}^{K} e_{ij}s_{jq}d_{pq} - r\sum_{j\neq i}^{N} s_{jp}w_i w_j \quad \text{for } 1\le p\le K$$

i-행의 스핀 값 계산 : $s_{ip}^{new} = e^{\Phi_{ip}/T} / \sum_{q=1}^{K} e^{\Phi_{iq}/T}$

[7] 스핀 평균치를 업데이트 한다.
업데이트에 따른 비용 변화를 계산한다.

$$\Delta C = \sum_{p=1}^{K} \Phi_{ip}(s_{ip}^{new} - s_{ip})$$

i-행의 스핀 값을 갱신한다.

$$s_{ip} = s_{ip}^{new} \qquad \text{for } 1\le p\le K$$

[8] }
[9] $T = aT$
[10] }

그림 4 매핑 문제에 대한 평균장 알고리즘의 구조

주어진 문제의 특성과 비용함수에 따라 민감하게 작용하므로 다음과 같이 적절한 스케줄을 선택한다.

- 초기 온도: T_0
  초기 온도는 태스크 수(N) 만큼 평균장 어닐링을 수행하였을 때 비용의 변화가 허용치 ε(= 0.5) 이하일 확률이 95%이상일 때의 온도로 설정한다.
- 최종 온도: T_f
  최종온도는 20회의 온도 변화동안 연속해서 비용의 변화가 ε/1000 내에 존재할 때의 온도로 설정한다.
- 임의의 온도 k에서의 Markov 체인 길이: L_k
  각 온도에서의 평형상태에 도달하기 위한 상태변환의 회수로 연속해서 태스크 수만큼 비용의 변화가 허용치 ε내에 존재할 때의 상태변화 수로 설정한다.
- 온도 감소율: $T_{k+1} = {}_aT_k$
  온도 감소에 쓰이는 상수 a는 실험적으로 구해진 값으로 0.9로 정하였다.

3.2 유전자 알고리즘

유전자 알고리즘(genetic algorithm, GA)[5,6]은 자연계의 적자생존의 원리에 기초하여 생물학적 진화 시스템을 시뮬레이션 한 것으로 함수 최적화, 순회판매원문제(traveling salesman problem, TSP), 태스크 할당, 이미지 프로세싱 등 중요한 최적화 문제에 대한 근사 최적해를 구하는데 성공적으로 사용되어왔다.

그림 5는 멀티프로세서 매핑 문제에 대한 부하균형을 고려한 GA 솔루션을 표현한 것으로 유전자 알고리즘

[1] t ← 0
[2] 개체집단 P(t) 를 초기화한다.
[3] P(t) 를 평가한다.
[4] while (not 종료조건) do {
[5] t ← t + 1
[6] P(t-1) 에서 P(t) 를 선택(selection)한다.
[7] 생식, 교차, 돌연변이 연산자를 통해 P(t) 를 변경시킨다.
(구현을 위해 사용된 세팅을 아래에서 설명한다.)
[8] P(t) 를 평가한다.
[9] }

그림 5 유전자 알고리즘 구조

구현에 사용된 파라미터 세팅의 구체적인 내역은 다음과 같다.

- 개체 표현
  태스크의 순서대로 할당되는 노드번호의 순서를 문자열로 표현한다. 즉, 태스크 0부터 4까지 5개의 태스크를 프로세서 노드에 할당되어 있는 상태를 표현한 문자열 "1,5,4,1,5"은 태스크 0과 3는 노드 1, 태스크 1과 4는 노드 5, 노드 2는 노드 4에 할당되어 있는 상태를 표현한다.
- MFA에서 GA로의 개체 생성
  MFA에서 사용되는 N×K 스핀 행렬과 같은 확률로 랜덤하게 개체를 생성한다. 예로서, 임의의 태스크가 노드 0부터 4까지 할당되는 스핀 값이 0.2, 0.4, 0.1, 0.1, 0.2라면 주어진 태스크가 노드 0에 할당될 확률은

0.2, 노드 1에는 0.4, 노드 2와 3에는 0.1, 노드 4에는 0.2의 할당 확률을 가지고 개체를 생성하게 된다.

- 개체집단(population)의 크기
  개체(individual) 수로 본 실험에서는 100으로 설정하였다.
- 비용함수
  MFA와 같은 1차식 비용함수를 사용하였다.
- 선택(selection) 연산자
  집단의 전체 비용에 대한 개체의 비용 비율만큼 랜덤 함수를 사용하여 개체를 선택한다.
- 생식(reproduction) 연산자
  선택 연산자에 의하여 선택된 개체를 가지고 집단의 크기만큼의 새로운 집단을 생성한다.
- 교차(crossover) 연산자
  개체의 순서대로 2개의 개체를 선택하여 유전자 즉, 태스크의 노드 할당 상태를 나타내는 문자열의 부분을 서로 교환한다. 교환될 태스크의 선택은 전체 태스크수의 1/4이하로 제한 하였으며, 선택은 랜덤하게 이루어진다. 교차 연산자를 수행할 확률은 0.8로 설정하였다.
- 돌연변이(mutation) 연산자
  임의의 개체를 확률 0.05 만큼 선정하여 교환 및 이동연산을 수행한다. 교환 확률은 0.1, 이동확률은 0.9로 설정하였다. 교환 연산자는 한 개체에서 두 태스크를 임의로 설정하여 두 태스크의 할당된 노드를 서로 교환한다. 이동연산자는 2개 이하의 태스크를 임의로 선정하여 태스크에 할당된 노드를 무작위하게 변화시킨다.
- 최적 유전자 보전
  집단에서 최적의 비용을 가진 개체는 항상 집단에 남아 있도록 보전한다.
- 연산자 수행 회수
  GA 종료조건에서 각 온도에서 수행되는 유전연산자 수행 회수는 20번으로 설정하였다.
- GA 개체집단으로부터 MFA 스핀 행렬 생성
  GA 집단에 대해 생식 연산자를 수행하여 새로이 집단을 만든 후 집단에서의 태스크의 노드 할당 비율로 스핀 행렬을 생성한다. 예로서 개체 10개에서 태스크 0 의 노드 할당이 0,1,0,1,0,1,0,1,0,0이면 태스크 0의 스핀은 *0.6, 0.4, 0.0, 0.0,…*로 설정된다.
- SGA(Genetic Algorithm with Simulated Annealing)
  주어진 온도에서 MFA에서의 평형 상태를 GA 수행시에도 유지하기 위해서 GA 연산자 적용시 발생하는 비용 변화를 SA에서 사용하는 Metropolis Criterion에 따라서 상태 변화를 수행한다. 주어진 Metropolis Criterion은 다음과 같다.

$$\Pr[\Delta C \text{ is accepted}] = \min\left(1, \exp\left(\frac{\Delta C}{T}\right)\right)$$

여기서 ΔC는 연산자 수행시 발생하는 비용변화를 나타내며, T는 현재 온도를 가리킨다.

3.3 MGA 합성 알고리즘

분산메모리 멀티프로세서 병렬시스템에서 부하균형을 고려한 효율적인 매핑을 위한 평균장 어닐링(MFA)과 유전자 알고리즘(GA)의 장점을 결합한 합성 알고리즘(mean genetic algorithm, MGA)을 기술한다. MGA에서는 개선된 평균장 어닐링 알고리즘을 적용하였다. 즉, MFA에서는 각 온도에서 열평형 상태에 도달한 후 각 부하의 각 프로세서에 대한 할당이 확률로 표현되므로, 그 확률에 따라 유전자 알고리즘을 적용하기 위한 개체 집단을 만들 수 있다. 따라서 각 온도에서 열평형 상태 도달 후 만들어진 집단에 대하여 유전자 연산을 수행한 후 집단 내의 유전자 비율에 따라 다음 온도의 어닐링을 위한 상태를 결정한다. 이와 같이 평균장 어닐링과 유전자 알고리즘을 충분히 낮은 온도에서 시스템이 동결(freeze)될 때까지 위의 연산을 반복 수행한다. 그림 6은 이러한 과정의 윤곽을 의사 코드로 기술한 것이다.

```
[1] 변수 및 문제, 노드 구조 초기화한다.
[2] 스핀 행렬을 생성 한다.
[3] 부하대 통신비 계수 r 설정한다.
[4] 초기온도 T0를 설정하여 T= T0 세팅한다.
[5] while T ≥ 최종 온도 Tf do {
[6]       MFA 수행한다.
[7]       스핀 행렬로부터 GA 집단 생성 한다.
[8]       SGA 수행; /* GA 연산자 수행시 발생하는 비용함수 변화를
[9]                    SA의 Metropolis 임계식을 이용하여 상태변화를
[10]                   수행한다. */
[11]      GA 개체집단으로부터 MFA 스핀 행렬을 생성한다.
[12]      부하대 통신계수 r 조정한다.
[13]      T= aT 로 온도를 감소시킨다.
[14]}
```

그림 6 MGA 합성 알고리즘의 구조

4. 시뮬레이션 결과

본 장에서는 3장에서 설명한 하이브리드 합성 알고리즘(MGA)을 같은 비용함수를 사용하는 MFA 및 MFA와 같은 1차 비용함수를 사용한 유전자 알고리즘인 GA-1과 비교한다. 시뮬레이션은 C 프로그램으로 작성하였으며 여러 대의 PC상에서 실행하였다. 또한 GA-2는 각 노드의 부하의 제곱을 최소화하는 최소제곱 부하균형 기법을 비용함수로 사용하는 유전자 알고리즘을 나타낸다. 제안된 MGA 알고리즘의 실험 결과를 상대적으로 우수한 성능을 나타내는 GA-2의 결과와 비교함으로써 객관적인 평가를 수행한다.

병렬처리시스템에서 대표적인 연결구조로 하이퍼큐브(hypercube)와 메쉬(mesh) 구조를 생각할 수 있는데 하이퍼큐브는 노드가 많을수록 Diameter가 높아져 통신비용에 과부하를 초래하는 단점이 있는 반면 메쉬는 계산 비용이 상수로써 3D 메쉬로 확장성(scalability) 측면

의 장점이 있다. 실제 예를 들어, 분산메모리 병렬 컴퓨터로 한 때 인기 있었던 Intel 하이퍼큐브 iPSC/2, /860 시스템들은 이미 벌써 단종되었고 이후에 메쉬 구조를 갖는 인텔 Paragon으로 대체되었다. 또한 제안된 비용함수는 병렬처리시스템에 독립적인 일반식으로 특정 연결망구조 및 문제 특성에 영향을 받지 않는다. 실험에 사용된 노드 및 간선의 개수를 비롯한 실험 문제에 대한 선정은 Bultan과 Aykanat 실험모델[1]을 참고하여 알고리즘 비교 목적으로 대상 문제 크기를 결정하였다.

이러한 근거에 기초하여 우리 실험 환경에서는 태스크 크기는 200과 400에 대해서, 멀티프로세서 노드는 메쉬 구조로 연결되었다고 가정하였으며, 각 태스크의 크기는 [1..10]의 값 중 균등분포로 설정하였고, 각 태스크 사이의 통신 크기는 [1..5]의 값 중에서 역시 균등분포로 선택하였다. 통신의 개수는 일정하게 정하였으며 본 실험에서는 태스크 수의 1, 2, 3배를 통신의 크기로 각각 정하였다. 각 실험에서 같은 문제에 대해서 10회씩 2번 수행하여 총 20회를 수행한 결과의 평균값이다.

4.1 통신대 노드 부하의 비 : r

사용되는 1차식 비용함수에서 계수 r은 노드 부하와 노드사이의 통신 부하간의 균형을 이루기 위하여 사용된다. 초기에 난수 발생기를 이용하여 각 태스크의 노드 할당이 일항 분포를 이루게 하며 이 분포를 스핀(spin) 값을 통해서 나타낸다. 초기 분포가 이루어진 후 계수 r은 다음 식을 이용하여 구해진다.

$$r = \frac{\text{통신부하}}{\text{노드수} \times \text{노드부하}} = \frac{\sum_{i=1}^{N}\sum_{j\neq i}^{N}\sum_{ip=1}^{K}\sum_{q\neq p}^{K} e_{ij}s_{ip}s_{jp}d_{pq}}{K\times\sum_{i=1}^{N}\sum_{j\neq i}^{N}\sum_{ip=1}^{K} s_{ip}s_{jp}w_i w_j}$$

노드부하와 통신부하의 비를 조정하는 계수 r은 MFA 과정의 각 온도에서 변화된 통신부하를 반영하기 위하여 각 온도에서 다음 식에 따라 변화된다. r_{old}는 전 온도에서 사용된 비율을 말하며 r_{new}는 현재 온도에서 새로이 계산된 계수 r을 가리킨다.

$$r_{new} = \begin{cases} 0.9\times r_{old} & \text{if } r_{new} < r_{old} \\ r_{old} & Otherwise \end{cases}$$

표 1은 초기의 계수 r을 고정시켰을 때와 r값을 각 온도에서 변화시켰을 때의 각 노드에 할당된 부하의 종료 시간 중 최대종료시간을 표로 나타내었다. 표 1에서 N은 부하의 수, $|E|$는 총 통신 개수, K는 노드의 수를 각각 나타낸다. 표 1에서 알 수 있듯이 r값을 각 온도에서 변화시키는 것이 훨씬 좋은 결과를 나타낸다.

표 2는 각 알고리즘에서의 최대종료시간을 나타낸다. 기존의 MFA 및 1차 비용식을 이용한 GA-1알고리즘은 최소 제곱 비용함수를 이용한 GA-2 알고리즘보다 최대종료시간이 더 길게 나타난 반면 새로이 제안한 MGA는 평균 9%의 성능 향상을 가져 왔다. 그러나 MGA의 성능향상이 통신의 개수가 적을 때 크게 나타난 반면 통신의 개수가 증가함에 따라 성능향상비가 감소한다. 이는 표 1의 MFA의 성능향상비에서 알 수 있듯이 노드 부하대 통신비 계수 r의 선택이 적절하게 이루어지지 않은 것에 기인하는 것으로 추정된다. 따라서 계수 r의 변화를 적절하게 수행할 수 있는 알고리즘의 개발이 요구되어 진다. 또한 문제의 크기가 커질수록 성능향상이 증가하는 것은 더 큰 문제에 MGA를 적용할 수 있는 가능성을 제시한다.

표 3은 총 통신비용을 나타낸다. GA-2에 비하여 MFA 및 GA-1은 통신비용이 크게 증가한 반면 MGA는 평균 9%의 감소를 보여준다. 노드 사이의 부하 불균형 비율 즉, 노드의 (최대종료시간-최소종료시간)/최대

표 1 부하와 통신비 계수 r의 변화에 따른 최대종료시간

문제 크기			초기 r 유지		계수 r 변화		성능 향상	
N	|E|	K	MFA	MGA	MFA	MGA	MFA	MGA
200	200	16	336.7	134.7	138.2	120.2	59%	11%
	400	16	707.5	324.2	525.9	320.1	26%	1%
	600	16	845.0	526.8	767.2	544.3	9%	-3%
	200	36	193.4	90.3	84.7	72.0	56%	20%
	400	36	430.5	235.8	307.3	218.5	29%	7%
	600	36	651.6	420.0	559.2	388.2	14%	8%
400	400	16	627.1	256.4	279.1	222.8	56%	13%
	800	16	1190.0	617.2	888.1	587.0	25%	5%
	1200	16	1862.8	971.9	1557.4	987.5	16%	-2%
	400	36	410.5	143.9	152.9	128.7	63%	11%
	800	36	834.5	410.9	617.0	385.2	26%	6%
	1200	36	1376.8	714.7	1065.5	693.0	23%	3%
						평균	33%	7%

표 2 각 알고리즘의 최대종료시간 및 GA-2를 기준으로 한 성능향상 비교

문제 크기			최대종료시간				성능 향상			
N	|E|	K	MFA	GA-1	GA-2	MGA	MFA	GA-1	GA-2	MGA
200	200	16	138.2	184.6	147.7	120.2	6%	-25%	0%	19%
	400	16	525.9	389.4	326.7	320.1	-61%	-19%	0%	2%
	600	16	767.2	579.4	544.7	544.3	-41%	-6%	0%	0%
	200	36	84.7	136.6	90.2	72.0	6%	-52%	0%	20%
	400	36	307.3	281.9	222.2	218.5	-38%	-27%	0%	2%
	600	36	559.2	454.9	391.0	388.2	-43%	-16%	0%	1%
400	400	16	279.1	364.7	284.4	222.8	2%	-28%	0%	22%
	800	16	888.1	709.1	618.8	587.0	-44%	-15%	0%	5%
	1200	16	1557.4	1075.2	1041.1	987.5	-50%	-3%	0%	5%
	400	36	152.9	244.9	169.6	128.7	10%	-44%	0%	24%
	800	36	617.0	530.7	415.0	385.2	-49%	-28%	0%	7%
	1200	36	1065.5	850.5	732.8	693.0	-45%	-16%	0%	5%
						평균	-29%	-23%	0%	9%

표 3 각 알고리즘의 총 통신비용 비교

문제 크기			총 통신 시간				성능 향상			
N	|E|	K	MFA	GA-1	GA-2	MGA	MFA	GA-1	GA-2	MGA
200	200	16	414.6	627.55	523.2	346.8	21%	-20%	0%	34%
	400	16	3650.5	3260.9	1907.2	2084	-91%	-71%	0%	-9%
	600	16	6525.6	5744	3482.1	4215.8	-87%	-65%	0%	-21%
	200	36	539	1718.1	858.2	490.5	37%	-100%	0%	43%
	400	36	4002.2	4903.4	3096.7	2956.3	-29%	-58%	0%	5%
	600	36	8608.2	8546.1	5513	6231.2	-56%	-55%	0%	-13%
400	400	16	994.3	2370.5	1093	629.7	-%	-117%	0%	42%
	800	16	8089.1	6714.6	3918.2	4004.2	-106%	-71%	0%	-2%
	1200	16	14677	11743	7017.9	8348.4	-109%	-67%	0%	-19%
	400	36	1062.7	3539.5	1714	852.4	38%	-107%	0%	50%
	800	36	10021	10360	6222.3	5603	-61%	-66%	0%	10%
	1200	36	19937	17780	11008	11868	-81%	-62%	0%	-8%
						평균	-43%	-72%	0%	9%

표 4 각 알고리즘의 노드 부하 불균형 비율 비교

문제 크기			평균 퍼센트 부하 불균형				성능 향상			
N	|E|	K	MFA	GA-1	GA-2	MGA	MFA	GA-1	GA-2	MGA
200	200	16	47%	42%	44%	39%	-9%	3%	0%	11%
	400	16	68%	49%	70%	61%	2%	30%	0%	13%
	600	16	65%	45%	79%	60%	18%	43%	0%	25%
	200	36	66%	71%	59%	58%	-13%	-21%	0%	1%
	400	36	85%	73%	81%	77%	-5%	10%	0%	5%
	600	36	85%	68%	89%	78%	5%	24%	0%	13%
400	400	16	47%	37%	37%	32%	-27%	0%	0%	15%
	800	16	61%	37%	63%	57%	4%	42%	0%	9%
	1200	16	49%	35%	75%	54%	34%	53%	0%	28%
	400	36	60%	59%	51%	50%	-16%	-16%	0%	2%
	800	36	79%	62%	75%	72%	-6%	17%	0%	4%
	1200	36	75%	60%	84%	70%	11%	28%	0%	17%
						평균	0%	18%	0%	12%

종료시간*100으로 각 노드에 부하가 편향되게 분포되어 있는 정도를 백분율로 표시하였다. 일련의 표 2, 4 ,6의 실험결과를 통하여 MGA 알고리즘으로 얻은 솔루션은 MFA, GA-1, GA-2로 얻은 것보다 종료시간 및 통신 부하 측면에서 좋은 할당을 보여줌을 확인할 수 있다.

표 4에서는 노드 사이의 부하 불균형 비율 즉, 노드의

표 5 각 알고리즘의 수행시간 비교

문제 크기			수행 시간 (단위: 초)				성능 향상			
N	|E|	K	MFA	GA-1	GA-2	MGA	MFA	GA-1	GA-2	MGA
200	200	16	6.2	16.4	19.4	22.4	68%	16%	0%	-15%
	400	16	8.5	19.4	22.2	33.2	62%	13%	0%	-50%
	600	16	11.1	19.8	25.8	42.5	57%	23%	0%	-65%
	200	36	22.1	19.4	30.9	46.9	28%	37%	0%	-52%
	400	36	27.6	22.4	36.7	59.4	25%	39%	0%	-62%
	600	36	33.4	24.5	43.3	74.9	23%	43%	0%	-73%
400	400	16	26.2	61.7	87.4	95.6	70%	29%	0%	-9%
	800	16	40.7	70.3	109.9	150.9	63%	36%	0%	-37%
	1200	16	51.7	63.7	119.1	190.9	57%	47%	0%	-60%
	400	36	94.9	77.6	145.4	212.4	35%	47%	0%	-46%
	800	36	122.9	76.4	138.9	260.6	12%	45%	0%	-88%
	1200	36	148.1	70.6	154.5	317.5	4%	54%	0%	-106%
						평균	42%	36%	0%	-55%

(최대종료시간-최소종료시간)/최대종료시간*100으로 각 노드에 부하가 편향되게 분포되어 있는 정도를 백분율로 표시하였다. 여기서의 실험결과는 MGA 알고리즘으로 얻은 솔루션은 MFA, GA-1, GA-2로 얻은 것보다 종료시간 및 통신 부하 측면에서 좋은 할당을 보여준다.

표 5는 각 알고리즘 수행시간을 초 단위로 표시한 것이다. 최대종료시간과 통신비용을 부하 불균형 비율을 함께 고려하여 솔루션 품질을 결정짓는다. MFA, GA-1, GA-2, MGA 알고리즘들의 평균 성능 측정치를 요약하여 비교한 것을 표 6에서 나타냈다. 표 5, 6에서 보여주듯이 표시된 수치는 GA-2 알고리즘을 기준으로 한 성능향상 퍼센트(%)를 나타낸 것이다. MFA 알고리즘이 가장 빠르고 그 다음으로 GA-1, GA-2 순으로 수행시간이 빠르다. 반면에 MGA는 상대적으로 더 오랜 시간이 걸렸다. 이 결과는 좋은 품질을 얻기 위해서는 더 많은 시간이 소요된다는 것을 보여준다.

표 6은 각 알고리즘의 수행시간과 태스크 종료시간의 trade-off 관계를 표시한 것으로 MGA 알고리즘의 비교적 긴 수행시간은 전체적으로 짧은 태스크 종료시간으로 보상될 수 있다. 본 실험에서 MGA가 GA-2에 비교하여 최대종료시간은 평균 9%의 향상이 있었으나 알고리즘 수행시간은 55%가 더 걸렸다. 그러나 주어진 문제에서 사용횟수가 여섯일 경우 6×9(최대종료시간 향상비)=54% 이므로 알고리즘 수행시간 55%에 가깝다. 따라서 사용빈도가 높아질수록 MGA가 비교대상 GA-2 알고리즘보다도 우수한 성능을 보이게 된다. 단 실험에서는 부하균형에 사용되는 시간과 실제 태스크가 수행되는 시간을 별도로 비교하지 않았고 두 시간 개념을 분리하지 않고 같다는 전제하에 비교했다. 그러므로 앞서 보여준 표 2-5 결과와 함께 태스크 할당문제의 크기가 커질수록 사용빈도가 많을수록 알고리즘 수행시간은 오래 걸리더라도 보다 좋은 할당 솔루션을 얻을 수 있고 부하균형에 소요되는 시간은 태스크 수행시간의 단축으로 보상될 수 있다.

표 6 각 알고리즘의 평균 성능 측정치(%) 비교

	MFA	GA-1	GA-2	MGA
최대종료시간	-29	-23	0	9
부하 불균형	0	18	0	12
통신비용	-43	-72	0	9
수행시간	42	36	0	-55

5. 결 론

본 논문에서는 병렬 분산메모리 멀티프로세서 시스템에서 부하균형을 고려한 새로운 매핑 알고리즘(MGA) 기법을 제안했다. 제안된 MGA는 MFA와 GA의 장점을 결합한 합성 알고리즘으로 다른 알고리즘(MFA, GA-1, GA-2)들에 비해 솔루션 품질이 상대적으로 더 좋다. 즉, 태스크의 최대종료시간 및 통신 부하 측면에서 비교한 결과 혼합형 휴리스틱 기법을 이용하면 다른 알고리즘에 비하여 수행시간이 더 걸리지만 기존의 방법들 보다 개선된 실험결과를 얻을 수 있었다. 수행시간 측면에서는 MFA, GA-1, GA-2, MGA 순으로 차이를 보였으나 할당 솔루션 품질 면에서 MGA가 가장 좋은 효과를 보였다. 이러한 결과는 좋은 품질을 얻기 위해서는 더 많은 시간이 소요된다는 것을 보여준다. 즉, MGA 알고리즘의 긴 수행시간은 짧은 태스크 종료시간으로 보상될 수 있다. 합성 알고리즘 MGA를 이용하면 GA-2에 비하여 알고리즘 수행시간은 더 걸린다 할지라도 최대종료시간의 향상을 보이며 문제에 대한 사용횟수가 증가할수록 MGA는 다른 알고리즘보다도 우수한 성능을 얻을 수 있다. 결론적으로 요약하면 할당 알고리

즘 수행시간과 태스크 종료시간은 trade-off 관계로 태스크 할당문제의 크기가 커질수록 사용빈도가 많을수록 알고리즘 수행시간은 오래 걸리더라도 보다 좋은 할당 솔루션을 얻을 수 있음을 다양한 비교 실험연구를 통해 확인했다.

참 고 문 헌

[1] Bultan, T. and C. Aykanat, "A New Mapping Heuristic Based on mean-field Annealing," *Journal of Parallel & Distributed Computing*, 16, pp.292-305, 1992.

[2] Heiss, H.-U. and M. Dormanns, "Mapping Tasks to Processors with the Aid of Kohonen Network," *Proc. High Performance Computing Conference*, Singapore, pp. 133-143, Sep. 29-30, 1994.

[3] 박경모, 홍철의, "Performance of Heuristic Task Allocation Algorithms," *CUK Journal of Natural Science*, Vol. 18, pp. 145-155, December 1998.

[4] Salleh, S. and A. Y. Zomaya, "Multiprocessor Scheduling Using Mean-Field Annealing," *Proc. of the First Workshop on Biologically Inspired Solutions to Parallel Processing Problems (BioSP3)*, pp. 288-296, March 30, Orlando, Florida, 1998.

[5] Zomaya, A.Y. and Teh Y.-W., "Observations on Using Genetic Algorithms for Dynamic Load-Balancing," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 12, No. 9 , pp. 899-911, Sept. 2001.

[6] 박경모, "멀티프로세서 태스크 할당을 위한 유전자 알고리즘과 시뮬레이티드 어닐링 비교", 한국정보처리학회 논문지, 제6권, 9호, pp. 2311-2319, 1999년 9월.

[7] 홍철의, "분산 시뮬레이티드 어닐링을 이용한 복합 재료 재단", 한국정보과학회 논문지 B, 제29권 1호, pp. 20-29, 2000년 2월.

[8] 홍철의, 박경모, "병렬처리에서 평균장 어닐링과 유전자 알고리즘을 이용한 부하균형", 한국정보과학회 2003 가을학술발표논문집(I), 제30권 2호, pp. 364-366, 2003년 10월 24-25일.

홍 철 의

1985년 한양대학교 금속공학과 졸업(학사). 1989년 미국 New Jersey Institute of Technology 전산학(석사). 1992년 미국 Univ. of Missouri-Rolla 전산학(박사). 1992년~1997년 한국전자통신연구원 선임연구원. 1997년~현재 상명대학교 정보통신학부 조교수. 관심분야는 분산 및 병렬 알고리즘, 최적화 기법

박 경 모

1980년 중앙대학교 전산학과 졸업(학사) 1983년 서울대학교 계산통계학과 전산과학(CS) 전공(석사). 1990년 미국 New Jersey Institute of Technology 컴퓨터학(석사). 1994년 미국 George Mason University 정보기술(IT)/컴퓨터(박사) 1983년~1985년 삼성전자 컴퓨터연구개발실 근무. 1990년~1995년 GMU CS Dept. IT&E Lab. 연구원. 1995년~1996년 한국전자통신연구원 시스템소프트웨어 연구원. 1996년~현재 가톨릭대학교 컴퓨터정보공학부 교수 관심분야는 컴퓨터시스템 성능평가, 분산/병렬처리, 정보보안 등