하이퍼큐브 다중컴퓨터에서 반복 타스크 분할에 의한 통신 비용 최소화

김 주 만† · 윤 석 한† · 이 철 훈††

요 약

본 논문에서는 병렬 프로그램을 구성하는 2^n개의 타스크 모듈들을 n-차원 하이퍼큐브 다중 컴퓨터에 전체 통신 비용이 최소가 되도록 일대일 매핑하는 문제를 다룬다. 하이퍼큐브에서 최적 매핑을 구하는 것은 NP-complete문제이다. 본 논문에서는 먼저 하이퍼큐브 다중 컴퓨터에서의 매핑 문제를 그래프 상에서의 최대 컷세트 집합을 구하는 문제로 변환시키는 그래프 변형 기법을 제안한다. 이러한 그래프 변형 기법을 사용하여 기존의 그래프 이분할 방법을 변형된 그래프 상에 반복 적용함으로써 하이퍼큐브에 타스크 모듈들을 효율적으로 일대일 매핑하는 반복 매핑 알고리즘을 제안한다. 여러가지 타스크 그래프 상에서의 실험을 통해, 제안된 반복 매핑 알고리즘이 기존의 *greedy*나 *recursive* 매핑 알고리즘들 보다 성능이 우수함을 보인다. 특히 제안된 알고리즘은 하이퍼큐브-isomorphic, 메쉬등과 같은 정형 그래프 상에서 성능이 우수하며, 거의 모든 정형 그래프에서 최적 매핑을 찾음을 보인다.

Minimization of Communication Cost using Repeated Task Partition for Hypercube Multiprocessors

Joo-Man Kim† · Suk-Han Yoon† · Cheol-Hoon Lee††

ABSTRACT

This paper deals with the problem of one-to-one mapping of 2^n task modules of a parallel program to an n-dimensional hypercube multicomputer so as to minimize to total communication cost during the execution of the task. The problem of finding an optimal mapping has been proven to be NP-complete. We first propose a graph modification technique which transfers the mapping problem in a hypercube multicomputer into the problem of finding a set of maximum cutsets on a given task graph. Using the graph modification technique, we then propose a repeated mapping scheme which efficiently finds a one-to-one mapping of task modules to a hypercube multicomputer by repeatedly applying an existing bipartitioning algorithm on the modified graph. The repeated mapping scheme is shown to be highly effective on a number of test task graphs, it increasingly outperforms the greedy and recursive mapping algorithms as the number of processors increase. The proposed algorithm is shown to be very effective for regular graphs, such as *hypercube-isomorphic or 'almost' isomorphic* graphs and meshes; it finds optimal mappings on almost all the regular task graphs considered.

† 정 회 원 : ETRI 컴퓨터 · 소프트웨어 기술연구소 책임연구원
†† 정 회 원 : 충남대학교 컴퓨터공학과 교수
논문접수 : 1997년 11월 27일, 심사완료 : 1998년 9월 14일

1. 서 론

VLSI 및 컴퓨터 네드워킹 기술의 발달로 인하여 여러 가지 다중 컴퓨터 개발이 실용화 되고 있다. 하이퍼큐브는 그들 중 하나로서 구조가 간단하고 고도의 병렬성을 내재하고 있으며, 다른 구조로의 매핑이 우수하기 때문에 많은 연구 대상이 되어 왔다. 이진 n-큐브 처럼 알려진 n-차원 하이퍼큐브는 2^n 프로세서(노드)를 갖는 고도의 동시성 약결합 다중 컴퓨터(highly concurrent loosely-coupled multicomputer)이다.

하이퍼큐브 설계, 성능 평가등 여러가지 연구들이 지금까지 많이 진행 되어 왔다.[1]-[3] 하이퍼큐브 토폴로지를 기반한 다중 컴퓨터 시스템으로 Intel iPSC [4], NCUBE[5], Caltech/JPL[6], 그리고 the Connection machine[7]등이 연구 개발 되었다. 하이퍼큐브 다중 컴퓨터에 병렬 프로그램을 수행하기 위하여 타스크 모듈들은 하이퍼큐브 각 프로세서(노드)중 한 프로세서에 할당 되어야 할 것이다. 노드 프로세서에 타스크 모듈을 할당하는 문제는 매핑 문제(*mapping problem*)라고 부른다.[8]

많은 병렬 프로그램은 처리 능력보다 통신에 의하여 많은 제한을 받는다. 즉, 계산 제한적이기 보다 통신 제한적이다. 일부 fine-grain 병렬 프로그램은 한 메시지에 대한 반응으로 10개 정도의 적은 명령어들을 수행한다.[9] 또한 타스크 수준에서도 계산용 타스크는 여러 개의 통신 모듈로 구성되어 있으며, 이들 모듈이 다른 프로세서에 할당 되었을 때 프로세서간 통신량은 과도하게 나타날 수 있다. 이러한 프로그램을 효율적으로 수행하기 위하여 통신 네트웍은 시스템 전반적으로 동시 다발적인 과도한 통신량을 다룰 수 있어야 한다. 본 논문에서는 하이퍼큐브 다중 컴퓨터에서 프로세서간 통신 부하가 최소화 되도록 병렬 프로그램을 구성하는 통신 타스크 모듈의 집합을 매핑하는 문제에 대해 고찰한다. 하지만 이 문제는 NP-complete 문제임이 이미 증명 되어 있다.[10] 따라서, 최적 매핑 대신에 최적에 가까운 매핑을 빨리 찾을 수 있는 휴어리스틱 방법을 사용 하여야 한다. 하이퍼큐브의 매핑 문제를 다루는 greedy 알고리즘은 [11] 하이퍼큐브에 통신 그래프를 매핑하기 위해 그래프 지향적 방법(graph-oriented strategy)을 이용하고 있다. 이 greedy 알고리즘은 선형 배열, 하이퍼큐브-isomorphic, 그리고 isomorphic에 근사한 통신 그래프 등과 같은 몇몇 그래프 형태에서 아주 잘 수행되지만 그 외 다른 그래프 형태에서는 그렇지 못하다. 또 Ercal 등은 Kernighan-Lin이 제안한 그래프 분할 방법[13]을 기반으로 타스크 그래프를 반복하여 재귀적인 이분할을 수행하는 재귀적 반복 이분법(recursive divide-and-conquer)을 제안하였다.[12] 본 논문에서는, 참고문헌 [14]의 그래프 이분할 휴어리스틱(graph bipartitioning heuristic)을 기반으로 하이퍼큐브 다중 컴퓨터에 병렬 프로그램의 타스크 모듈을 효과적으로 매핑하기 위한 반복 매핑 알고리즘(Repeated Mapping Algorithm)을 제안한다. 이러한 반복 매핑 방법은 그래프 변형과 이분할을 반복적으로 수행하는 것으로서, 부 그래프 상에서 재귀적으로 이분할하는 재귀적 매핑 방법[12]과는 다르다. 재귀적 매핑 방법의 단점은, 지역적인 관점에서 이분할 하기 때문에, 비록 각각의 부그래프에서 최적 이분할을 하더라도, 매핑 문제는 최적으로 풀리지 않는다는 것이다. 반면에, 반복 매핑 방법은 각각의 반복 구간에서 전체적인 관점에서 이분할을 하기 때문에, 각각의 이분할이 최적일 경우, 매핑 결과는 최적이 된다. 다시말해 매 반복 구간 마다 최적으로 이분할 하므로서 최소의 총 통신 비용 갖는 최적 매핑을 찾는다. 본 논문의 주요 공헌은 매핑의 결과가 일대일이 되도록 하는 그래프 변형 기법이다. 여러가지 시뮬에이션 결과에 의하여 본 논문에서 제안한 반복 매핑 방법이 greedy나 recursive 매핑 방법 보다 성능이 우수함을 보인다.

본 논문은 다음과 같이 구성되어 있다. 2 절에서는 매핑 문제를 서술하기 위하여 먼저 시스템 모델에 대한 분석적 정의와 가정을 도출하고 매핑 문제를 정식화한다. 3 절에서는 하이퍼큐브의 일부 간단한 토폴로지 특성을 도출하고, 이 구조에 효과적으로 적용할 수 있는 프레임웍으로서 컷세트 공식을 정형화하여 기술한다. 4 절에서 참고문헌 [14]의 휴리스틱을 기반한 효율적인 그래프 분할 알고리즘을 제안하고, 아울러 그래프 분할 알고리즘이 하이퍼큐브 다중 컴퓨터에서 매핑 문제를 성공적으로 적용할 수 있게 하는 그래프 변형 기법에 대하여 기술한다. 5 절에서 다양한 타스크 그래프를 사용, 각각의 실험 결과를 보이며, greedy 알고리즘과 recursive 매핑 알고리즘과 성능을 비교하여 평가하며, 마지막 6 절에서 결론을 맺는다.

2. 매핑 문제 서술

2.1 하이퍼큐브의 분석적 정의

n-차원 하이퍼큐브는 다음과 같이 구성된다. 우선, 2^n 노드/프로세서는 0에서 2^n-1 범위로 2^n 개의 이진수 번호가 부여되며, 이들 이진 주소에서 오직 한 비트만이 다른 이진 주소를 갖는 두 노드간에 직접적인 통신 링크가 존재한다. 그러므로 n-차원 하이퍼큐브 다중 컴퓨터는 자신의 기억장치를 갖는 2^n 개의 프로세서 $\{P_k \mid 0 \le k \le 2^n - 1\}$로 구성되며, 또한 각각의 프로세서는 n 개의 인접 이웃 프로세서와 연결 되므로 전체적으로 $n2^{n-1}$개의 양방향 통신 링크를 가진다. 이러한 하이퍼큐브의 특성을 바탕으로 매핑을 전제하여 다음과 같은 가정을 세운다.

[가정 1] 하이퍼큐브 각 노드의 프로세서는 동일한 성능을 가지며, 통신 링크도 동일 대역폭을 갖인다.

[가정 2] 대부분의 기존 하이퍼큐브 다중 컴퓨터 처럼 모듈 상호 통신은 메시지 교환을 통해 수행된다

각 메시지의 길이는 고정 길이의 패킷으로 나누어지며, 모듈들 사이의 통신양은 그들 사이에 수행중 교환된 패킷의 수로 표현한다.

하이퍼큐브 다중 컴퓨터의 각 프로세서(혹은 노드) P_k의 번지를 k의 이진 표현 $k^n k^{n-1} \ldots k^1$이라고 하자. 임의의 두 프로세서 P_k와 P_l 사이의 Hamming 거리를 $d_{k,l}$라고 하면, $d_{k,l}$은 P_k와 P_l 사이에 거쳐야 할 최소 통신 링크의 수이다. 즉, P_k와 P_l사이를 움직이는 통신 링크의 가장 작은수로서 $d_{k,l} = \sum_i k^i \oplus l^i$로 나타낼 수 있다.

[가정 3] 본 논문에서 고려중인 하이퍼큐브 다중 컴퓨터는 e-cube 라우팅을 사용한다.

e-cube 라우팅은 출발 번지와 목표 번지의 이진값을 LSB에서 MSB로 상이한 비트를 고정된 순서로 수정하면서 출발지와 목적지 사이에 패킷이 라우팅하도록 한다. 따라서, 각 패킷들은 출발 노드에서 목적 노드로 지정된 최단 경로를 따라 라우팅 된다.

[가정 4] 일반성 상실없이, n-차원 하이퍼큐브로 타스크 모듈의 집합을 매핑 할 때 타스크의 수는 2^n 개이다.

정수 n에 대해 타스크의 모듈 수가 M이 $2^{n-1} < M < 2^n$인 경우에는 모조(dummy) 모듈을 추가하여 전체 모듈 수를 2^n 개로 만들 수 있다. 또한 $M > 2^n$인 경우에는 타스크 분할 방법을 제안한 참고문헌 [14]에서 기술한 것 처럼 작업 부하 분할 스킴에 의하여 몇 개의 모듈을 그룹화하여 클러스터링할 수 있고, 그룹의 수는 모듈 사이의 전체 통신 구조를 계산하여 하이퍼큐브의 사이즈와 일치하게 한다.

이러한 가정에 대해 하이퍼큐브에서 작성된 대부분의 병렬 프로그램은 이 법칙에 따라 작성 되었으며, 존재하는 하이퍼큐브 다중 컴퓨터는 하나의 노드에 대해 다중 프로그래밍 환경을 지원하지 않기 때문에 이러한 가정이 비 현실적이지 않다.

2.2 모델의 정식화

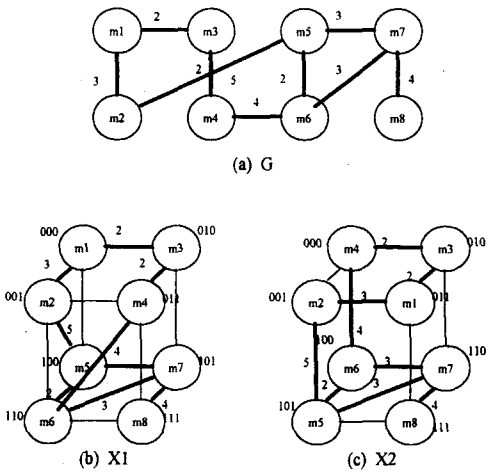
모듈과 프로세서간의 매핑은 일대일 함수 $X: m_i \rightarrow p_k$로 표현한다. 여기에서 $X(i)$는 모듈 m_i가 매핑된 프로세서의 주소를 나타낸다. 프로세서가 모듈 m_i를 수행하는데 소요되는 수행 비용을 $comp(m_i)$라고 하자. 하이퍼큐브의 각 프로세서는 동일한 성질을 가지기 때문에 매핑에 관계없이 전체 수행 비용은 일정하다. 즉, $\sum_{i=1}^{M} comp(m_i)$를 가질 것이다. 그러나 매핑에 따라서 전체 통신 비용은 달라질 것이다. 전체 통신 비용은 타스크를 수행하는 동안에 각 통신 링크가 사용된 시간 단위의 합으로 정의한다. 다시말해서, 통신 비용은 타스크 수행 시 인스턴스에 의해 사용된 통신 링크 자원들을 시간 단위로 측정한 값이다.

거리가 h인 경로를 따라 하나의 패킷을 전달하는데 소요되는 시간을 $c(h)$라고 하자. 여기에서 패킷 전달 이외의 목적으로 사용된 시간은(즉, 통신 경로 개설 등) 무시할 수 있다고 가정한다. 패킷 스위칭을 하는 하이퍼큐브에 대해서는 $c(h) = h * c(1)$을 가진다. 이 관계는 회선 스위칭(circuit switching)인 경우에는 이 관계식이 보다 부정확 할 수도 있으나, 자유 경로를 찾기위한 "call request" 신호가 각 링크를 아주 짧은 시간 동안에만 사용한다고 했을 경우에는 회선 스위칭을 하는 하이퍼큐브에서도 이 관계식이 좋은 근사치가 될 것이다.[15]

하나의 패킷이 하나의 링크를 통과하는데 걸리는 시간인 $c(1)$을 통신 비용의 기본 단위로 정의한다. 따라서 매핑 X 하에서 한 타스크를 수행하는데 소요되는 통신 비용은

$$COST(X) = \sum_{i<j} W_{i,j} \cdot d_{x(i),x(j)}$$

이다. 예를들어 그림 1에서 8개의 모듈로 구성된 예제 타스크 그래프와 이들이 3-큐브에 매핑된 두가지 다른 매핑 X_1과 X_2가 나타나 있다. X_1과 X_2의 전체 통신 비용은 각각 40과 31 이다. 매핑에 상관없이 전체 수행 비용은 일정하기 때문에 단지 프로세서간 통신 비용만이 타스크를 매핑하는데 고려할 필요가 있다. 이런점이 참고문헌 [17]-[19]에서 기술한 다중 컴퓨터 상에서의 기존 타스크 할당 연구들과 본 논문의 할당에 관한 작업이 구별되는 점이다.



(그림 1) 예제 타스크 그래프 G의 매핑 X1, X2
(Fig. 1) An example task graph G with mapping X1, X2

그러나 하이퍼큐브는 또한 대칭이기 때문에 임의의 두 매핑 X_a와 X_b는 임의의 n-비트 이진값 x에 대해 $X_a(i) \oplus x = X_b(i)$, (단,$1 \le i \le M$)이라면 동일한 통신 비용에 이르게 될 것이다. 이러한 두 매핑을 *equivalent* 하다고 한다. 본 논문에서 고려하는 매핑 문제는 타스크 모듈을 프로세서에 매핑하기 위하여 전체 통신 비용이 최소인 일대일 매핑 X_0를 구하는 문제로서 다음과 같은 목적 함수로 나타낸다.

$$COST(X_0) = \min_X COST(X) = \min_X \left(\sum_{i<j} W_{i,j} \cdot d_{x(i),x(j)}\right)$$

3. 컷세트 공식

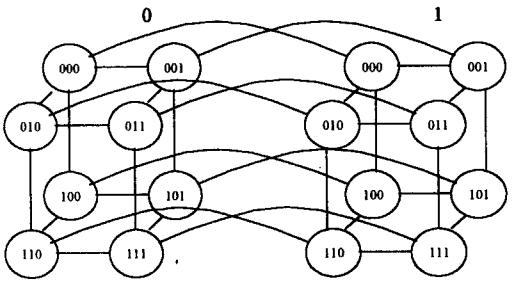
n-큐버는 그래프 Q_n으로 표현되며 이 그래프의 노드 집합 V_n은 0에서 부터 2^n-1 까지의 이진수 번지를 각각 가지는 2^n개의 노드로 구성되고, 그때 두 노드 사이의 링크는 각 노드에 부여된 이진 번호에서 1-비트가 달라질 때 그 두 노드 사이의 엣지 즉, 직접적인 연결이 존재한다.

정의 1 : n-큐브인 Q_n은 2^n개의 n-차원 대수 벡터 즉, 이진 코디네이터 0 또는 1을 갖는 벡터로 구성된 노드 집합 V_n의 무방향 그래프로서, 두 노드는 정확이 하나의 코디네이터가 다를 때 서로 이웃 한다.

n-큐브의 중요한 특징중의 하나는 차원이 낮은 큐브로 부터 재귀적으로 구성될 수 있다는 것이다. 다시 말해 버텍스가 0에서 $2^{n-1}-1$까지 차례로 번호가 부여되는 두개의 동일한 $(n-1)$-큐브를 고려해 보면, 첫번째 $(n-1)$-큐브의 모든 버텍스를 동일 번호를 갖는 두번째 버텍스에 결합 하므로서 n-큐브를 구할 수 있다. 실제로 이것은 첫번째 큐브의 각 노드를 $0x_{n-1}x_{n-2}\ldots x_1$로, 두번째 큐브에서는 $1x_{n-1}x_{n-2}\ldots x_1$로 번호를 다시 부여하여도 된다. ($x_{n-1}x_{n-2}\ldots x_1$는 $(n-1)$-큐브의 두 유사 노드를 나타내는 대수 벡터이다.) 이것은 그림 2에서 $n=4$로 설명된다. 4-큐브는 첫번째 3-큐브를 두번째 3-큐브와 일치하는 노드에 결합하므로서 구해진다. n-큐브의 정의를 사용하여 i번째 번지 비트가 0인 노드와 i번째 번지 비트가 1인 노드를 연결하는 에지를 제거하므로서 n-큐브를 2개의 $(n-1)$ 큐브로 분할 할 수 있다. 이것은 달리 말해 i번째 방향으로의 이분할이라고 부른다. n-큐브 각 노드의 번지는 n-비트를 갖기 때문에 n-큐브를 $(n-1)$-큐브로 분할하기 위하여 n개의 서로 다른 방법이 있을수 있다. 일반적으로 말해서, k개의 다른 방향을 따라 반복적으로 이분할 하므로서 (단, $1 \le k \le n$), n-큐브는 2^k개의 $(n-k)$-서브 큐브로 분할된다.

n-큐브는 n개의 다른 방향을 따라 반복적으로 분할하므로서 2^n 노드로 완전히 분할 된다. 임의의 두 노드는 단지 하나의 i 번째 bit가 다를때 i 번째 방향을 따라 분할 하므로서 나누어 질 것이다. 하이퍼큐브의 이러한 특성을 이용하여 우리는 한 모듈이 매핑될 프

로세서의 k 번째 비트를 결정하는 k 번째 이분할로 타스크 그래프를 2개의 서브 그래프로 반복적으로 이분할 한다. 2^n 개의 모듈을 가진 주어진 타스크 그래프를 n-차원 하이퍼큐브로 부분적 맵을 반복하여 전체 모듈을 맵할 수 있다. 즉, n 번째 이분할은 각 모듈이 매핑되는 프로세서의 완전한 번지를 결정할 것이다.



(그림 2) 4차원 하이퍼큐브 Q_4
(Fig. 2) 4-Dimensional Hypercube : Q_4

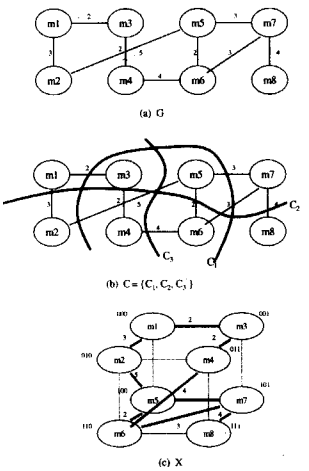
정의 2 : $|V|=2^n$을 갖는 타스크 그래프 $G(V, E)$의 컷세트 C_i는 $V_i \cap V_i' = 0$이고 $V_i \cap V_i' = V$인 V_i와 V_i'로 구분되는 엣지의 집합이다. 만일 $|V_i| = |V_i'| = |V|/2.(=2^{n-1})$이라면, C_i는 균등 컷세트라 부른다. C_i는 가중치 $W(C_i)$을 가지며 이것은 그 속에 있는 엣지값들의 합이다.

정의 3 : $|V|=2^n$을 갖는 타스크 그래프 $G(V, E)$에서 $(C_i \mid 1 \le i \le n)$인 C를 n 개의 균등 컷세트라 하자. 그때, $1 \le k \le n$이고, 집합 V가 정확히 2^{n-k} 타스크 모듈을 가지는 2^k 부분 집합으로 분할된다면, 집합 C에서 k개의 컷세트 $C_1, C_2, \ldots, C_k$를 가지므로, 그때 집합 C는 *admissible* 하다고 말한다. 집합 C의 가중치는 그 집합이 가지는 모든 컷세트들의 가중치의 합이다. 즉, $W(C) = \sum_i W(C_j)$이다.

매핑 X의 전체 통신 비용은 다음 수식으로 나타낼 수 있다.

$$\begin{aligned} COST(X) &= \sum_{a<b} W_{a,b} \cdot d_{x(a)} \cdot d_{x(b)} \\ &= \sum_{a<b} W_{a,b} \cdot \left(\sum_{i=1}^{n} X(a)^i \oplus X(b)^i\right) \\ &= \sum_{i=1}^{n} \left(\sum_{a<b} W_{a,b} \cdot \left(\sum_{i=1}^{n} (X(a)^i \oplus X(b)^i)\right)\right) \\ &= \sum_{i=1}^{n} W(C_i) = W(C) \end{aligned}$$

예를들어, 그림 3에서와 같이 2^3개의 노드로 구성된 타스크 그래프 G에서 집합 $C = \{C_1, C_2, C_3\}$는 *Admissible*하다. 여기에 대응하는 매핑 X의 전체 통신 비용은 62이고 이것은 집합 C의 가중치와 같다. 즉, $COST(X) = W(C_1) + W(C_2) + W(C_3) = 21 + 27 + 14 = 62$이다. 각 컷세트 C_i의 가중치 $W(C_i)$는 i번째 방향을 가진 통신 링크(즉, i번째 번지 비트가 0인 프로세서와 i번째 번지 비트가 1인 프로세서를 연결하는 링크)상에서의 통신 비용과 같다. 따라서, 위의 보조 정리에서 알 수 있듯이, 매핑 문제는 그래프 상에서 최소 가중치를 가지는 *Admmisible* 집합 C_o를 찾는 문제와 동일하다.



(그림 3) *Admissible* 집합 C와 대응하는 매핑 X
(Fig. 3) An admissible set C and its corresponding mapping X.

4. 반복 이분할 매핑 알고리즘

4.1 반복 분할 기법

참고문헌 [14]에서 제안된 분할 알고리즘은 주어진 그래프의 노드 갯수 N에 대해 분할을 수행 하였을 때, 시간 복잡도(*time complexity*)가 $O(N^2)$임을 밝히므로서 그래프 분할을 위한 효율적인 알고리즘 임을 증명하였다. 그 논문에서는 그래프 분할 문제에서 크기에 제한을 받을 경우 그래프 변환 기법을 사용하면 어떤 크기에도 제한을 받지 않고 기존 그래프를 *maxcut* 문제로 변환 된다는 것을 보여 주었다. 변환된 그래프에서 최대의 가중치를 가진 컷세트는 원 그래프에서 최

소의 균등 컷세트와 일치한다. 또한 그 논문에서는 *maxcut* 문제를 위한 효율적인 알고리즘을 제시하였다. 그들의 알고리즘은 Kernighan과 Lin의 알고리즘[13]과 다른 여러가지 제안된 알고리즘등과 비교 하였을 때 아주 성능이 우수함을 보여 주었다. 그러므로 그래프의 노드가 본 논문에서 고려하고 있는 타스크 그래프와 같이 모두 동일한 크기를 가질 때, 이것은 정확히 균등 분할을 보장한다. 여기서 사용된 기본적인 이분할 알고리즘은 참고문헌 [14]에 있는 알고리즘를 사용하였다.

본 논문에서는 $N=2^n$인 그래프의 균등 *N-way* 분할을 구하기 위해 먼저 분할의 각각을 균등한 두개의 분할로 분해하며, 다음 번 분할이 가능하도록 그래프를 변형한다. 이러한 반복적인 이분할 수행 단계동안 부분적으로 타스크 모듈을 프로세서에 매핑한다. 각 모듈에 대해 k 단계에서 매핑된 프로세서의 번지중 k 번째 비트가 결정되며, n 번째 단계의 이분할은 각 모듈이 매핑될 프로세서의 전체 번지를 결정할 것이다.

본 논문에서 다루는 이분할 알고리즘은 그래프 변환 기법(graph modification technique)의 효능에 의해 정확히 균등 분할을 보장하며, 더욱이 n 이분할의 결과는 *admissible*이다. 즉, 각 k-번째 단계에서 이분할 한 후, 정확히 2^{n-k}개의 타스크 모듈이 2^k개의 $(n-k)$ 큐브에 할당된다. 그러므로 마지막 n-번째 단계에서 이분할 한 후, 각 모듈은 매핑될 프로세서에 대한 유일한 번지에 할당된다. 즉, 매핑의 결과는 일대일 이다. 각 k-번째 단계에서의 이분할이 컷세트 C_k에 일치한다고 하자. 또한 타스크 그래프의 모듈의 집합이 컷세트 $C_1, C_2, \ldots C_k$에 의해 2^k개의 부집합 $P_k^1, P_k^2, \ldots, P_k^{2^k}$로 분할 된다고 하자. $P_{k-1}^i = P_k^{2i-1} \bigcup P_k^{2i}$= 즉, $k-1$개의 컷세트 $C_1, C_2, \ldots C_{k-1}$에 의해 2^{k-1}개의 부집합인 P_{k-1}^i의 각각은 다음번 컷세트 C_k에 의해 두 부집합 P_{k-1}^{2i-1}와 P_k^{2i}로 파티션 된다.

처음 어떠한 분할도 하기 전 최초의 타스크 모듈을 P_0^1 즉, $P_0^1 = V$이라고 할때, 정의에 의해서 집합 $C=\{C_1, C_2, \ldots, C_n\}$는, 만일 모든 $1 \le k \le n$에 대해 $|P_k^1| = |P_k^2| = \cdots = |P_k^{2^k}|$이면 *admissible*이다. 이분할 하기 전 매번 k 단계에서 타스크 그래프 $G(V, E)$는 컷세트가 다음처럼 *admissible*한 결과가 되도록 G_k^*로 변형된다. G의 임의의 2 노드 m_a와 m_b에 대해

1. 만일 두 노드가 $k-1$ 컷세트 $C_1, C_2, \ldots C_{k-1}$에 의해 분할되지 않는다면 (즉, 임의의 $i, 1 \le i, \le 2^{k-1}$에 대해 $m_a \in P_{k-1}^i$이고 $m_b \in P_{k-1}^i$), 그때 두 노드 사이의 엣지값을 $R - W_{a,b}$ 되게 수정한다. 여기에서 증가치 R은 $R > \sum_{a<b} W_{a,b}$인 임의의 양수 값이다. (원래 그래프 G에서 그들 사이의 엣지가 없다면, 엣지값 R을 가지는 새로운 엣지를 만든다.)
2. 그렇지 않으면, 그들 사이의 엣지값을 $-W_{a,b}$로 바꾼다.

보조 정리 1: 타스크 그래프 G와 그것의 k번째 변형된 그래프 G_k^*에 대해서, 본 알고리즘에서 구한 컷세트 $G_k^*(C_k)$가 $G_k^*(G)$의 노드들을 A_k와 B_k로 분할 한다고 하자.
그러면

$$W(C_k^*) = \sum_{i=1}^{2^{k-1}} |A_k \bigcap P_{k-1}^i| \cdot |B_k \bigcap P_{k-1}^i| \cdot R - W(C_k)$$이다.

증명: 각각의 2^{k-1}개의 부집합 P_{k-1}^i가, $1 \le i \le 2^{k-1}$, C_k에 의해 P_k^{2i-1}, P_k^{2i}로 분할되고, $P_k^{2i-1} = P_{k-1}^i \bigcap A_k$, $P_k^{2i} = P_{k-1}^i \bigcap B_k$이라고 하자. 그러면

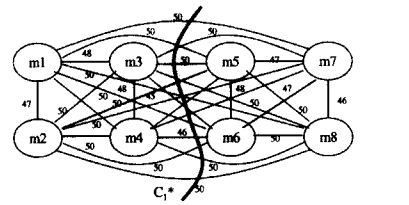
$$W(C_k^*) = \sum_{i=1}^{2^{k-1}} \sum_{m_c \in P_k^{2i-1}} \sum_{m_d \in P_k^{2i}} R - \sum_{m_c \in A_k} \sum_{m_d \in B_k} W_{c,d}$$

$$= \sum_{i=1}^{2^{k-1}} |P_k^{2i-1}| \cdot |P_k^{2i}| \cdot R - W(C_k).$$

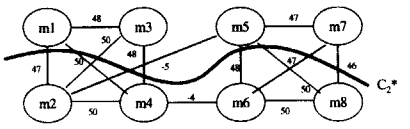
변형된 그래프 G_k^*의 각 컷세트 G_k^*는 분할된 부집합들의 크기에 대한 정보와 원래의 타스크 그래프 G 상에서의 대응하는 컷세트의 가중치에 대한 정보를 모두 가지고 있다. 만약 각각의 컷세트 G_k^*가 각기 해당하는 그래프 G_k^*상에서의 최대 컷세트인 경우, 모든 부집합 P_k^i는 크기가 동일하므로 (즉, $\sum_i |P_k^{2i-1}| \cdot |P_k^{2i}| = 2^{2n-k-1} =$ 상수), G상에서의 대응하는 컷세트 G_k는 최소 *admissible* 컷세트가 된다. 이것은 다음의 정리에서 증명이 된다.

정리 1 : $|V| = 2^n$인 주어진 타스크 그래프 $G(V, E)$에 대해서, 각각의 컷세트 G_k^*가, $1 \le k \le n$, 해당하는

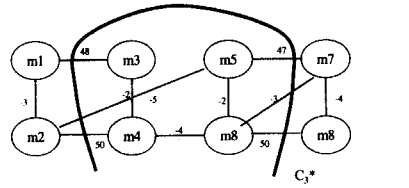
변형 그래프 G_k^* 상에서 최대 컷세트일 경우 집합 $C^* = \{C_1^*, C_2^*, \ldots C_n^*\}$는 *admissible*하다.



(a) C_1^* on G_1^* (W(C_1^*) = 791



(b) C_2^* on G_2^* (W(C_2^*) = 381



(c) C_3^* on G_3^* (W(C_3^*) = 187

(그림 4) 변형된 그래프에서의 admissible 집합 $C^*(R=50)$
(Fig. 4) An admissible set C^* on the modified graphs ($R=50$)

증명: 각각의 컷세트 C_k^*가 G 상에서 C_k에 대응한다고 하자. 모든 k에 대해서 $|P_k^1| = |P_k^2| = \ldots = |P_k^{2^k}|$임을 보임으로써 집합 C^*가 *admissible*함을 증명할 수 있다. 이것은 다음과 같이 변수 k에 대한 귀납법으로 증명한다.

(1) $k=1$인 경우, 타스크 모듈의 집합 P_0^1는 컷세트 C_1^*에 의해 P_1^1과 P_1^2로 분할된다. 분할 (P_1^1, P_1^2)이 균등 분할이 아니라고 가정하자. 즉, $|P_1^1| \neq |P_1^2|$. 그러면 $|P_1^{1'}| = |P_1^{2'}|$이 만족하도록 집합 P_0^1을 $P_1^{1'}$과 $P_1^{2'}$으로 분할하는 또 다른 컷세트 $C_1^{*'}(C_1')$가 그래프 $G_1^{*'}(G)$상에 존재한다. 그러면, $|P_1^1| + |P_1^2| = |P_1^{1'}| + |P_1^{2'}|$=상수 이므로, $|P_1^1| \cdot |P_1^2| + 1 \leq |P_1^{1'}| \cdot |P_1^{2'}|$이 성립한다. 따라서,

$$W(C_1^{*'}) - W(C_1^*) = |P_1^{1'}| \cdot |P_1^{2'}| \cdot R - W(C_1') - W(C_1) - (|P_1^1| \cdot |P_1^2| \cdot R - W(C_1')) \geq R - (W(C_1') - W(C_1)) > = 0$$

여기에서 마지막 부등식은 $R > \sum_{a<b} W_{a,b}$에 기인한다. 그러므로, $W(C_1^{*'}) - W(C_1^*)$이다. 이것은 (C_1^*)이 최대 컷세트라는 사실에 모순이다. 따라서, $|P_1^1| = |P_1^2|$.

(2) $k = i-1$일 경우에도 만족한다고 가정하자. 그러면, $|P_{i-1}^1| = |P_{i-1}^2| = \ldots\ldots = |P_{i-1}^{2^{i-1}}|$. 타스크 모듈들이 C_1^*에 의해 A_i와 B_i로 분할된다고 하자. 그리고 $k = i$인 경우 만족되지 않는다고 가정한다. 그러면, $1 \leq j \leq 2^{i-1}$인 모든 j에 대해서, $|A_i' \cap P_{i-1}^j| = |B_i' \cap P_{i-1}^j| = 2^{n-1}$이 성립하도록 타스크 모듈을 A_i'와 B_i'로 분할하는 또 다른 컷세트 $C_i^{*'}$ (C_i')가 그래프 $G_k^*(G)$상에 존재하게 된다.

그러면 $(\sum_{j=1}^{2^{i-1}} |A_i \cap P_{i-1}^j| + |B_i \cap P_{i-1}^j|)$

$= \sum_{j=1}^{2^{i-1}} |A_i' \cap P_{i-1}^j| + |B_i' \cap P_{i-1}^j|$

=상수이기 때문에

$\sum_{j=1}^{2^{i-1}} |A_i \cap P_{i-1}^j| \cdot |B_i \cap P_{i-1}^j|$

$+1 \leq \sum_{j=1}^{2^{i-1}} |A_i' \cap P_{i-1}^j| \cdot |B_i' \cap P_{i-1}^j|$

따라서,

$$W(C_i^{*'}) - W(C_1^*) = \sum_{j=1}^{2^{i-1}} |A_i' \cap P_{i-1}^j| \cdot |B_i' \cap P_{i-1}^j| \cdot R - W(C_i') - \left(\sum_{j=1}^{2^{i-1}} |A_i \cap P_{i-1}^j| \cdot |B_i \cap P_{i-1}^j| \cdot R - W(C_i) \right) \geq R - (W(C_i') - W(C_i)) \geq 0.$$

그러므로, $W(C_i^{*'}) > W(C_i^*)$ 이다. 이것은 C_i^*가 최대 컷세트라는 사실에 모순된다. 그러므로 $k=i$일 경우에도 성립한다. 따라서, 귀납법에 의해서 모든 k에 대해서도 만족한다.

정리 1에 의하면, *admissible*한 최소 컷세트 집합을 구하는 문제가 본 논문에서 제안한 그래프 변형 기법을 사용하여 최대 컷세트 집합을 구하는 문제로 변환이 된다. 따라서, 원래의 문제가 두 가지 최적화 목적을 가지는 반면에, 변환된 문제에서는 오직 한가지 최적화 목적만을 가지므로, 휴어리스틱 알고리즘을 고안하기가 훨씬 쉬워진다. 예를들어, 그림 3(a)의 타스크 그래프에 대해서, 각각의 변형된 그래프 $G_k^{*'}$와 그것의 최대 컷세트 $C_k^{*'}$가 그림 4에 나타나 있다. 여기에서 집합 $C^* = \{C_1^*, C_2^*, C_3^*\}$는 *admissible*하고, 이들 컷세트에 각각 대응하는 그래프 그림 3(a)에 있는 컷세트

C_k는 해당하는 그래프 상에서 각각 최소 컷세트이다.

4.2 반복 매핑 알고리즘

지금까지 그래프 상에서 균등 컷세트를 구하는(즉, 그래프를 균등하게 이분할하는) 효율적인 알고리즘이 많이 제시되어 왔다[13,14]. 참고문헌 [14]에서는 그래프 상에서 가중치가 최소인 균등 컷세트를 찾는 문제가 (즉, 최소 컷세트 문제) 그래프 변형을 통해 최대 컷세트를 찾는 문제로 변환됨을 보이고, 그래프에서 최대 컷세트를 찾는 MAXCUT 알고리즘을 제안하였다. 이 알고리즘에 대한 자세한 설명은 생략하고 간단하게 동작 원리를 설명하면 다음과 같다. 먼저 그래프 G가 주어지면, 이것을 G^*로 변형한다. 변형된 그래프 G^*의 임의의 두 노드 m_1, m_j 사이의 엣지값은 $C_{i,j}$이다. 그런 다음, 그래프 G^*의 노드들을 임의의 두 부집합 (A, B)로 이분할 한다. 이분할 (A, B)에서의 각 노드들의 이득 $(gain)$을 다음과 같이 정의한다

$$g(m_a) = \sum_{m_i \in A_k} c_{ai} - \sum_{m_j \in B_k} c_{aj}, \forall m_a \in A_k,$$
$$g(m_b) = \sum_{m_j \in B_k} c_{bj} - \sum_{m_i \in A_k} c_{bi}, \forall m_b \in B_k.$$

즉, 각 노드의 이득이란 그 노드를 다른 부집합으로 옮겼을 때 컷세트의 가중치가 증가되는 양이다. 각 부집합에서 이득이 가장 큰 노드를 선택하여 다른 부집합으로 옮김으로써 보다 개선된 이분할을 찾는 작업을 되풀이한다. 이 알고리즘은 연속적인 반복 수행 구간으로 구성이 되며, 각 구간에서는 모든 노드들이 정확히 한번씩 옮겨진다. 즉, 각 구간에서 옮겨질 노드는 그 구간에서 아직 옮겨지지 않은 노드들 중에서 이득

```
Algorithm MRB

입력 : |V| =2^n으로 구성된 타스크 그래프 G(V, E)
출력 : 일대일 맵핑 X
변수 정의
        X[1..N][1..n] : boolean;  /* X[i][k] = 모듈 m_i의 k 번째 비트 할당 */
        P[1..N][0..n] : integer; /* 모듈 m_i가 위치하는 분할 P_k^{p[i][k]}  */

1)      for I = 1 to N do  /* 초기 P_0^1 을 구성한다 */
            P[i][0] = 1;
            for j = 1 to n do
                    X[i][j] = 0;  /* 맵핑 초기화 */
2)      for k = 1 to n do
2.1)        for i = 1 to N do  /* 그래프 G로부터 변환 그래프 G_k^* 를 생성한다 */
                for j = 1 to N do
                    if P[i][k-1] = P[j][k-1]  /* 모듈이 동일 subset에 위치 */
                        then C_{i,j} W_{i,j};  /* 에지 무게를 수정한다 */
2.2)    (A_k, B_k) MAXCUT(G_k^*): /* G_k^* 상에서 maxcut을 찾는다 */
2.3)    for i = 1 to 2^{n-1} do  /* 2^{n-1} subsets을 구성한다 */
            if m_i ∈A_k then P[i][k] =2×P[i][k-1]-1;
            else P[i][k] =2×P[i][k-1];
2.4)    for i = 1 to N do  /* k 번째 비트를 할당 */
                if m_i ∈A_k then X[i][k] = 0;
            else X[i][k] = 1;
end.    /* 맵핑 X를 구하여 종료 */
```

(그림 5) 알고리즘 MRB
(Fig. 5) The Algorithm MRB

이 가장 큰 노드로 선택된다. 각 구간 동안에 $|V|$개의 새로운 이분할이 얻어지며, 이들 중 가장 컷세트의 가중치가 최대인 이분할을 선택하여, 그 다음 구간의 초기 분할로 사용한다. 컷세트의 가중치에 더 이상 개선이 없을 때까지 이와같은 수행이 계속된다. 알고리즘 MAXCUT의 시간 복잡도는 $O(V^2)$이다[14].

본 논문에서는 알고리즘 MAXCUT을 이용하여 반복 이분할을 함으로써 타스크 모듈들을 하이퍼큐브의 각 프로세서에 일대일 매핑하는 MRB(*Mapping by Repeated Bipartitioning*) 알고리즘을 그림 5와 같이 제안한다. 각각의 이분할을 통하여 각 모듈들은 자신이 매핑될 프로세서의 주소 비트를 하나씩이 결정된다. 이와같은 과정을 n번 반복함으로써 모든 모듈들에 대해서 그들이 매핑될 프로세서의 주소가 완전히 결정되고, 다음 정리 2에 의해서 이 매핑은 일대일이 된다. 알고리즘 MRB는 3 단계로 구성된다. 첫번째 단계에서는 (step 2.1) 이분할 (A_k, B_k)가 *admissible* 하도록 그래프 G를 G_k^*로 변형한다. 두번째 단계에서는 (step 2.2) 알고리즘 MAXCUT을 이용하여 그래프 G_k^*를 이분할 한다. 이 단계는 시간 소비가 가장 많은 단계이다. 마지막 단계에서는 (step 2.3, 2.4) 2^k개의 부집합을 만들고, step 2.2의 이분할의 결과에 따라서 각 모듈에 대해 자신이 매핑될 프로세서의 k번째 주소 비트를 할당한다. MRB의 시간 복잡도는 알고리즘 MAXCUT을 n번 반복 수행하기 때문에 $O(nN^2)$이다.

정리 2: $|V|=2^n$인 타스크 그래프 $G(V, E)$에 대해 변형 그래프 G_k^*상에서 알고리즘 MRB를 통해 구한 집합 $C^*=\{C_k^* \mid 1 \le k \le n\}$는 *admissible* 하다. 즉, 대응하는 매핑 X는 일대일이다.

증명: 다음과 같이 k에 대한 귀납법으로 $|P_k^1|=|P_k^2|=|P_k^{2^k}|$임을 보임으로써, 집합 C^*가 *admissible*하다는 것을 증명한다.

(1) $k=1$인 경우, 타스크 모듈들은 알고리즘 MAXCUT에 의해서 구해진 컷세트 C_1^*에 의해 P_1^1, P_1^2로 이분할된다. 알고리즘 MAXCUT의 특성으로 인해 이분할 (P_1^1, P_1^2)하에서 모든 노드들의 이득은 0과 같거나 큰 양의 정수값을 가질 것이다. (P_1^1, P_1^2)이 균등 분할이 아니라고 가정하자. 즉, $|P_1^1| \neq |P_1^2|$. 그때, $|P_1^1| + |P_1^2| = 2^n$=짝수이므로, $||P_1^1| - |P_1^2|| \ge 2$. 또한, 일반성 상실없이, $|P_1^1| \ge |P_1^2| + 2$라고 가정한다. 그래프 G에서 임의의 두 모듈 m_i와 m_j사이의 엣지값은 그래프 G와 G^* 상에서 각각 $W_{i,j}$와 $c_{i,j}$이다. 그때, P_1^1에 있는 각 노드 m_a의 이득 $g(m_a)$는 다음과 같다.

$$\begin{aligned} g(m_a) &= \sum_{m_i \in P_1^1} c_{ai} - \sum_{m_i \in P_1^2} c_{aj} \\ &= \left((|P_1^1| - 1) \cdot R - \sum_{m_i \in P_1^1} W_{a,i} \right) \\ &\quad - \left((|P_1^2|) \cdot R - \sum_{m_j \in P_1^2} W_{a,j} \right) \\ &\ge R - \left(\sum_{m_i \in P_1^1} W_{a,i} - \sum_{m_i \in P_1^2} W_{aj} \right) \ge 0. \end{aligned}$$

다시 말해서, 분할 (P_1^1, P_1^2)이 균등 분할이 아니고, $|P_1^1| > |P_1^2|$인 경우, P_1^1에 있는 모든 노드는 양수값의 이득을 가진다. 이것은 모순이다. 따라서, (P_1^1, P_1^2)는 균등 이분할이다. 즉, $|P_1^1| = |P_1^2|$이다.

(2) $k=i-1$인 경우에도 만족한다고 가정한다. 그러면, $|P_1^1| = |P_1^2| = \cdots = |P_{i-1}^{2^{i-1}}|$이다. 타스크 모듈들이 컷세트 C_i^*에 의해 A_i와 B_i로 이분할 되고, 모든 $j(1 \le j \le 2^{i-1})$에 대해서, $A_i' \bigcap P_{i-1}^j = P_i^{2j-1}$이고 $B_i' \bigcap P_{i-1}^j = P_i^{2j}$이라고 하자. (1)에서와 마찬가지 이유로, 각 노드들의 이득은 음수가 되지 않아야한다. 그리고, $k=1$인 경우에는 만족하지 않는다고 가정하자. 이것은 $|P_i^{2j-1}| \neq |P_i^{2j}|$인 j가 $1 \le j \le 2^{i-1}$ 사이에서 적어도 하나 존재함을 의미한다. 그때, $|P_i^{2j-1}| + |P_i^{2j}| = |P_{i-1}^j| = 2^{i-1}$이기 때문에, $||P_i^{2j-1}| - |P_i^{2j}|| \ge 2$이다.

$|P_i^{2j-1}| \ge |P_i^{2j}| + 2$라고 가정하면, 그때, P_i^{2j-1}에 있는 각 노드 m_a의 이득 $g(m_a)$는 다음과 같다.

$$\begin{aligned} g(m_a) &= \sum_{m_b \in A_i} c_{ab} - \sum_{m_c \in B_i} c_{ac} \\ &= \left((|P_1^{2j-1}| - 1). R - \sum_{m_b \in A_i} W_{a,b} \right) \\ &\quad - \left((|P_i^{2j}|). R - \sum_{m_c \in B_i} W_{a,c} \right) \\ &\ge R - \left(\sum_{m_b \in A_i} W_{a,b} - \sum_{m_c \in B_i} W_{a,c} \right) \\ &\ge 0. \end{aligned}$$

이것은 모순이다. 따라서 admissible하기 위한 조건은 $k=i$일 때에도 만족한다. 그러므로 귀납법의 원리에 의해 모든 k에 대해서 만족한다.

5. 실험 결과

본 논문에서 제안한 알고리즘 MRB의 성능을 측정하기 위해서 여러 종류의 임의 및 정형 그래프 상에서 모의 실험을 하였으며, 기존의 greedy 및 recursive 매핑 알고리즘과 비교 하였다. 모의 실험은 *Sparse, Normal* 그리고 *Dense*등 세 가지 종류의 임의 그래프 상에서 실험을 하였다. *Sparse, Normal* 그리고 *Dense* 그래프들은 각각 엣지의 개수가 *2N(N-1)/14, 3N(N-1)/14* 및 *4N(N-1)/14* 인 그래프로 정의하였다. 또한 엣지값은 주어진 범위 내에서 임의로 설정하였다. 즉, 범위-k인 그래프의 각 엣지값은 1과 *k*사이에서 무작위로 선택하였다. 각각의 그래프 종류에 대해서 세가지 경우로 구분하였다. (즉, *k*=1,5,10.) 각 경우마다 100개의 임의 그래프 상에서 매핑을 하였으며, 이들 매핑 비용들의 평균값과 표준 편차에 대해 실험한 결과를 기록하였다.

표 1은 3-차원 임의 타스크 그래프에 대한 실험 결과를 보여주고 있으며, 여기서 최적해(opt.)는 단순한 *exhaustive* 탐색 알고리즘을 통해 얻었다. 3가지 방법 즉, *greedy, recursive* 그리고 MRB 모두는 *ramdom*으로 생성한 매핑보다 아주 뛰어난 매핑 결과를 보여주었다. 일반적으로, *recursive* 알고리즘이 *greedy* 알고리즘 보다 조금 우수한 매핑 결과를 보였으며, 그들 중에서 MRB 알고리즘의 성능이 모든 경우에 대해서 가장 우수하였다. MRB 알고리즘은 3-차원 타스크 그래프에서 최적해에 근사한 매핑 결과를 보였다. 표 2는 10-차원에서의 실험 결과를 보여준다. 이 경우에 대해서는 *exhaustive* 탐색을 통한 최적해를 구하는 것이 현실적으로 불가능하기 때문에 제시를 하지 않았다. (예를들어, 4-차원 타스크 그래프만 하더라도 16!, 즉 가지의 매핑 방법이 존재한다.)

〈표 2〉 10-큐브 상에서의 임의 타스크 그래프에 대한 결과 (N=1024)(모든 데이터는 1/1000로 축소)
〈Table 2〉 Results for random task graphs on 10-cubes (N = 1024). (All datas are downscaled to 1/1000)

Graph type	Range	random		greedy		recursive		MRB	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
sparse	1	786	1.60	765	1.67	745	1.41	737	1.52
	5	2357	5.22	2328	5.20	2218	5.18	2192	4.83
	10	4719	14.21	4701	15.34	4439	13.17	4387	13.14
normal	1	1310	2.15	1299	2.22	1267	2.38	1258	2.33
	5	3933	6.69	3924	6.37	3773	5.85	3742	5.79
	10	7177	16.15	6785	17.18	5943	16.34	5830	16.32
dense	1	1867	1.67	1802	1.71	1675	1.63	1637	1.62
	5	5583	6.29	5282	6.42	4860	6.98	4735	6.60
	10	10179	17.65	9692	19.20	8719	18.58	8594	18.25

이들 표에서 보듯이, 차원이 증가할수록 (즉, 프로세서의 수가 증가할수록) MRB 알고리즘의 성능이 *greedy* 및 *recursive* 매핑 알고리즘에 비해 성능이 더욱 우수하다는 것을 알 수 있다.

6. 결 론

본 논문은 병렬 프로그램에서 타스크 모듈을 하이퍼큐브 각 프로세서에 일대일로 매핑하였을 때, 총 통

〈표 1〉 3-큐브 상에서의 임의 타스크 그래프에 대한 결과 (N=8)
〈Table 1〉 Results for random task graphs on 3-cubes (N = 8)

Graph type	Range	random		greedy		recursive		MRB		Opt.	
		Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.	Avg.	Std.
sparse	1	13.36	4.69	10.30	4.02	9.12	3.48	8.82	3.34	8.66	3.26
	5	43.44	15.94	33.06	14.06	29.40	12.09	29.06	11.71	28.30	11.49
	10	82.39	30.55	56.82	25.78	51.86	22.93	51.28	23.06	49.76	22.29
normal	1	25.08	5.23	22.22	5.71	20.12	5.20	19.56	4.95	19.26	4.86
	5	70.95	18.24	58.33	16.52	53.92	13.62	53.12	13.34	51.90	13.14
	10	124.65	35.48	106.06	34.08	94.74	29.00	93.90	28.44	91.64	28.45
dense	1	33.28	4.88	31.12	5.31	28.76	4.54	28.40	4.43	28.22	4.55
	5	101.14	18.58	88.12	14.78	82.24	13.70	81.30	13.39	79.82	12.91
	10	178.33	32.90	163.08	32.03	153.86	29.84	151.72	29.02	149.70	29.61

신 비용을 최소화하는 가장 효율적인 방법을 제안하였다. 이 알고리즘은 반복 이분할 기법(Repeated Bi-partitioning Strategy)에 기초를 두고 있다. 먼저, 그래프 변형 기법을 사용하여 하이퍼큐브 상에서의 매핑 문제가 그래프에서의 최대 컷세트 집합을 구하는 문제로 변환됨을 보였다. 또한 균등 이분할을 보장하는 참고문헌 [14]의 그래프 이분할 알고리즘을 반복 적용함으로써 매핑 문제를 효율적으로 해결할 수 있음을 보였다.

제안된 알고리즘의 성능을 분석하기 위해서, 여러가지 임의 및 정형 그래프 상에서 실험을 하였으며, 이들 그래프에 대해서 본 알고리즘의 성능이 우수함을 보였다. 또한, 프로세서의 수가 증가할수록, 기존의 *greedy* 및 *recursive* 알고리즘에 비해 성능이 훨씬 우수하다는 것을 보였다. 실제 문제에 있어서, 타스크 그래프는 보통 정형 그래프이거나 혹은 이미 알려진 구조를 지니게 된다. 제안된 알고리즘은 하이퍼큐브 -*isomorphic* 혹은 '거의' *isomorphic*한 그래프나 또는 메쉬와 같은 정형 그래프 상에서 더욱 성능이 우수하다는 것을 알 수 있었다.

본 논문에서 제안한 비용 함수에서는 통신 비용만을 고려하고 있으나, 앞으로는 부하 균등이나, 타스크의 반응 시간, 또는 시스템 자원의 이용률 등을 고려할 수 있도록 많은 연구가 필요하다고 본다. 또한 본 논문에서는 아주 간단한 병렬 수행 모델을 가정하고 있으나, 앞으로는 타스크들 사이의 선행 관계나 실시간 특성 등을 고려한 새로운 모델에 대한 연구도 수행되어야 한다.

참 고 문 헌

[1] M.-S. Chen and K. G. Shin, "Embedding of interacting task modules into a hypercube," *Proc. Of the Second Conf. On Hypercube Concurrent Computers and Applications,"* pp.122-129, Oct. 1986.

[2] B. Becker and H. U. Simon, "How robust is the n-cube," Proc. *Of 27th Annual Symp. On Foundations Computer Sci.,* pp.283-291, Oct. 1986.

[3] J. Kim, C.R.Das, and W. Lin, "A processor allocation scheme for hypercube computers," *Proc. Of the 1989 Conf. On Parallel Processing,* pp.232-238, Aug. 1989.

[4] J. Rattner, "Concurrent processing: a new direction in scientific computing," *Proc. Of AFIPS Conf.*, Vol.54, pp.157-166, 1985.

[5] J. P. Hayes, T. N. Mudge, et al., "Architecture of a hypercube supercomputer," *Proc. Of the 1986 Conf. On Parallel Processing,* pp.653-660, Aug. 1986.

[6] J. C. Peterson, et al., "The Mark III hypercube ensemble concurrent processor," *Proc. Of the 1985 Conf. On Parallel Processing,* pp.71-73, Aug. 1985.

[7] W. D. Hills, *The connection machine,* MIT Press, Cambridge, MA, 1985.

[8] S.H. Bokhari, "On the mapping problem," *IEEE Trans. On Computers*, Vol.C-30, No.3, pp.207-214, Mar. 1981.

[9] W. J. Dally, "Deadlock free message routing in multiprocessor interconnection network," *IEEE Trans. On Computers*, Vol.C-36, No.5, pp.547-553, May 1987.

[10] D. W. Krumme, K. N. Venkataraman, and G. Cybenko, "Hypercube embedding is NP-complete," *Proc. Of the first Conf. On Hypercube Concurrent Computers and Applications,* pp.148-157, Aug. 1985.

[11] W.-K. Chen and E.F. Gehringer, "A graph-oriented mapping strategy for a hypercube," *Proc. Of the Third Conf. On Hypercube Concurrent Computers and Applications,* pp.200-209, Jan. 1988.

[12] F. Ercal, J. Ramanujan, and P. Sadayappan, "Task allocation onto a hypercube by recursive nincut bipartitioning," *Proc. Of the Third Conf. On Hypercube Concurrent Computers and Applications*, pp.210-221, Jan. 1988.

[13] B.W.Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs," *Bell Syst. Tech. J.,* Vol.49, pp.291-307, Feb. 1970.

[14] C.-H. Lee, C. -I. Park, and M. Kim, "Efficient algorithm for graph-partitioning problem using a problem transformation method," *Computer-Aided*

Design, Vol.21, No.10, pp.611-618, Dec. 1989.

[15] B.-R. Tsai and K. G. Shin, "Communication-oriented assignment of task modules in hypercube multicomputers," *Proc. Of 12-th Int'l Conf. On Distributed Comput. Syst.*, pp.38-45, June 1992.

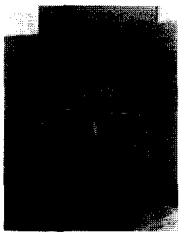
[16] P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Comput. Networks*, Vol.3, pp.85-93, Jan. 1977.

[17] H. S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Trans. On Software Engineering*, Vol.SE-3, No.1, pp.85-93, Jan. 1977.

[18] C.-H. Lee, D. Lee, and M. Kim, "Optimal task assignment in linear array networks," IEEE *Trans. On Computers*, Vol.C-41, No.7, pp.877-880, July 1992.

[19] C.-H. Lee and Kang G. Shin, "Optimal Task Assignment in Homogeneous Systems," *IEEE Trans. On Parallel and Distributed Systems*, Vol.8, No.2, pp.119-129, Feb. 1997.

[20] J.-M Kim, S-H Yoon, and C-H Lee, "An efficient repeated one-to-one mapping of task modules in hypercube multicomputer," *Proc. Of the 16th IASTED Int'l Conf. On APPLIED INFORMATICS.*, pp.364-367, Feb 1998.



김 주 만

jmkim@computer.etri.re.kr

1984년 숭실대학교 전자계산학과(공학사)

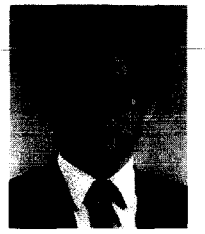
1998년 충남대학교 컴퓨터공학과(공학석사)

1995년~1996년 미국 노벨 및 SCO사 국제공동연구 파견근무

1985년~현재 ETRI 컴퓨터·소프트웨어기술연구소 OS연구팀 책임연구원

1994년 정보처리 기술사(전자 계산기 조직 응용)

관심분야 : 운영체제, 병렬 처리, 결함 허용 시스템



윤 석 한

shyoon@computer.etri.re.kr

1977년 고려대학교 전자공학과 졸업(공학사)

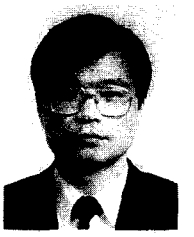
1986년 한국과학기술원 전산학과 졸업(공학석사)

1995년 고려대학교 전자공학과 졸업(공학박사)

1977년~1985년 한국전자기술연구소 선임연구원

1985년~현재 ETRI 컴퓨터·소프트웨어기술연구소 컴퓨터시스템 연구부 책임연구원(부장)

관심분야 : 컴퓨터 구조, 병렬처리 구조, 멀티미디어처리 시스템



이 철 훈

chlee@comeng.chungnam.ac.kr

1983년 서울대학교 전자공학과(공학사)

1988년 한국과학기술원 전기 및 전자공학과(공학석사)

1992년 한국과학기술원 전기 및 전자공학과(공학박사)

1983년~1986년 삼성전자 컴퓨터개발실 연구원

1992년~1994년 삼성전자 컴퓨터사업부 선임연구원

1994년~1995년 University of Michigan 객원연구원

1995년~현재 충남대학교 컴퓨터공학과 조교수

관심분야 : 운영체제, 병렬처리, 결함허용 및 실시간 시스템