論文 83-20-6-12

의사 쌍대 그래프 모델을 이용한 동적 태스크 할당 방법
(Dynamic Task Assignment Using a Quasi-Dual Graph Model)

金 德 壽*, 朴 容 震**

(Deok Soo Kim and Yong Jin Park)

要 約

본 논문에서는, 처리 능력이 다른 두 개의 프로세서에 태스크를 최적으로 할당하기 위해, 동적인 재배치를 고려할 수 있는 의사 쌍대 그래프 모델을 제안한다. 전체 처리 비용을 최소화하기 위하여 태스크를 구성하고 있는 모듈들을 두 프로세서에 최적 할당하는데, 이 그래프상에 복잡도가 $O(n^2)$인 최단 경로 결정 알고리즘을 적용하여 해결할 수 있음을 보였다.

Abstract

We suggest a Quasi- dual graph model in consideration of dynamic module assignment and relocation to assign task optimally to two processors that have different processing abilities. An optimal module partitioning and allocation to minimize total processing cost can be achieved by applying shortest-path algorithm with time complexity $O(n^2)$ on this graph model.

I. 序 論

요구되는 정보 처리량의 증가에 따라, 다수의 컴퓨터를 연결해 사용하는 컴퓨터 네트워크가 발달하기 시작하고, 또한 다중 프로세서 시스템들이 개발되고 있다.

응용 기술도 발전하여, 자원(resources)의 공유와, 일을 여러 개의 프로세서에 적절히 분담시켜 협동해 해결하기 위한 분산 처리에 대한 연구가 여러 측면에서 진행중이다.

그 중의 하나가, 커다란 태스크가 있을 때 이것을 여러 개의 프로세서가 분담하여 처리하는 것인데,[1,4,10,12] 그 프로세서들의 특성이 다르면 모듈들을 처리하는데 드는 비용도 다르므로 다음과 같은 문제가 발생한다. 즉, 태스크를 어떻게 분할하고(partitioning), 각 프로세서에 어떻게 할당(allocation)해야 전체 처리 시간 또는 비용이 최소화 될 것인가와, 이때 모듈화된 태스크를 이루는 모듈간의 상호 통신 비용은 어떻게 다루어야 할 것인가 하는 것 들이다.

이러한 문제에 대해서 Stone[2,3,4] 등은 그래프 이론적 측면에서 네트워크 플로우 알고리즘을 이용한 최적해를 구하는 방법을 제시한 바 있으며 그림1과 같은 모델을 사용했다.

한편, 처리중에 모듈이 프로세서들간에 이동하는 동적인(dynamic) 경우에는 조금 더 복잡해져서 더 많은 인자들이 고려되어야 하는데, 듀얼 시스템상의 이런 문제에 대해서는 Bokhari[10] 가 역시 네트워크 플로우 알고리즘을 이용해 해결하는 방법을 제시했다.

그러나 이 방법은 여러 개의 비용함수를 나타내는 아아크들이 그래프 모델상에 복잡하게 연결되어야 하

*準會員, **正會員, 漢陽大學校 工科大學 電子工學科
(Dept. of Electron. Eng., Han-yang Univ.)

接受日字 : 1983年 9月 15日

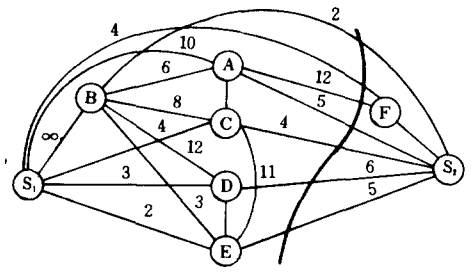

그림 1. 정적 모듈 할당을 위한 그래프 모델
Fig. 1. A graph model for static module assignment.

고, 복잡도가 큰 맥스-플로우 민-컷(max-flow min-cut) 네트워크 플로우 알고리즘[5-8]을 적용해야만 한다.

본 논문에서는, 단순화된 새로운 의사 쌍대 그래프 모델(quasi-dual graph model)을 제안하고 그 그래프 상에서 최단 경로 알고리즘(shortest path algorithm)을 이용하여, 태스크를 최적으로 할당하는 방법을 제시하고 그 예를 보인다.

II. 준 비

일반적으로, 태스크는 여러 개의 모듈로 구성되어 있으며, 모듈의 수가 프로세서의 수 보다 많은 것이 보통이다.

정적 태스크 할당 문제(static task assignment)에서는 모듈들이 어느 프로세서에 한번 할당되면 일이 완료될 때까지 다른 프로세서로 이동되지 않는다. 그러나, 우리가 다루려고 하는 동적 태스크 할당(dynamic task assignment) 문제는, 처리 과정중 모듈이 한 프로세서에서 다른 것으로 이동하며 모듈간의 상호 통신 패턴도 변화가 가능하다. 이 동적 할당문제에서의 척도(measure)는 시간 또는 비용의 함수인 코스트로 나타내며 어느 코스트이든 같은 단위를 갖는다고 가정한다.

이때 전체 코스트는 처리 비용과, 모듈간 상호 통신 비용(IMC cost)과, 재배치 비용(relocation cost)의 합이다. 처리비용은 한 모듈의 실행 비용(execution cost) 또는 나머지 모듈들이 각 프로세서에 남아 있는 주재 비용(residence cost)인데, 실행중인(active) 모듈은 별표(*)로 나타내며, 그 모듈의 코스트가 실행 비용이다. 일반적으로 실행 비용이 주재비용보다 크다.

본 논문에서는 듀얼 프로세서 시스템에 대해 생각할 것인데, 두 개의 프로세서 특성이 다르므로 모듈의 실행 비용과 주재 비용으로 구성된 처리 비용은 프로세서에 따라 다른 값을 갖는다. 처리 비용은 P2 프로세서에 대한 코스트 대 P1 프로세서에 대한 것으로 표현하며 cost[P2/P1]의 형식으로 쓴다. 모듈간 상호 통신 비용(IMC cost, inter-module communication cost)은 실행 모듈과 나머지 모듈간의 통신 비용을 뜻한다. 재배치 비용은 한 모듈이 다른 프로세서로 이선할 때 소요되는 비용이다.

여기서, 단 한 모듈만이 액티브하게 실행되며, 모듈간의 상호 통신 패턴이 변화되지 않는 시간 대를 페이즈(phase)라 정의하자. 그리고 한 페이즈내에서는 모듈의 재배치가 생기지 않고, 페이즈에서 페이즈로의 천이 과정에서만 재배치가 일어난다고 한다. 각 페이즈는 처리 비용, IMC 비용, 재배치 비용 등에 관한 정보를 갖는다.

4개의 모듈과 5개의 페이즈를 갖는 동적 할당 문제에 대한 코스트의 테이블이 다음과 같이 주어졌다고 하자.

표1에서 보면, 페이즈 5에서는 재배치 비용이 없는데, 이것은 처리가 완료되어 더 재배치할 필요가 없

표 1. 동적 할당을 위한 비용 정보 테이블
Table 1. A cost information table for dynamic assignment.

페이즈	모 듈	처리비용 [P2/P1]	IMC비용	이동비용
1	A	3 / 2	2	3
	B*	16 / 18	–	2
	C	1 / 4	6	3
	D	2 / 4	3	2
2	A	4 / 1	0	3
	B	2 / 3	3	2
	C*	12 / 15	–	1
	D	2 / 4	5	2
3	A*	∞ / 18	–	3
	B	6 / 3	4	2
	C	2 / 5	2	1
	D	2 / 4	0	4
4	A	7 / 3	2	2
	B*	20 / 17	–	1
	C	2 / 4	4	3
	D	1 / 3	5	3
5	A	4 / 2	4	
	B	5 / 2	0	
	C	3 / 5	3	
	D*	15 / 19	–	

(주) * : 실행모듈
처리비용 : 실행비용 또는 주재비용

기 때문이다. 표 1에서 주어진 값을 이용해서 각 모듈을 노드로, 모듈 간의 통신이나 재배치 상태를 아아크로 표현한다. 또한 실행 비용 또는 주재 비용인 처리 비용을 노드의 웨이트(weight)로, IMC 비용은 수평 아아크의 웨이트로, 재배치 비용은 수직 아아크의 웨이트로 주어 그림 2와 같은 동적 할당을 위한 그래프 모델을 얻는다.

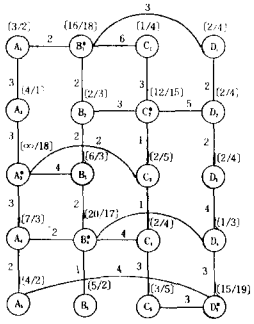

그림 2. 동적 할당을 위한 그래프 모델
Fig. 2. A graph model for dynamic assignment.

이때 실행 비용 또는 주재 비용으로 구성되는 처리 비용은 프로세서에 따라 다르며, 전체 비용 함수는 식 (1)로 표현된다.

전체비용=처리비용(실행비용 또는 주재비용)
+IMC 비용+재배치 비용 (1)

또한, 한 프로세서로 할당된 모듈들은 하나의 모듈로 간주되어(fusing), 그 모듈들간의 상호 통신 비용은 0로 취급한다.

Ⅲ. 의사 쌍대 그래프 모델화

본 Ⅲ장에서는 원 그래프 모델부터 그것과 쌍대관계가 있는 의사 그래프 모델을 구하고 그 웨이트를 부여하는 방법에 대해 기술한다.

먼저, 원 그래프 그림 2는 좌 우편에 점선으로 IMC 비용이 0이고 재배치 비용이 무한대(∞)인 더미(dummy) 노드를 갖고 있는 것으로 볼 수 있다. 또한 IMC가 없는 노드 쌍은 IMC비용이 0인 아아크가 있는 것으로 여긴다. 그러면 그림 2는 그림 3과 같은 모델이 되며, 여기서 모듈사이를 통과하는 두터운 선은 P1과 P2로의 분할을 나타낸다.

1. 모델의 구성

그림 3의 더미 노드와 아아크를 갖는 그래프 모델에서, 네 개의 노드로 이루어진 사각형안에 각각 새 노드를 삽입한다. 그리고 위와 아래 쪽에 두 개의 노드

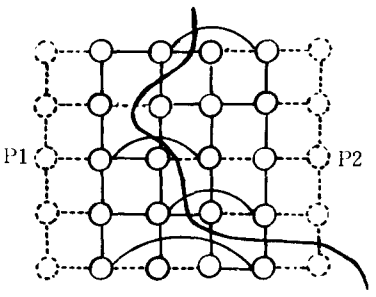

그림 3. 더미 노드를 갖는 모델의 임의 할당
Fig. 3. An arbitrary assignment with dummy nodes.

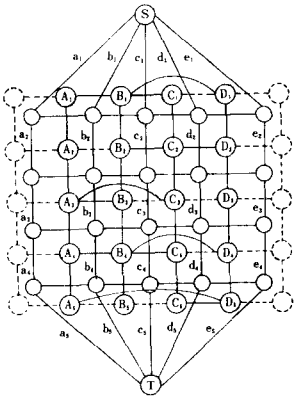

그림 4. 의사 쌍대 그래프 모델
Fig. 4. Quasi-dual graph model.

를 첨가해 아아크로 연결하면 그림 4와 같은 망 구조의 새로운 의사 쌍대 그래프가 얻어지는데, 이하 Q-dual 그래프 모델이라 한다.

이 Q-dual 그래프에서 아아크는, 그림 3의 원 그래프를 P1과 P2의 두 프로세서로 할당하기 위해 분할하는 임의의 컷(cut)이 지나가는 모든 경로가 된다. 또한 노드 S와 T는 그 분할의 컷이 시작하고(starting) 끝나는 노드(termination node)이다.

Q-dual 그래프 모델의 노드들을 연결하는 가로 아아크는 그림 2의 원 그래프에서 세로 아아크를 통과하는 컷의 일부이다. 그리고 Q-dual 그래프의 세로 아아크는 그림 3의 모듈과 모듈사이를 분할한다.

예를 들면, Q-dual 그래프의 아아크 C_3는 페이즈 3에서 모듈 B_3와 모듈 C_3 사이를 지나는 컷의 일부로서, C_3 왼쪽의 모듈 A_3, B_3는 프로세서 P1에, 오른쪽의 모듈 C_3, D_3는 P2에 할당한다는 것을 의미한다. 특히 아아크 a_1은 모듈 A_1, B_1, C_1과 D_1이 모두 P2로, 아아크 e_1은 모듈 A_1, B_1, C_1, D_1들이 모두

P1으로 할당되는 것을 뜻한다. 만일 컷이 a_1, a_2, a_3, a_4, a_5 이거나 e_1, e_2, e_3, e_4, e_5이면 P2 또는 P1의 한 프로세서상에서 처리되는 것이 가장 비용이 적게 든다는 것을 뜻하나, 이런 경우는 고려하지 않는다.

이때 만일 모듈이 m개이고 페이즈가 p개이면 원 그래프 모델은 mp개의 노드로 구성되며, Q-dual 그래프는 (m+1)개의 가로와 (p-1)개의 세로 노드를 가지므로 (m+1)(p-1)개의 노드로 구성된다. 따라서 m>p-1이면 더 적은 수의 노드로 구성되는 간단한 모델이 된다.

2. 웨이트 부여법

Q-dual 그래프의 가로 아아크는 원 그래프의 세로 아아크와 교차하므로, Q-dual 그래프의 가로 아아크 웨이트는 원 그래프의 세로 아아크 웨이트인 재배치 비용과 각각 같음을 쉽게 알 수 있다. 따라서 우리는 그림3에서 새로운 가로 아아크 웨이트와 직교하는 원 그래프의 세로 아아크 웨이트와 같게 부여하면 된다.

한편, Q-dual 그래프의 세로 아아크는 원 그래프의 모듈들을 두 개의 다른 프로세서로 배당하게 하는 분할을 나타내는 컷과 대응하는데, 모듈 쌍간의 IMC를 나타내는 아아크와 교차하고 있다. 이때 웨이트를 구하는 방법을 알아 보기 위해, 여기서는 6개의 모듈로 구성된 한 페이즈를 고려하고 이 모듈들은 서로 완전한 상호 통신이 이루어지고 있다고 가정한다.

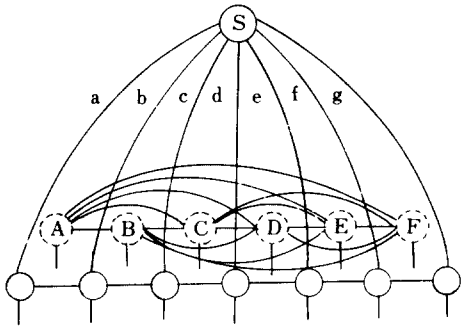

그림5. 6개의 모듈로 된 모델과 Q-dual 그래프 간의 관계

Fig. 5. A relationship between 6-module model and Q-dual graph model.

그림5에서 a, b, c, d, e, f, g는 원 그래프 모델의 모듈들을(점선으로 표시된 것) 분할할 컷들이며, 이들에 의해 각 모듈은 P1 또는 P2 프로세서중의 하나로 배당될 것이다. 아아크 a가 컷이라면 모든 모듈은 다 프로세서 P2로 할당될 것이며, 반대로 아아크 g는 모든 모듈이 P1으로 할당됨을 의미한다. d는 왼쪽의 모듈 A, B, C는 프로세서 P1에, 오른쪽의 모듈 D, E, F는 P2로 할당되게 하는 컷이다. 이것을 표로 표시하면 다음과 같다.

표 2. 각 아아크에 따른 모듈의 프로세서로의 배당
Table 2. Assignment of modules to processors according to each arc.

아아크 / 모듈(노드)	a	b	c	d	e	f	g
A	2	1	1	1	1	1	1
B	2	2	1	1	1	1	1
C	2	2	2	1	1	1	1
D	2	2	2	2	1	1	1
E	2	2	2	2	2	1	1
F	2	2	2	2	2	2	1

이 표2에서 아아크 C는, 모듈 A, B는 1이므로 P1에 모듈 C, D, E, F는 2이므로 P2로 할당되는 것을 알 수 있다.

또한 그림5에서 보듯이 Q-dual 그래프 모델의 세로 아아크 a, b, c, d, e, f, g는 원 그래프 모듈사이의 컷을 교차하는데, 모듈 A와 B사이의 IMC 패턴을 AB, D와 F 모듈간의 IMC를 DF라고 쓴다면 각 아아크 b, c, d, e, f가 교차하는 IMC 패턴은 다음과 같다.

표 3. 모듈 쌍간의 IMC 패턴
Table 3. IMC patterns between each module pair.

컷	a	b	c	d	e	f	g
IMC 패턴		AB C D E F	AC D E F BC D E F	AD E F BD E F CD E F	AE F BE F CE F DE F	AF B C D E	

(주) AD
E 는 AD, AE, AF를 뜻한다.
F

표 3을 일반화하면 n개의 모듈(모듈 1, 2, …, n−1, n) 간의 교차되는 IMC 패턴은 다음 표 4와 같다. 단 m<n이고 m과 n은 양의 정수이다.

표 4. 임의 개의 모듈 쌍간의 확장된 IMC 패턴
Table 4. Extended IMC patterns between any module pair of a lot.

컷	a	b	c	d	… … …		…		
모듈 쌍간의 IMC 패턴		1 2 1 3 1 4 1 m 1 (n−1) 1 n 2 3 2 4 2 m 2 (n−1) 2 n 3 4 (m−1) m (n−2) (n−1) n							

위의 표 2와 표 3을 보고, Q-dual 모델의 각 아아크의 웨이트를 쓰면 다음과 같다.

$W(a) = E[(A+B+C+D+E+F)|_{P2}]$

$W(b) = E[(A)|_{P1} + (B+C+D+E+F)|_{P2}] + C[(A)(B+C+D+E+F)]$

$W(c) = E[(A+B)|_{P1} + (C+D+E+F)|_{P2}] + C[(A+B)(C+D+E+F)]$

$W(d) = E[(A+B+C)|_{P1} + (D+E+F)|_{P2}] + C[(A+B+C)(D+E+F)]$

$W(e) = E[(A+B+C+D)|_{P1} + (E+F)|_{P2}] + C[(A+B+C+D)(E+F)]$

$W(f) = E[(A+B+C+D+E)|_{P1} + (F)|_{P2}] + C[(A+B+C+D+E)(F)]$

$W(g) = E[(A+B+C+D+E+F)|_{P1}]$

여기서 W(x)는 아아크 x의 웨이트 함수이고 E(x)는 실행 비용 또는 주재 비용으로 이뤄지는 처리 비용을 뜻하며, C(x)는 IMC 비용이다. 그리고 $C[(A+B)(C+D+E+F)] = C[AC+AD+AE+AF+BC+BD+BE+BF]$로 모듈 A, C 간의 IMC 비용+A, D 간의 IMC 비용+…의 합을 의미하여, $E[(A)|_{P1} + (B+C)|_{P2}]$는 모듈 A의 P1에 관한 처리 비용과 모듈 B, C의 P2에 대한 처리 비용의 합을 뜻한다. 특히 W(a)와 W(g)는 IMC 패턴과 교차하지 않으므로 처리 비용의 합으로만 이루어짐에 주목하자.

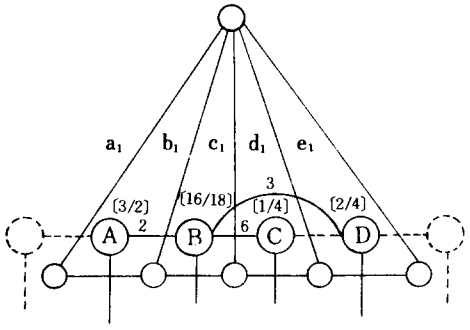

그림 6. Q-dual 그래프의 각 아아크 웨이트 부여의 예
Fig. 6. Illustration of weighting each arc of the Q-dual graph model.

이제 이 방법을 원래 문제인 4 모듈, 5 페이즈 시스템에서 본다.

아아크 C_1은 모듈 A, B를 P1에, 모듈 C, D는 P2 프로세서로 할당하는 컷이며 BC와 BD의 IMC 패턴과 교차한다. 앞의 방법에 근거해 웨이트를 부여하는 간단한 방법을 보자. 각 모듈의 [P2/P1] 형태의 처리 비용중 세로 아아크 x와 가까운 쪽의 처리 비용을 선택하고, 그것들과 교차하는 IMC 비용을 합한다. 즉 $x = C_1$이면, A, B 모듈은 처리 비용중 P1에 해당되는 slash의 오른쪽 값을, 모듈 C, D는 P2에 해당되는 왼쪽 값을, 즉 A와 B 모듈은 2와 18을, C와 D는 1과 2를 선택하여, 그 값에 IMC 비용 3과 6을 더한 값이 아아크 c_1의 웨이트가 된다.

$W(c_1) = (2+18)+(1+2)+(3+6) = 32(\text{units})$

같은 방법으로 각 아아크의 웨이트를 부여하면 된다.

Ⅳ. 최적 할당

동적 태스크 할당 문제에서 최적 할당이란, 각 페이즈별로 모듈들을 각 프로세서에 적절히 배당하여 Ⅱ의 식(1)의 전체 비용을 최소화하는 것에 대응한다. 그러면, Ⅲ의 2절에서와 같이 각 아아크의 웨이트가 기록된 Q-dual 그래프 모델을 완성했을 때, 최적 할당을 위해 컷을 결정하는 방법을 알아 본다.

Ⅲ장에서 언급했듯이, Q-dual 그래프 모델의 각 아아크는 원그래프 모델의 모듈을 프로세서 P1과 P2에 할당하는 컷의 일부이고, 그 웨이트는 그때 각 부분에서 소요되는 경비를 의미한다. 따라서, 최단 경로 결정 알고리즘을 사용해 S에서 T에 이르는 최단 경로를 구하면, 그때 선택된 Q-dual 그래프상의 경로 자체가 원 그래프 모델의 분할을 나타내는 컷이 되며, 최단 경로 알고리즘이 노드 T에 도달하면서 계산되어진 값이 이 시스템에서 모듈이 각기 P1과 P2에 할

당되어 태스크를 완전히 처리하는 최소화된 전체비용이 된다.

이때 Q-dual 그래프 모델상의 아아크 웨이트는 양의 값이므로, 최단 경로 알고리즘은 기존의 Dijkstra[11]의 것을 사용한다.

네트워크 플로우 알고리즘을 사용하는 Stone[2,4], Bokhari[10] 등의 기존 방법은 맥스-플로우 민-컷 알고리즘을 채택했는데 Ford and Fulkerson의 것[5]을 Edmonds and Karp가 구체화한 것[6], 또는 Dinic[7], Karzanov[8]의 개선된 것들을 사용하고 있다. 이 들 네트워크 플로우 알고리즘의 시간 복잡도(time complexity)는 m : 아아크 수, n : 노드 수 일 때 $O(nm^2) = O(n^5)$ 또는 $O(n^2m)$ 이고, Karzanov의 알고리즘은 $O(n^3)$ 이다.[9] 반면 Dijkstra의 최단 경로 알고리즘의 복잡도는 $O(n^2)$ (n : 노드의 수) 이므로, 본 알고리즘을 이용한 연산이 더욱 우수함을 알 수 있다.

앞에서 예로 든 그래프 모델에 최단 경로 결정 알고리즘을 적용한 결과를 보면 그림 7과 같고, 이때 이 태스크를 처리하기 위한 최소화된 전체 비용은 105이며, Q-dual 그래프 모델상에 선택된 경로와 같게 그림 3의 원 그래프 모델의 모듈들이 할당됨을 알 수 있다. 즉, 모듈 $A_1, A_2, A_3, B_3, A_4, A_5$는 프로세서 P1으로, 모듈 $B_1, C_1, D_1, B_2, C_2, D_2, C_3, D_3, B_4, C_4, D_4, B_5, C_5, D_5$는 P2로 할당된다.

여기서 모듈 B를 주목해 보면, 처음과 둘째 페이즈 동안에는 P2 프로세서에 있었으나 페이즈 3에서는 P1으로, 그리고 페이즈 4에서는 다시 P2로 이동하며 전체 처리 비용을 최소화하는데 기여하고 있다.

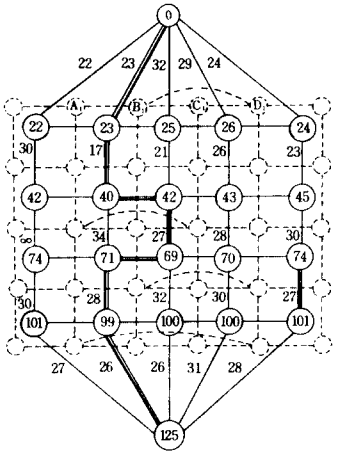

그림 7. 최단 경로 알고리즘 이용한 최소 비용 할당
Fig. 7. Min-cost assignment using a shortest-path algorithm.

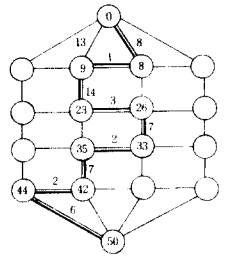

그림 8. Q-dual 그래프를 이용해 해결한 Brkhari의 예제
Fig. 8. Bokhari's example solved through Q-dual graph model.

참고로 본 논문의 Q-dual 그래프 모델을 사용한 방법을 Bokhari의 예제에 적용하면 다음 그림 8과 같으며, 네트워크 플로우 알고리즘을 이용한 방법과 동일한 결과를 갖는다.

V. 結 論

분산 처리 시스템에서 처리 능력이 다른 두 프로세서에, 커다란 태스크를 분할하여 모듈들을 최적 할당하는, 그 전체 처리 방법을 제시했다. 이 방법은 모듈이 각 페이즈에 따라 한 프로세서에서 다른 프로세서로 이동되는 동적인 문제에 관한 것이다.

비용에 관한 정보 테이블에서 Q-dual 그래프 모델을 구성하고, 그 그래프 모델에 최단 경로 결정 알고리즘을 적용하여 최단 경로를 얻었다. 이때 Q-dual 그래프 모델상의 경로가 원 그래프 모델상의 컷과 같으며, 이때 구해진 값이 최소화된 전체 처리 비용을 나타낸다.

그리고 본 논문에서 제시한 방법은 네트워크 플로우 알고리즘을 사용했던 경우와 같은 결과와 같으며, 시간 복잡도가 더 낮은 최단 경로 알고리즘을 이용하기 때문에 그 연산이 더 우수함을 알 수 있었다.

다수의 컴퓨터로 네트워크화된 분산 처리 시스템과 태스크를 임의 개의 프로세서로 분할, 할당하는 방법과, 컨커런트한 처리에 적합한 새로운 모델화와 그 처리법의 개발이 또한 요구된다.

VI. 謝 意

본 연구의 일부는 한국전기통신연구소의 1982년도 프로젝트 "고장 감내형 분산 제어계 구조에 관한 연구"의 지원을 받았다.

양승택 선임부장, 유완영부장, 여재홍실장(현 대영전자연구소 부소장)께 감사드린다.

參 考 文 獻

[1] V.B. Gylys and J.A. Edwards, *Optimal Partitioning of Workload for Distributed Systems.* in Dig. COMPCON Fall., pp. 353-357, 1976.

[2] H.S. Stone, "Multiprocessor scheduling with the aid of network flow algorithms," *IEEE Trans. Software Eng.*, vol. SE-3, pp.85-93, Ja. 1977.

[3] H.S. Stone, "Critical load factors in distributed computer systems," *IEEE Trans. Software Eng.*, vol. SE-4, pp.254-258, May 1978.

[4] G.S. Rao, H.S. Stone, T.C. Hu, "Assigment of tasks in a distributed processor systems with limited memory", *IEEE Trans. Computers,* vol C-28, no.4, pp.291-298, Apr. 1979.

[5] L.R. Ford, Jr. and D.R. Fulkerson, *Flows in Networks.* Princeton, NJ: princeton Univ. Press, 1962.

[6] J. Edmonds and R.M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems", *J.A.C.M.*, vol. 19, pp.248-264, Apr. 1972.

[7] E.A. Dinic, "Algorithm for solution of a problem of maximum flow in a network with power estimation," *Sov. math. Dokl.*, vol.11, no.5, pp.1277-1280, 1970.

[8] A.V. Karzanov, "Determining the maximal blow in a network by the method of preflows," *Sov. math. Dokl.*, vol.15, no.2, pp. 434-437, 1974.

[9] T.C. Hu, *Combinatorial Algorithms.* Addison-Weseley, 1982.

[10] S.H. Bokhari, "Dual processor scheduling with dynamic reassignment," *IEEE Trans. Software Eng.*, vol. SE-5, no.4, July 1979.

[11] E.W. Dijkstra, "(*A Note on Two Problems in Connection with Graphs*)", Numerische mathematic 1, pp. 269-271, 1959.

[12] 박용진, 김덕수 et al., 고장 감내형 분산 제어계 구조에 관한 연구, 한국전기통신연구소 1982년도 프로젝트 최종보고서, pp. 145-300, Feb. 1983.