論文98-35C-3-5

유전자 알고리즘을 이용한 다중 디스크 데이터 배치 방식

(Multidisk Data Allocation Method based on Genetic Algorithm)

安 大 榮 *, 朴 圭 晧 **, 林 基 郁 ***

(Dae-Young Ahn, Kyu-Ho Park, and Kee-Wook Rim)

요 약

본 논문에서 고찰한 다중 디스크 데이터 배치 문제는 입출력 처리의 병렬성이 극대화 되도록 데이터 블록들을 다중 디스크에 배치하는 방식을 찾는 것이다. 이 문제는 NP-hard의 부류에 속하는 문제로 알려져 있기 때문에 휴리스틱 접근 방식을 이용해야 한다. 최근에 제안된 대부분의 효율적인 방식들은 적용되는 디스크의 수가 2의 지수 승이어야 한다는 제약을 가지고 있어서 시스템 확장 시에 가격과 공간의 측면에서 문제를 제기할 수 있다. 본 논문에서는 유전자 알고리즘(*Genetic Algorithm*)을 이용하여 디스크의 수에 대한 제약을 제거하면서도 고성능의 입출력 성능을 제공하는 새로운 데이터 배치 방식을 제안한다. 스키마 이론을 이용하여 제안된 방식의 수렴성을 수학적으로 증명하였고, 기존의 제안된 General Disk Modulo(GDM), Binary Disk Modulo(BDM), Error Correcting Code(ECC) 방식들과 본 논문에서 제안된 방식에서 구해진 결과를 시뮬레이션을 통하여 비교하였다. 실험 결과는 제안된 본 방식이 디스크의 수에 제한이 없는 GDM 방식보다는 항상 좋은 결과를 제공하고 있으며, 디스크의 수에 제한을 가한 경우에도 BDM 방식과 견줄 수 있는 성능을 나타냄을 보여 준다.

Abstract

Multi-disk data allocation problem examined in this paper is to find a method to distribute a *Binary Cartesian Product File* on multiple disks to maximize parallel disk I/O accesses for partial match retrieval. This problem is known to be NP-hard, and heuristic approaches have been applied to obtain sub-optimal solutions. Recently, efficient methods have been proposed with a restriction that the number of disks in which files are stored should be power of 2. In this paper, we propose a new disk Allocation method based on Genetic Algorithm(GA) to remove the restriction on the number of disks to be applied. Using the schema theory, we prove that our method can find a near-optimal solution with high probability. We compare the quality of solution derived by our method with General Disk Modulo, Binary Disk Modulo, and Error Correcting Code methods through the simulation. The simulation results show that proposed GA is superior to GDM method in all cases and provides comparable performance to the BDM method which has a restriction on the number of disks.

* 正會員, 韓國電子通信研究院 컴퓨터構造研究室
(Computer Architecture Section, ETRI)
** 正會員, 韓國科學技術員 電氣 및 電子工學科
(Department of Electrical Engineering, KAIST)
*** 正會員, 韓國電子通信研究院 情報通信研究管理團
(Institite of Information Technology Assessment, ETRI)
接受日字:1997年8月29日, 수정완료일:1998年2月18日

Ⅰ. 서 론

최근의 데이터 베이스 시스템은 활용 영역이 광범위해졌을 뿐 아니라, 구축된 데이터의 크기도 급속도로 커지고 있다. 이러한 추세는 디스크 입출력의 성능에 관한 관심을 불러 일으키게 되었는데, 그 이유는 디스크 입출력 성능이 데이터 베이스 시스템의 성능을 결정하는 주 요소로 등장했기 때문이다. 디스크 입출력 성능을 향상시키기 위한 노력의 결과 중 하나로서 디

스크 어레이가 출현하였는데, 디스크 어레이 시스템에서는 디스크에 대한 입출력 요구가 여러 개의 디스크들에서 병렬로 처리되어 입출력 응답 시간이 눈에 띄게 개선되었다[1] [2] [3]. 이러한 입출력 병렬성은 데이터 베이스 응용의 하나로서 최근 많은 관심을 받고 있는 다중 어트리뷰트 파일(*multiattribute file*)에 대한 부분 일치 질의어(*partial match query*) 처리에 유용하게 적용되어 질의 응답 시간을 크게 향상시킬 수 있는 중요한 요소로 인식되고 있다.

본 논문에서 고찰하는 다중 디스크 데이터 배치 문제는 다중 어트리뷰트 파일 중의 하나인 Binary Cartesian Product File(*BCPF*)의 버킷(*bucket*)들을 다중 디스크에 배치하는 방법을 찾는 문제로서, 주어진 질의어 처리를 위한 디스크 입출력이 최대로 병렬로 수행되도록 하는 것이 설계 목표이다. 이 문제는 NP-complete의 부류에 속하는 문제로 알려져 있으며[4], 현재까지 제안된 대부분의 방식들은 휴리스틱(*heuristic*) 접근 방식을 이용하고 있다[4]-[11]. RDA(*Random Data Allocation*) 방식[11]은 버킷들을 랜덤 숫자 생성기를 이용하여 임의의 디스크로 할당하는 방식이다. 이 방식은 비교적 간단하기는 하지만, 이후 기술될 다른 방식에 비해 성능이 떨어지는 것으로 알려져 있다. DM(*Disk Modulo*) 방식[4]은 버킷 어드레스의 해밍 웨이트(*hamming weight*)를 이용하여 배치하는 방식으로서, 버킷 어드레스 $[i_1, i_2, \cdots, i_n]$와 디스크의 수 m에 대하여 할당될 디스크d를 $d=\left(\sum_{j=1}^{j=n} i_j\right) modulo\ m$ 의 사상 함수를 적용하여 결정하는 방식이다. 이 방식은 버킷 어드레스와 디스크의 수가 정해지면 고정적으로 버킷들의 배치가 결정되기 때문에 모든 디스크에 버킷들이 균일하게 할당되지 않고 특정 디스크에 편중될 수 있어서 입출력 성능이 저하되는 경우가 발생할 수 있다. 이러한 단점을 보완하기 위해 제안된 방식이 GDM(*General Disk Modulo*) 방식[4]이며, 식 $d=\left(\sum_{j=1}^{j=n} i_j \times p_j\right)$ *modulo* m에 의해 할당될 디스크가 결정된다. GDM 방식의 디스크 사상 함수에서 변수 값 p_j는 디스크의 수 m에 대하여 소수값(*prime value*)을 갖는 임의의 정수로 정의되어 있는데, Du와 Soboleski[5]는 최적화를 만족시키는 p_j의 조합은 주어진 시스템 구성에 대하여 없을 수도 있으며, 구하기도 매우 어렵다고 기술하였다. 또한, DM 방식이RDA 방식보다 더 우수한 성능을 제공하며, GDM 방식이 DM 방식 보다 더 우수함을 보였다. BDM(*Binary Disk Modulo*) 방식[6]은 디스크의 수가 2의 지수 승일 때 적용되며, 최적화와 응답 속도의 면에서 DM 방식보다 우수함이 증명되었다. BDM 방식의 디스크 할당 결정식은 GDM 방식에서 적용된 것과 같으나, p_j의 값이 다음의 식 $p_j=2^{(j \bmod ulo \log_2 m)}$으로 대치된다. ECC(*Error Correcting Code*) 방식[7] [12]은 각 디스크에 배치될 버킷들의 주소가ECC가 되도록 하는 방식으로서, 하나의 디스크에 배치되는 버킷의 주소들의 해밍 거리(*hamming distance*)가 최대가 되도록 배치시키는 방식이다. 이와 같이 배치함으로써 같은 디스크에 할당되는 버킷들의 주소가 서로 인접되지 않게 하여 최대의 입출력 병렬성을 추구한다. ECC 방식은 앞에서 설명한 다른 방식들보다 우수한 성능을 보장하는 것으로 알려져 있다.

지금까지 설명한 5가지 방식들은 적용되는 디스크의 수에 따라 크게 두 부류로 구분될 수 있다. RDA, DM, GDM 방식은 디스크의 수에 제한을 두지 않는 방식들이며, BDM, ECC 방식은 디스크의 수가 2의 지수 승일 때만 적용되는 방식이다. 성능적인 측면에서 종합해 보면, BDM, ECC 방식들이 디스크의 수에 제한을 두지 않는 방식들에 비해 같은 시스템 구성에서 훨씬 좋은 성능을 제공 한다. 그러나 디스크의 수가 2의 지수 승이어야 한다는 제약은 사용자가 시스템을 확장하고자 할 때, 경제적인 면에서나 공간적인 면에서 많은 문제를 제기한다. 예를 들어 현재 32개의 디스크를 장착하고 있는 시스템에서 같은 데이터 배치 알고리즘을 적용하여 저장 용량을 확장하려면 최소한 32개의 디스크를 추가로 구매하여야 하며, 이들을 장착 배치해야 할 새로운 기구물과 공간을 필요로 한다. 이러한 문제는 이 후 시스템 용량을 추가로 확장하고자 할 때는 더욱 심각한 문제를 발생시킨다.

본 논문에서는 디스크의 수에 대한 제약을 제거하면서도 디스크 수에 제한을 둔 방식과 견줄 수 있는 입출력 성능을 제공하는 방식으로서, 유전자 알고리즘(*Genetic Algorithm*)[13] [14]을 이용한 새로운 데이터 배치 방식을 제안한다. 유전자 알고리즘은 생태계의 유전자 교환을 통한 진화 방식과 적자생존 법칙에 기반을 둔 탐색(*search*) 방식으로서 여러 응용 분야에서 효율적으로 적용될 수 있는 견고성(*robustness*)을 그 특징으로 가지고 있어서 다양한 분야의

최적화 문제에 성공적으로 적용되었다.

본 논문의 구성은 다음과 같다. 2 장에서는 다중 디스크 데이터 배치 문제를 형식화하여 정의한다. 3 장에서는 본 논문에서 제안하는 방식에 관한 설계 개념을 제시하고,자세한 구현 내용은 4장에서 설명한다. 5 장에서는 스키마 이론을 이용한 알고리즘의 수렴성을 증명과정을 제시하고, 6 장에서는 시뮬레이션을 이용한 실험 결과로서 제안된 알고리즘의 특성 및 타 방식과의 성능 비교 결과를 제시 한다. 끝으로 7장에서는 결론 및 추후 연구 과제에 대하여 기술한다.

Ⅱ. 문제 정의

본 논문에서 고찰한 다중 디스크 데이터 배치 문제는 디스크에 대한 접근의 병렬성이 극대화 되도록 데이터 블록들을 다중 디스크에 배치하는 방식을 찾는 문제이다 이 문제를 보다 형식적으로 설명하기 위하여 다음의 몇 가지 용어를 정의한다.

(정의 1.1):

F는 n개의 어트리뷰트를 가진 레코드(*record*) 화일이고, D_i는 F의 i 번째 어트리뷰트의 영역(*domain*)으로서, 개의 독립적인 부분집합 $D_{i1}, D_i2, \cdots, D_{im_i}$로 나누어 진다고 가정한다. 이때, F의 데이터 블록(*bucket*)을 구성하는 모든 레코드가 $D_{1i_1} \times D_{2i_2} \times \cdots \times D_{ni_n}$로 나타내지는 영역에 있을 때, F를 *Cartesian Product File* 이라고 정의한다. 여기서, D_{ji_j}는 $D_{j1}, D_{j2}, \cdots, D_{jm_j}$중의 하나의 부분 집합이다. 또한, $D_{1i_1} \times D_{2i_2} \times \cdots \times D_{ni_n}$로 나타내지는 데이터 블록을 $[i_1, i_2, \cdots, i_n]$와 같이 표현한다. BCPF는 스트링 $[i_1, i_2, \cdots, i_n]$의 요소 값이 0과 1의 값으로 표현된다.

(정의 1.2):

부분일치 질의어는 q: $(A_1 = a_1, A_2 = a_2, \cdots, A_n = a_n)$ 와 같이 표현되며, 여기서 는 i 번째 어트리뷰트 영역의 값이거나 미정의 값(*로 표현)이다.

(정의 1.3):

부분 일치 질의어 q에 대한 질의응답 집합(*query response set*), $R(q)$는 주어진 질의어에 대응하는 데이타 블록들의 집합이다. 예로서, 부분일치 질의어 q = [01**] 에 대한 질의 응답 집합은 다음과 같다.

$R(q) = \{0100, 0101, 0110, 0111\}$

(정의 1.4):

주어진 질의어 q에 대한 질의 응답 시간은 max $\{N_0, N_1, \cdots, N_{m-1}\}$로 정의한다. 여기서, $N_i(0 \le i \le m-1)$는 디스크 i에서 응답하는 데이터 블록의 수이다. 그림 1에서는 $N_0 = 1, N_1 = 1, N_2 = N_3 = \cdots = N_{m-2} = 0, N_{m-1}$이며, 질의 응답 시간은 N_{m-1}의 값에 의해 2가 된다.

(정의 1.5):

n개의 어트리뷰트를 가진 주어진 부분일치 질의어 q와, m개의 디스크로 구성된 파일 F에 대하여 상수 b_{max}를 식 (1)과 같이 정의한다.

$$b_{max} = \lceil 2^n / m \rceil \quad (1)$$

이때, max $\{N_0, N_1, \cdots, N_{m-1}\} \le b_{max}$를 만족하면, 그 데이터 배치 방식은 주어진 부분 일치 질의어 q에 대하여 최적화(*strictly optimal*) 되었다고 정의한다. 또한 모든 발생 가능한 부분 일치 질의어에 대하여 최적화되었으면, 그 데이타 배치 방식은 최적화되었다고 정의한다.

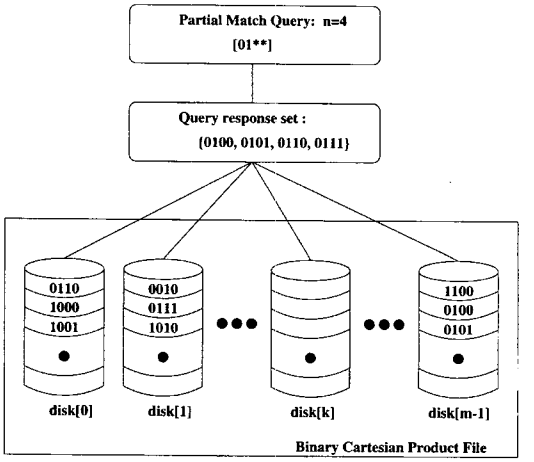


그림 1. 다중 디스크 데이터 배치 문제:예제
Fig. 1. Multidisk Data Allocation Problem:Example.

정의된 용어를 이용하여 다중 디스크 할당 문제를 그림 1을 이용하여 설명하면 다음과 같다. 사용자로부터의 입출력 요구는 제일 먼저 부분 일치 질의어의 형태로 시스템에 요구된다. 예로서, 그림 1에서는 어트리뷰

트의 수가 4일 경우의 부분 일치 질의어 [01**]가 요구되었다.

이 부분 일치 질의어에 응답되는 질의 응답 집합은 0100, 0101, 0110, 0111등 네 개의 버킷들이며, 이들은 m개의 디스크로 구축된 BCPF에서 읽혀지게 된다. 그림 1의 경우 버킷 0100과 0101이 디스크 $m-1$에 위치해 있기 때문에 질의 응답 시간은 2가 된다. 만약 0100, 0101 두개의 버킷 중 하나가 디스크 $k(2 \le k \le m-2)$에 배치되었다면 질의 응답 시간은 1이 된다. 따라서 다중 디스크 데이터 배치 문제는 정의된 용어를 이용하여 다음과 같이 형식화하여 요약할 수 있다.

주어진 조건:

n-attribute BCPF, 디스크 집합 $D=\{d_i \mid 0 \le i \le m-1\}$

구하고자 하는 답:

BCPF를 구성하는 버킷 집합 $B=\{d_i \mid 0 \le i \le 2^n-1\}$를 m개의 디스크에 다음의 목표를 만족시키며 배치하는 방법

목표:

부분일치 질의어에 대한 질의 응답 시간의 최소화

Ⅲ. 설계 개념

제안하는 다중 디스크 데이터 배치 방식의 설계 개념에 대한 설명은 버킷벡터(*bucketvector*)의 정의로부터 시작된다. 질의어의 어트리뷰트의 수가 n이고, 디스크의 수가 m 일 때, 하나의 버킷벡터는 $m \cdot b_{\max}$개의 정수들로 구성된 문자열(*string*)이다. 이 정수들은 0부터 2^n-1사이의 값을 가지는 2^n개의 음이 아닌 정수들과 $-(m \cdot b_{\max}-2^n)$과 -1 사이의 값을 가지는 $m \cdot b_{\max}2^n$개의 음의 정수들로 구성된다. 음이 아닌 정수 값은 디스크에 배치될 실제 데이터를 포함하는 버킷들을 의미하며 하나의 버킷벡터에서 특정 값의 정수는 오직 하나만 존재한다. 음의 정수들은 실제로는 존재하지 않는 가상 버킷을 의미하는데 그 용도는 추후 설명된다. 다음으로, 정의된 버킷벡터에 다음과 같은 디스크 사상 함수 $f: B \rightarrow D$ 를 정의한다.

$$f(b_i) = j \bmod ulo\ m \qquad (2)$$

집합 B는 버킷벡터의 구성 요소인 정수(=버킷)들의 집합이고, D는 배치될 디스크의 집합이며, 식 (2)에서 j는 버킷 b_i가 버킷벡터 내에서 위치하는 순서이다.

Position	0	1	2	3	4	5	6	7	8
Permutation	1	3	7	0	6	4	2	5	-1

그림 2. 버킷벡터 예 1
Fig. 2. Bucketvector example 1.

Position	0	1	2	3	4	5	6	7	8
Permutation	2	4	6	1	-1	0	3	7	5

그림 3. 버킷벡터 예 2
Fig. 3. Bucketvector example 2.

예로서m=3 이고 n=3일 때 그림 2와 같은 순열을 가지는 버킷벡터는 디스크 사상 함수를 적용했을 때 다음과 같은 디스크 배치 결과를 나타낸다.

- Disk 0에 배치되는 버킷 : 1, 0, 2
- Disk 1에 배치되는 버킷 : 3, 6, 5
- Disk 2에 배치되는 버킷 : 7, 4, -1

그림 3과 같은 순열의 버킷벡터의 경우는 다음과 같이 버킷들이 디스크들에 분산 배치 된다.

- Disk 0에 배치되는 버킷 : 2, 1, 3
- Disk 1에 배치되는 버킷 : 4, 7, -1
- Disk 2에 배치되는 버킷 : 0, 5, 6

한편, 음의 정수의 용도를 그림 2를 이용하여 설명하면 다음과 같다. 그림 2에서 만일 음의 정수가 없다면 어떤 순열의 버킷벡터에 대해서도 디스크 2에는 항상 2개만의 버킷이 할당된다. 이 이유는 버킷벡터의 위치 8에는 항상 빈 공간이기 때문에 디스크 사상 함수를 적용하면 위치 8에 해당하는 디스크 2에 빈 공간이 할당되기 때문이다.이때 빈 공간을 나타내는 음의 정수를 버킷벡터를 구성하는 하나의 요소로 정의하면 버킷벡터의 순열에 따라 빈 공간이 여러 디스크들 중의 임의의 디스크에 할당될 수 있다. 앞의 예에서 보는 바와 같이 서로 다른 순열의 버킷벡터는 디스크 사상 함수의 적용에 의해 서로 다른 디스크 배치가 됨을 알 수 있다. 즉, 버킷벡터와 디스크 사상함수를 도입하면 다중 디스크 데이터 배치 문제는 다음과 같이 변환 될 수 있다.

주어진 조건:

어트리뷰트 수 = n, 디스크의 수 = m

구하고자 하는 답:

다음의 정수 집합 B의 요소들로 구성되는 순열(*permutation*)들 중에서 다음의 목표를 만족시키는 하나의 순열

$$B=\{b_i|(0\le i\le m\cdot b_{\max}-1,-(m\cdot b_{\max}-2^n)\le b_i\le 2^n-1)\cap(b_j\neq b_k \text{ for } j\neq k)\}$$

목표:

순열의 판별 값(*fitness value*)을 최대화

본 고에서 사용되는 순열의 판별 값은 앞장에서 정의한 평균 질의 응답 시간을 이용하여 계산되며 다음의 식으로 표시 된다.

$$T_{avg}=\sum_{i=1}^{n-1}T_iP_i \tag{3}$$

식 (3)에서 T_i는 i개의 어트리뷰트가 미정의 값을 가진 모든 부분일치 질의어에 대한 응답시간을 평균한 값이고, P_i는 모든 발생 가능한 부분일치 질의어들에 대한 i개의 미정의 어트리뷰트 값을 가지는 부분일치 질의어들의 비율이다. 미정의 값을 가지는 어트리뷰트의 수가 i인 부분일치 질의어의 수는 $C_i=\binom{n}{i}$이므로, T_i는 식 (4)로 표현할 수 있다.

$$T_i=\frac{1}{C_i}\sum_{j=0}^{C_i}\max_j\{N_0,N_1,\cdots,N_{m-1}\} \tag{4}$$

한편, i=0인 경우는 질의 응답 시간이 항상 1이 되고, i=n인 경우는 항상 $b_{\max}$의 고정 값을 가지므로 두 가지의 경우는 제외하여 P_i를 식 (5)로 나타낸다.

$$P_i=\frac{C_i}{\sum_{k=1}^{n-1}C_k} \tag{5}$$

결국, 다중 디스크 데이터 배치 문제는 여러 발생 가능한 정수 순열 집합들 중에서 최적의 판별 값에 근사하는 하나의 순열을 찾는 검색(*searching*) 문제로 변환됨을 알 수 있다. 한편, 본 검색 문제의 복잡성(*complexity*)은 다음과 같이 계산된다. 정의된 버킷벡터와 디스크 사상 함수를 적용하면 서로 다른 순열을 가지는 버킷벡터의 평균 질의 응답시간은 서로 다를 수 있다. 그러나, 같은 디스크에 할당된 정수들 사이의 자리 바꿈은 순열의 판별 값에 영향을 미치지 않는다. 또한, 디스크 m_i에 배치된 정수들의 집합을 S_i라 하면 S_i사이의 자리 바꿈 판별 값을 바꾸게 하지 않는다. 따라서, $s(=b_{\max})$를 하나의 디스크에 배치되는 정수의 수라고 하고, $k(=m\cdot b_{\max}=m\cdot s)$를 하나의 순열을 구성하는 정수의 수라고 정의하면, 발생 가능한 순열의 수 $G_m(s)$는 다음과 같이 계산된다.

$$\frac{1}{m!}\cdot\binom{k}{s}\cdot\binom{k-s}{s}\cdot\binom{k-2s}{s}\cdots\binom{s}{s}=\frac{k!}{m!\cdot(s!)^m} \tag{6}$$

함수 $G_m(s)$의 특성은 $G_m(s+1)$의 표현식을 계산하고 두 함수의 비를 구하여 유추할 수 있는데, $G_m(s+1)$는 다음의 식으로 표현된다.

$$G_m(s+1)=\frac{\{(s+1)\cdot m\}!}{m!\cdot\{(s+1)!\}^m}=\prod_{i=1}^{m}\frac{i+m\cdot s}{s+1}G_m(s) \tag{7}$$

따라서, 두 함수의 비율은 다음과 같다.

$$\frac{G_m(s+1)}{G_m(s)}=\prod_{i=1}^{m}\frac{i+m\cdot s}{s+1} \tag{8}$$

식 (8)에서 순열을 구성하는 정수의 수를 무한대로 취하고 그 한계 값을 구하면 다음과 같은 결과를 얻는다.

$$\lim_{s\to\infty}\frac{G_m(s+1)}{G_m(s)}=\prod_{i=1}^{m}m=m^m \tag{9}$$

식 (9)에서 보는 바와 같이 다중 디스크 데이터 배치 탐색문제는 지수적으로 증가하는 복잡도를 가지는 알고리즘임을 알 수 있다. 따라서 이 문제를 풀기 위해서는 휴리스틱 알고리즘을 적용하여 근사 최적 값을 구해야 한다. 본 고에서는 휴리스틱 탐색 방식으로서 유전자 알고리즘을 이용한다.

Ⅳ. 다중 디스크 데이터 배치 방식 구현

3장에서 나타낸 바와 같이 다중 디스크 데이터 배치 문제는 버킷벡터와 디스크 사상 함수를 이용하여 지수 승의 복잡도를 가지는 탐색 문제로 변환된다. 본 연구에서는 탐색 문제를 풀기위한 도구로서 유전자 알고리즘을 이용한다. 유전자 알고리즘은 여러 분야의 최적화 문제에 성공적으로 적용되어 그 견고성이 입증되고 있으며, 최근에도 다양한 분야의 최적화 및 탐색 문제의 해결 방법으로 많은 연구가 진행되고 있다.

본 고에서 제안된 유전자 알고리즘을 이용한 알고리즘의 구조는 그림 4와 같다. 일반적인 유전자 알고리즘처럼 5개의 독립적인 단계인 초기화(*ini-*

tialization), 객체 평가(*evaluation*), 객체 재생성(*reproduction*), 유전자 교배(*crossover*), 유전자 돌연변이(*mutation*)로 구성된다.

```
Procedure GA()
begin
    Initialize Population;
    While termination criterion not reached do
    begin
        Evaluation;
        Reproduction;
        Crossover;
        Mutation;
    end;
end.
```

그림 4. 유전자 알고리즘 구조
Fig. 4. Structure of Genetic Algorithm.

알고리즘은 정해진 최대 반복 횟수 만큼 반복 수행되며, 최대 반복 횟수는 컴퓨터 수행 시간과 알고리즘의 수행 결과에 대한 질(*quality*)의 측면에서 상호 보완적으로 결정된다. 본 알고리즘이 반복 횟수가 무한대로 수행되었을 때, 최적의 해답을 구할 수 있는지에 대한 알고리즘의 수렴성은 다음 장에서 설명한다.

1. 객체의 문자열 표현(*String Representation*)과 초기화

유전자 알고리즘을 설계할 때 가장 우선적으로 정해야 하는 것은 최적화 하고자 하는 문제를 유한한 길이의 문자열(*string*)로 만드는 것으로서 문자열의 구현 방식은 최적화하고자 하는 문제의 종류에 따라 달라진다. 본 고에서는 다중 디스크 데이터 배치 문제에 대한 문자열 표현 방식으로서 3 장에서 정의한 버킷벡터를 이용한다.

초기화 과정에서는 초기 객체군(*Initial population*)이 형성되는데, 독립적으로 만든 p개의 버킷벡터 집합, $P=\{p_i|0\le i\le p-1\}$로 구성된다. 구하고자 하는 최적의 디스크 배치 방식은 초기 객체군들에 반복적으로 유전자 오퍼레이타를 적용하여 만들어 지며, 초기 개체군의 크기 p는 구하고자 하는 문제의 크기에 따라 결정되는 값으로서 실험을 통해 정해진다.

2. 객체 평가

이 과정에서는 객체군을 형성하는 각 버킷벡터의 우수성이 판별함수를 이용하여 측정된다. 본 논문에서는 3장에서 정의한 평균 응답 시간, T_{avg}를 이용하여 판별함수를 정의한다. 그러나 다중 디스크 데이터 배치 문제의 근본적인 목적은 평균 질의 응답 시간을 최소화 시키기 위한 문자열을 찾는 반면, 유전자 알고리즘의 한 오퍼레이타인 재생성은 판별값을 최대화시키는 방향으로 전개되기 때문에, 유전자 알고리즘에 적용하기 위하여 식 (10)과 같은 판별 함수를 이용한다.

$$Fitness(p_i)=\frac{1}{T_{avg}(p_i)} \tag{10}$$

즉, 평균 응답시간의 값이 작을수록 판별함수의 값이 커지게 한다.

3. 객체 재생성(*Reproduction*)

재생성 오퍼레이타는 현 세대의 객체군의 판별 값을 기반으로 하여 다음 세대에 이용될 새로운 버킷벡터 집합을 만드는 기능을 담당한다. 이 오퍼레이타는 자연계의 적자생존의 법칙을 모형화한 것으로서, 우수한 판별 값을 가지는 버킷벡터는 번성하고 열등한 버킷벡터는 멸하게 하여 세대(*generation*)가 흐름에 따라 우수한 객체가 생존할 가능성을 높여주는 역할을 한다. 본 연구에서는 BRW(*biased roullet wheel*) 방식[13]을 적용하여 구현하였다. 이 방식의 첫 단계에서는 하나의 원형 룰렛판을 각 버킷벡터의 판별 값에 비례하도록 원주각을 배분하여 분할한다. 다음 단계에서는 0부터 2π사이의 값 중에서 임의의 값을 랜덤 생성기를 이용하여 만든다. 만일 생성된 값이 첫 단계에서 만들어진 원형 룰렛판의 버킷벡터 p_i에 할당된 원주각 영역에 해당되면 p_i를 다음 세대에 사용될 새로운 객체군에 포함 시킨다. 또한 알고리즘의 시뮬레이션 시간을 줄이기 위하여 현 세대에서 다음 세대로 옮겨갈 때, 가장 우수한 객체인 *BEST-POP*은 반드시 다음 세대를 위한 객체군에 포함되도록 한다. 위와 같은 과정은 다음 세대를 위한 객체군이 모두 만들어질 때까지 반복해서 수행된다.

4. 유전자 교배

유전자 교배는 2 개의 객체가 각 유전자의 일부를 상호 교환하여 새로운 2개의 객체를 만들어내는 유전자 오퍼레이타이다. 본 문제의 특성을 보면, 각 객체에 대한 판별함수가 객체의 유전자 배열 순서에만 영향을 받음을 알 수 있다. 이러한 문제에 적합한 유전자 교배 오퍼레이타가 PMX(*Partially matched crossover*) 오퍼레이타이다[13]. PMX 유전자 교배에서는 조건 $0\le i<j\le 2^n-1$을 만족하는 두개의 유전자 교배점 i, j를 랜덤하게 생성하고, i와 j 위치 사이에 있는

두 버킷벡터의 유전자들을 위치별로 교환함으로써 이루어 진다.

Position	0	1	2	3	4	5	6	7
bucketvector1	1	3	7	0	6	4	2	5
bucketvector2	0	7	3	6	1	5	2	4

그림 5. 유전자 교배 이전의 버킷벡터
Fig. 5. bucketvecters before crossover.

Position	0	1	2	3	4	5	6	7
bucketvector1	6	3	7	0	1	5	2	4
bucketvector2	0	7	3	1	6	4	2	5

그림 6. 유전자 교배 이후의 버킷벡터
Fig. 6. bucketvecters after crossover.

예로서, 그림 5에서 보는 바와 같이 버킷벡터 1, 2에 대하여, 유전자 교배 영역이 $s_1=4$와 $s_2=5$로 정해졌을 때를 가정하여 보면, 그림 6과 같은 새로운 두 객체가 만들어 진다. 즉, 버킷벡터 1의 유전자 6, 4와 버킷벡터 2의 유전자 1, 5를 상호 교환한 후, 버킷벡터 1에서는 유전자 1을 6으로, 유전자 5를 4로 교체하고, 버킷벡터 2에서는 유전자 6을 1로, 유전자 4를 5로 교체한다. 이와 같은 유전자 교배 후에는 그림 6과 같은 새로운 두 개의 버킷벡터가 만들어 지는데, 새로이 만들어진 두 개의 버킷벡터는 유전자 교배 이전의 두 버킷벡터에 포함되어 있었던 유전 정보들을 부분적으로 포함하게 된다.

유전자 교배는 *P-CROSS*라는 유전자 교배 확률 변수에 의해 제어된다. 유전자 교배의 대상으로 2개의 버킷벡터가 선정된 후, 랜덤 생성기를 이용하여 생성된 0과 1사이의 임의 실수 값이 *P-CROSS*보다 작을 경우에만 유전자 교배가 이루어진다. *P-CROSS*의 일반적인 값은 0.5와 0.8 사이의 값을 가지며 실험을 통하여 결정된다.

5. 유전자 돌연변이

유전자 돌연변이 오퍼레이타는 하나의 객체 내에서 임의의 유전자를 변형시켜서 새로운 객체를 만들 수 있게 하여 알고리즘이 지역최소치함정(*local minima's trap*)에서 벗어날 수 있게 도와준다. 제안된 유전자 알고리즘의 유전자 돌연변이 단계에서는 객체군이 p개의 버킷벡터로 구성되었을 때 $2^n \cdot p$개의 단위 돌연변이 오퍼레이터를 적용시킨다. 하나의 단위 돌연변이는 버킷벡터내에서 두개의 돌연변이 위치에 있는 유전자 값을 상호 교환함으로써 이루어 진다. 단위 돌연변이는 *P-MUT*라는 돌연변이 확률 변수 값에 의해 제어되며 이 값도 실험에 의해 결정된다. 그림 7은 하나의 유전자 돌연 변이 과정을 나타낸다.

Algorithm 유전자돌연변이

M0.: *POP-POS* ← 0.
M1.: Do step M2-M9 p times.
M2.: *BucketVector* ← population[*POP-POS*].
M3.: *MUT-POS* ← 0.
M4.: Do step M5-M8 2^n times for *BucketVector*.
M5.: *MUT-COIN* ← a random value in the range [0, 1].
If *MUT-COIN* > *P-MUT* goto step M8,
M6.: Select mutation site j randomly,
such that $0 \le j \le 2^n - 1$, (j *mod* m). ≠ (*MUT-POS* *mod* m).
M7.: Exchange the buckets positioned at j and *MUT-POS*
M8.: *MUT-POS* ← *MUT-POS* + *1*
M9.: *POP-POS* ← *POP-POS* +*1*

그림 7. 돌연변이 알고리즘
Fig. 7. Mutation Algorithm.

6. 전체 알고리즘

앞 절에서 설계된 기능들을 이용한 전체 알고리즘의 구성은 그림 8과 같다

V. 알고리즘의 수렴성 분석

본 장에서는 4장에서 설명한 다중 디스크 데이타 배치 유전자 알고리즘의 수렴성을 분석한다. 수렴성 분석의 궁극적인 목적은 제안된 알고리즘이 최적의 다중 디스크데이터 배치 결과를 높은 확률로 도출할 수 있는 지에 대한 여부를 증명하는 것이다.

1. 수렴성 분석 방법

유전자 알고리즘의 수행 과정을 수학적으로 표현하고자 하는 것은 현재 수행되고 있는 중요한 연구 분야의 하나이지만 아직까지 확실한 수학적인 모델은 제안되지 않고 있다. 그러나 스키마 이론[13]은 현재까지 제시된 모델 중에서 유전자 알고리즘의 동작의 핵심을 가장 잘 표현하고 있는 것으로 알려져 있다. 스키마는 문자열의 특정 위치에 있는 값들이 같은 모든 문자열

들의 대표모형(*similarity template*)이다. 예를 들어 스키마 **010은 문자열 집합 00010, 01010, 10010, 11010 들을 대표한다. 한편, 스키마 이론에 이용된 스키마에 관련한 몇 가지 정의는 다음과 같다. 하나의 스키마 H에 대하여

Algorithm 다중디스크데이타배치

G1.: Population Initialization.
G2.: Do step G3-G9 until the algorithm is convergent.
G3.: Map buckets of each population to disks using **disk mapping function** *f*.
G4.: Compute fitness value of each string by appling **fitness function** *Fitness*.
G5.: *BEST-POP* ← string with the highest fitness value.
G6.: Call **Reproduction**.
G7.: Do step G7.1-G7.3 NPOP/2 times.
G7.1.: Pick two strings randomly from *PARENT-POP*.
G7.2.: Call **Crossover**
G7.3.: Put the created new strings in *CHILD-POP* and remove from *PARENT-POP*.
G8.: Call **Mutation**
G9.: Copy *CHILD-POP* to *PARENT-POP*

그림 8. 다중 디스크 데이타 배치 알고리즘
Fig. 8. Complete Algorithm

- 고정점(*fixed position*)은 스키마에서 값이 정의된 위치이다. (예) 스키마 1**0에 대한 고정점은 첫번째('1')와 4번째('0') 위치이다.
- 스키마의 오더(*order*)는 $o(H)$로 나타내며, 스키마내에서 값이 정의된 고정점(*fixed position*)의 수이다. (예) 스키마 H = **1**0 에서 $o(H)$는 2이다.
- 정의 길이(*defining length*)는 스키마에서 고정자의 첫번째 위치와 최후 위치의 위치값의 차이로서 로 나타낸다. (예) 스키마 H = **1**0 에 대한 의 값은 3이다.

스키마 이론에 의하면 유전자 알고리즘에서는 짧은 정의 길이(*short defining length*), 작은 스키마 오더(*low order*), 평균 판별 값 이상의 판별 값을 가지는 스키마들이 연속되는 세대 생성 반복 과정에서 지수적으로 증가되면서 궁극적으로 최적의 결과를 구해낸다는 것이다. 따라서 어떤 유전자 알고리즘이 무한한 세대 생성 과정을 통하여 궁극적으로 최적의 답을 만들어 낼 수 있는 지의 여부는 그 유전자 알고리즘의 동작이 스키마 이론을 만족시키는가를 증명함으로써 판별될 수 있다. 본 고에서는 다음의 3 단계의 과정을 통하여 제안된 유전자 알고리즘의 수렴성을 증명한다.

(단계 1):

버킷벡터에 대한 스키마 *b_schema*를 정의한다.

(단계 2):

제안된 유전자 알고리즘의 재성성, 유전자 교배,유전자 돌연변이 오퍼레이타가 주어진 객체군에 포함된 *b_schemata*들에 주는 영향을 조사한다.

(단계 3):

앞 단계에서 조사된 *b_schmata*에 대한 영향이 스키마 이론과 일치하는지를 비교함으로써 제안된 유전자 알고리즘의 수렴성을 증명한다.

2. 수렴성 증명

증명의 첫 단계로 *b_schema*를 다음과 같이 정의한다. *b_schema*는 주어진 구성 n 과 m에 대하여 *b_schema*의 각 위치 값이 $-(m \cdot b_{\max} - 2^n)$과 $2^n - 1$사이의 정수 값이거나 특수 문자 * 로 표시되며, 스키마의 길이 l은 이다. 예를 들어 버킷벡터1= 2 -1 0 3 4 1 5 6, 버킷벡터2=0 -1 2 3 4 5 6 1이 주어졌을 경우에, 스키마 * -1 * 3 4 * * * 은 두 버킷벡터들의 *b_schema*가 될 수 있다.

두 번째 증명 단계인 유전자 오퍼레이터의 *b_schema*에 미치는 영향은 다음과 같다. 첫째로, 재성성이 스키마에 미치는 영향을 나타내기 위하여 임의의 시간 t에서, 객체군 내에 특정 스키마 H에 해당하는 객체의 수가 $m(H,t)$라고 하고, 해당 객체들의 판별 함수 값의 평균치를 $f(H)$로 가정한다. 만약 를 객체군 전체의 평균 판별 함수 값이라고 하면, 재생성기를 거친 후의 스키마 H에 해당하는 객체의 수인 $m(H,t+1)$은 BRW 방식의 구현 특성에 의해 식 (11)과 같이 정해진다.

$$m(H, t+1) = m(H, t) \cdot \frac{f(H)}{f} \tag{11}$$

한편, 유전자 교배 단계에서는 2개의 교배 점이 임의로 선정되고, 두 교배 점 사이의 유전자가 교환된다. 이때 만일 두 교배 점 사이에 고정 값이 있게 되면 교배전의 스크마는 없어지게 되므로 유전자 교배 후에 스키마가 계속 살아 있으려면 두개의 교배 점이 동시에 정의 길이 영역의 왼쪽이거나 오른쪽에 위치하여야 한다.

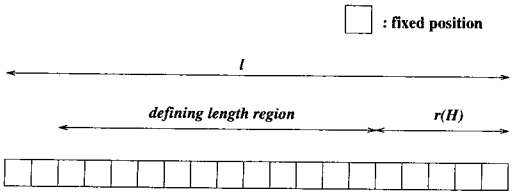


그림 9. *b_schema H* 표본
Fig. 9. A Sample *b_shema H*.

그림 9에서 보는 바와 같이 $r(H)$를 스키마의 오른쪽 끝에서 가장 오른쪽에 있는 고정 점의 위치 사이의 거리로 정의하면 유전자 교배 후에 스키마 H가 생존할 확률은 유전자 교배 확률 를 고려하여 식 (12)와 같이 표현된다.

$$p_c \cdot \left(\left(\frac{r(H)}{l} \right)^2 + \left(\frac{l - r(H) - \delta(H)}{l} \right)^2 \right) \quad (12)$$

식 (12)에서 덧셈의 첫 항목은 두개의 교배 점이 정의 길이 영역의 오른쪽에 있을 확률이고, 두 번째 항목은 왼쪽에 있을 확률이다. 유전자 돌연변이 단계에서 하나의 단위 유전자 돌연변이는 대상의 버킷벡터에 대하여 두개의 돌연변이점 $i, j((i \bmod m) \neq (j \bmod m))$간의 유전자 위치 바꿈을 통하여 수행된다. 이와 같은 단위 유전자 돌연변이에 의해 스키마가 변형되려면 다음의 두 가지 경우가 만족되어야 한다. 첫째, 두개의 돌연변이점 중 최소한 하나가 고정점 위치일 경우로서, 이 확률은 다음과 같다.

$$2 \cdot \frac{o(H)}{l} \cdot \left(1 - \frac{o(H)}{l}\right) + \left(\frac{o(H)}{l^2}\right) = \frac{2 \cdot o(H)}{l} - \left(\frac{o(H)}{l^2}\right) \quad (13)$$

식 (13)에서 첫째 항은 두 점 중의 하나가 고정점일 확률이며, 두 번째 항은 두 점이 모두 고정점일 확률이다. 두 번째로는 선택된 두 돌연변이점이 같은 디스크에 해당되지 않아야 한다는 것이다. 하나의 객체에서 같은 디스크에 해당하는 지점은 최대 $k(=b_{\max})$개 이므로, 같은 디스크에 해당하지 않을 확률은 $\frac{l-k}{l-1}$ 이다. 따라서, 유전자 돌연변이 이후 스키마가 살아 있을 확률은 유전자 돌연변이 확률 p_m을 적용하여 다음과 같이 정해진다.

$$1 - p_m \cdot \frac{l-k}{l-1} \cdot \left(\frac{2 \cdot o(H)}{l} - \left(\frac{o(H)}{l^2} \right) \right) \quad (14)$$

하나의 유전자 돌연변이 과정에서 위와 같은 단위 돌연변이가 $l \cdot p$개 만큼 독립적으로 일어나므로 하나의 유전자 돌연변이 수행 후 스키마 H가 생존할 확률은 다음과 같다.

$$\left(1 - p_m \cdot \frac{l-k}{l-1} \cdot \left(\frac{2 \cdot (H)}{l} \right) - \left(\frac{o(H)}{l^2} \right) \right)^{l \cdot p} \quad (15)$$

식 (15)에서 아주 작은 값의 p_m(<< 1) 에 대하여 다음과 같이 근사값을 취할 수 있다.

$$\left(1 - p_m \cdot l \cdot p \cdot \frac{l-k}{l-1} \cdot \left(\frac{2 \cdot o(H)}{l} - \left(\frac{o(H)}{l} \right)^2 \right) \right) \quad (16)$$

지금까지 구한 재생성기, 유전자 교배, 유전자 돌연변이의 효과를 하나로 묶어 낮은 차수에 대하여 정리하면 다음과 같다.

$$m(H, t+1) \geq m(H, t) \cdot \frac{f(H)}{f} \cdot \left[1 - p_c \frac{2}{l} \delta(H) - p_m \frac{2p(l-k)}{l-1} o(H)\right] \quad (17)$$

식 (17)이 의미하는 것은 *b_schema H*는 전체 평균 판별 함수값에 대한 스키마의 판별 함수 값의 비율에 따라 지수적으로 증가하며, 스키마 정의 길이 $\delta(H)$의 크기에 반비례 하고, 스키마 오더 $o(H)$의 크기에 반비례 하여 증가한다는 것으로서 이는 스키마 이론과 일치함을 나타낸다. 따라서 제안된 유전자 알고리즘은 매우 높은 확률로 최적의 다중 디스크 할당 방식을 찾아낼 수 있다.

VI. 시뮬레이션 연구

유전자 알고리즘을 설계할 때는 알고리즘이 제공하는 해답에 대한 질과 알고리즘의 수행 효율성이 고려되어야 한다. 결과에 대한 질은 알고리즘으로부터 구해진 결과가 목표 값에 얼마나 근접하고 있는지에 대한 척도이며, 효율성은 답을 구하기 위해 소요된 시뮬레이션 시간 및 소요된 메모리의 크기 등이다. 유전자 알고리즘에서 답에 대한 질과 효율성에 영향을 주는 요소는 유전자 오퍼레이타의 수행 알고리즘외에 객체군의 크기(*population size*), 유전자 교배, 유전자 돌연변이 확률, 세대 생성수(*number of generation*) 등이 있다. 이러한 요소들의 적정 값은 실험을 통하여 정해지며, 해답에 대한 질과 효율성의 상호 보완적인 절충(*trade-off*) 과정을 거친다. 본 장에서는 위의 같은 시뮬레이션 변수들의 적정 값을 결정하기 위한 한 실험 예를 설명하고, 그 실험값을 이용하여 구해진 제

안된 알고리즘의 성능 결과와 GDM, BDM, ECC 방식과의 성능 비교 결과에 대하여 기술한다.

1. 시뮬레이션 변수 결정

시뮬레이션의 변수를 결정하기 위한 실험의 한 예로서, 디스크의 수(m)가 8이고 어트리뷰트의 수(n)가 5인 구성에 대한 실험 결과를 그림 10과 그림 11에 나타내었다. 변수 값들의 상호 관계를 구하기 위해서는 하나의 변수 값을 고정하여야 하는데 이를 위하여 유전자 교환 확률(*P-CROSS*)을 일반적으로 많이 적용하는 0.6으로 고정하였다. 또한, 유전자 알고리즘은 비결정적(*non-deterministic*)이기 때문에 하나의 구성에 대하여 16개의 독립적인 시뮬레이션을 수행하였으며, 16 결과값의 평균치를 결과 데이터로 정하였다. 각 각의 독립적인 시뮬레이션은 서로 다른 랜덤 발생 종자(*seed*)를 이용하여 초기 객체군, 유전자 교환, 돌연 변이를 발생하여 수행하였다.

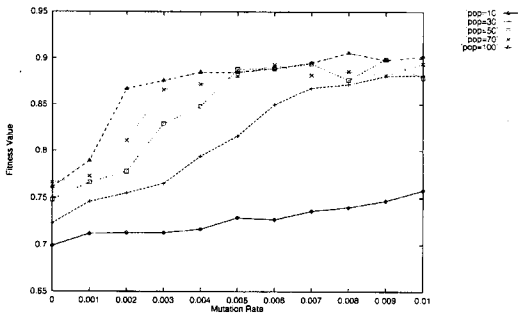


그림 10. 객체군의 수와 유전자 돌연변이율에 대한 판별 값의 변화 (800 generation)
Fig. 10. Effects of mutation rate and population size.

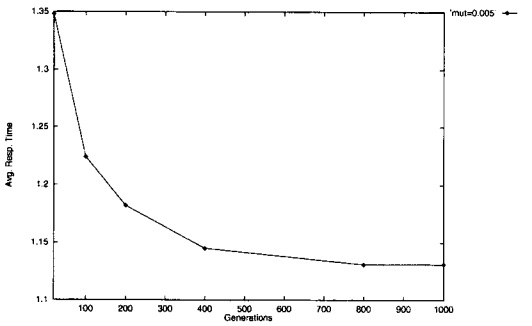


그림 11. 세대 생성 수에 대한 판별 값의 변화 (50 population)
Fig. 11. Effects of generations.

그림 10은 세대 생성 수를 800으로 고정시킨 상태에서 객체군의 크기의 변화에 대한 영향을 보여주는 그림으로서, 객체군의 크기가 클수록 더 좋은 결과를 보여준다. 객체군의 크기가 50보다 커지면 결과의 차이가 작아지며, 유전자 돌연변이율이0.005 보다 커지면 객체군의 크기가 50 이상의 모든 경우에 대하여 그 결과 값의 차이가 크게 줄어듬을 알 수 있다. 본 실험의 목적은 모든 구성에 대하여 적용될 수 있는 돌연변이율을 유추해 보는 것인데, 그림 10에서 보면 돌연변이율이 0.005에서 0.006 사이에서 객체군의 크기에 관계 없이 비슷한 결과를 얻을 수 있다. 시스템의 구성이 커지면 초기 객체군의 크기도 커지게 되는데, 이러한 변화에도 영향을 크게 받지 않는 유전자 돌연변이율인 0.005를 *P-MUT*의 값으로 정하여 실험을 수행 하였다. 그림 11은 객체군의 크기를 50으로 고정시키고 세대 생성수를 변화시킬 때의 결과를 보여준다. 유전자 돌연변이율이 0.005 이하일 때는 세대 생성수 200 이하에서 지역최적치에 수렴되어서 세대 생성수의 변화에 대한 효과가 잘 나타나지 않았다. 그림 11에서 보는 바와 같이, 유전자 돌연변이율이 0.005 일 경우는 세대 생성수가 커질수록 좋은 결과를 얻을 수 있으며, 800 보다 커지면 결과치의 변화가 급격히 감소함을 알 수 있다. 본 실험에서 최대 세대 생성수는 800 보다 크게 설정하여 수행하였다.

2. 성능 비교

제안된 디스크 할당 알고리즘의 성능을 기존의 타 방식과 비교하기 위하여 여러 가지 조합의 n, m에 대하여 실험을 수행하였다. 본 실험 과정에서 사용된 실험 변수는 앞 절에서 설명한 실험을 통하여 정해졌으며 표 1과 같다.

표 1. 시뮬레이션 파라미터
Table 1. Simulation parameters.

n	pop. size	*P-CROSS*	*P-MUT*	max. gen
5	50	0.6	0.005	1500
6	100	0.6	0.005	3500
7	100	0.6	0.005	5000

첫번째 실험으로GDM 방식과의 성능 비교를 수행하였는데, GDM 방식은 디스크의 수(m)에 제한을 가하지 않은 방식 중에서는 가장 좋은 성능을 제공하는 것으로 알려져 있기 때문에 비교대상으로 선정하였다. GDM 방식은 버킷벡터 $[i_1, i_2, \cdots, i_n]$를 디스크

$d=(\sum_{j=1}^{j=n} \times p_j) \bmod m$에 할당한다. 여기서 p_j는 디스크 수 m에 소수(*prime number*)인 양의 정수 값으로서, 참고문헌 [5]에 따르면, 최적의 를 구하기 위한 일반적인 방법은 없으며 매우 어려운 문제라고 기술하고 있다. 이러한 사실은 GDM 또한 비결정적인 알고리즘이라는 것을 의미한다. 따라서, 본 실험에서는 공평한 성능 비교를 위하여 하나의 GDM 구성에 대하여 100가지의 p_j조합을 생성하고, 각각의 구성에 대한 성능 결과를 구하였다. 그리고 그 중에서 가장 우수한 결과를 그 구성에서의 GDM 성능이라고 정하였다. 표 2에서 표 4는 그 실험 결과를 보여주고 있는데, 표에서 T_{OPT}는 이론적인 최적의 질의 응답 시간을 의미하고, T_{GA}, T_{GDM}은 각각 제안된 방식과 GDM 방식으로부터 구해진 질의 응답 시간이다. 제일 끝 난은 GDM 방식에 비하여 제안된 방식의 상대적인 성능 개선율을 나타낸다. 결과표에서 보는 바와 같이 제안된 방식은 GDM 방식과 비교하여 항상 좋은 결과를 제공하고 있으며, 최대 35% 의 성능 개선율을 보여준다.

표 2. n=5일 때 GDM 방식과의 성능 비교 결과

Table 2. GA Versus GDM method for n=5.

No. of Disks	T_{OPT}	T_{GA}	T_{GDM}	$\frac{T_{GDM}-T_{GA}}{T_{GDM}}$
2	2.2857	2.2857	2.2857	0
4	1.33	1.4095	2.0	0.30
6	1.2857	1.3286	2.0	0.34
8	1.0476	1.1048	1.4857	0.26
10	1.0476	1.0857	1.4857	0.27
12	1.0476	1.0714	1.4857	0.28
14	1.0476	1.0667	1.2571	0.15
16	1.0	1.0095	1.1429	0.12

두 번째 실험에서는 ECC 방식과 BDM 방식과의 성능 비교를 수행하였다. 앞에서 설명한 바와 같이 두 방식은 적용되는 디스크의 수가 2의 지수승이어야 한다는 제한 사항을 가지고 있다. ECC 방식은 디스크의 수에 대한 제한을 가지고 제안된 모든 방식들 중에서 가장 우수한 성능 제공하고, BDM 방식은 이에 버금가는 것으로 알려져 있다. 표 5는 디스크의 수에 대한 제한 조건을 부가한 상태에서 비교한 성능을 보여준다. 표 5의 ECC와 BDM의 응답 시간은 참고 문헌 [7]의 표 III에 제시된 결과를 인용하였다. 표 5에서 보는 바와 같이 제안된 방식은 ECC 방식에는 미치지 못하지만 BDM 방식과는 비슷한 결과를 제공함을 알 수 있다. 이러한 고무적인 결과는 제안된 방식이 디스크의 수에 대한 제한이 있는 경우와 없는 경우에 관계없이 사용될 수 있는 우수한 방법임을 보여준다.

표 3. n=6일때 GDM 방식과의 성능 비교 결과

Table 3. GA Versus GDM method for n=6.

No. of Disks	T_{OPT}	T_{GA}	T_{GDM}	$\frac{T_{GDM}-T_{GA}}{T_{GDM}}$
4	1.6386	1.7651	2.4578	0.28
6	1.5120	1.6054	2.4578	0.35
8	1.1446	1.2681	1.7349	0.27
10	1.1446	1.2575	1.7349	0.28
12	1.1265	1.2063	1.7349	0.30
14	1.1265	1.1792	1.3976	0.16
16	1.0180	1.0873	1.2891	0.16

표 4. n=7일때 GDM 방식과의 성능 비교 결과

Table 4. GA Versus GDM method for n=7.

No. of Disks	T_{OPT}	T_{GA}	T_{GDM}	$\frac{T_{GDM}-T_{GA}}{T_{GDM}}$
4	2.0680	2.4577	3.0476	0.19
6	1.8163	2.0092	3.0476	0.34
8	1.3061	1.6534	2.0758	0.20
10	1.2993	1.5466	2.0719	0.25
12	1.2517	1.4310	2.0758	0.31
14	1.2449	1.3469	1.6443	0.18
16	1.0612	1.3131	1.4538	0.10
18	1.0612	1.2711	1.6443	0.23
20	1.0612	1.2289	1.4538	0.15
22	1.0544	1.1973	1.3217	0.09
24	1.0544	1.1696	1.4538	0.20
26	1.0544	1.1438	1.2459	0.08
28	1.0544	1.1239	1.1448	0.02
30	1.0544	1.1142	1.4519	0.23
32	1.0068	1.0981	1.1292	0.03

표 5. ECC, BDM 방식과의 성능 비교 결과

Table 5. GA Versus ECC, BDM methods.

Config.	T_{OPT}	T_{ECC}	T_{BDM}	T_{GA}
(5,8)	1.048	1.086	1.295	1.105
(6,8)	1.145	1.193	1.398	1.268
(7,8)	1.306	1.361	1.655	1.653
(8,16)	1.142	1.213	1.539	1.537

Ⅷ. 결 론

본 논문에서는 다중 디스크 시스템에 Binary Cartesian Product File을 배치시키는 새로운 데이터 배치 방법을 제안하였다. 먼저 다중 디스크 데이터 배치 문제를 탐색 문제로 변환하였고 지수적으로 증가하는 복잡도를 가지는 문제임을 증명하였다. 이 문제를 풀기위한 휴리스틱 알고리즘으로서 유전자 알고리즘을 이용한 새로운 방법을 제시하였으며, 스키마 이론을 이용하여 제안된 방식이 매우 높은 확률로 최적에 가까운 데이타 배치 방식을 찾아낼 수 있음을 증명하였다. 시뮬레이션을 통하여 제안된 방식과 GDM, BDM, ECC 방식과의 성능 비교를 수행한 결과 디스크의 수에 대한 제한을 제거하면서도 우수한 성능을 제공함을 알 수 있었다. 비록 모든 경우에 대하여 이론적인 최적치는 만족시키지 못하지만 제안된 방식이 GDM 방식보다는 항상 우수한 성능을 제공하고, 디스크의 수에 대한 제한을 가한 상태에서도 ECC 방식에는 미치지 못하지만, BDM 방식과는 비슷한 성능을 제공함을 보여주었다. 이러한 실험 결과를 통하여 제안된 방식이 디스크의 수에 대한 제한이 있는 경우와 없는 경우에 관계 없이 사용될 수 있는 우수한 방법임을 알 수 있었다.

데이터 베이스 시스템에서 입출력 성능이 시스템의 성능을 결정하는 중요한 요소이며, 입출력 성능은 구축된 파일 시스템의 성능에 의해 결정된다. 따라서, 본 고에서 제안된 알고리즘은 다중 디스크로 구성된 데이터 베이스 시스템에 파일 시스템을 구축할 때, 데이터 블록들을 디스크들에 효과적으로 분배 시키는 도구로 사용됨으로써, 궁극적으로는 시스템의 성능을 향상시킬 수 있는 해법으로 적용될 수 있다.

향후, 보완되어야 할 연구 과제는 더 큰 n, m의 구성에 대한 알고리즘을 설계하는 것이다. n의 값이 큰 구성의 경우, 알고리즘이 지역최소치 함정에 수렴되지 않도록 초기 객체군의 수를 크게 설정하여야 한다. 이것은 알고리즘의 수행 시간이 크게 증가함을 의미하는데, 본 알고리즘을 컴퓨터에서 실험하는 과정에서 조사된 바에 따르면, 각 객체의 판별값(*fitness value*)인 평균 응답 시간을 계산하는데 상당히 많은 CPU 시간을 소비되는 것으로 밝혀졌다. 따라서, 하이퍼큐브와 같은 분산 메모리 MIMD 시스템에서 운용될 병렬 유전자 알고리즘에 관한 연구가 보완되어야 할 연구 과제로 남아 있다.

참 고 문 헌

[1] M. Y. Kim, "Synchronized disk interleaving," IEEE Trans. on Computers, vol. C-35, no. 11, Nov. 1986.

[2] D. A. Patterson, G. Gibson, and R. H. Katz, "A case for redundant array of inexpensive disks(RAID)," ACM SIGMOD Conf., Chicago, Illinois, June, 1988.

[3] L. N. Reddy and P. Banerjee, "An evaluation of multiple disk I/O systems," IEEE Trans. on Computers, vol. 38, no. 12, pp. 1680-1690, Dec. 1989.

[4] Y.Y. Sung, "Parallel searching for binary Cartesian product files," *Proc. ACM Commun. CSC 1985 Conf.*, pp. 163-172, Mar. 1985.

[5] H. C. Du and J. S. Soboleski, "Disk Allocation ffor Cartesian product files," ACM Trans. on Database Systems, vol. 7, no. 1, pp. 82-101, March 1982.

[6] H. C. Du, "Disk Allocation methods for binary Cartesian Product files," BIT, vol. 26, pp. 138-147, 1986.

[7] Faloutsos and D. Metaxas, "Disk Allocation Methods Using Error Correcting Codes," IEEE Trans. on Computers, vol. 40, no. 8, pp. 907-914, August 1991.

[8] M. H. Kim and S. Pramanik, "Optimal file distribution for partial match retrieval," Proc. 1988 ACM-SIGMOD Int. Conf. on Management of Data, pp. 173-182, 1988.

[9] J. U. Kim and T. G. Kim, "Multidisk partial match file design with known access pattern," Information Processing Letter, vol. 45, no. 1, pp. 33-39, jan. 1993.

[10] E. Goldberg and R. Lingle, "Alleles, loci, and the traveling salesman problem," Proc. Int. Conf. on Genetic Algorithms and Their Applications, pp. 154-159, 1985.

[11] C. Chang and C. Y. Chen, "Performance Analysis of the Generalized Disc Modulo Allocation Method for Multiple Key

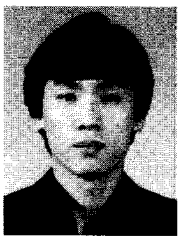
Hashing Files on Multi-disk Systems," The Computer Journal, vol. 30, no. 6, pp. 535-540, 1987.

[12] K.A.S. Abdel-Ghaffar and A.E. Abbadi, "Optimal Disk Allocation for Partial Match Queries," ACM Trans. on Database Systems., vol. 18, no. 1, pp. 132-156, March 1993.

[13] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and machine Learning, Addison-Wesley, 1989.

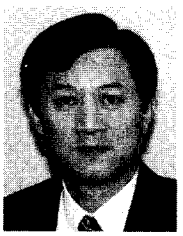
[14] M. Srinivas and L.M. Patnaik, "Genetic Algorithms: A Survey," IEEE Computer, pp. 17-26, June 1994.

저 자 소 개



安 大 榮(正會員)

1962년 2월 20일생. 1984년 한양대학교 전자공학과 졸업. 1986년 한국과학기술원 전기 및 전자공학과(공학석사). 1992년 ~ 현재 한국과학기술원 전기 및 전자 공학과 박사과정. 현재 한국전자통신연구원 컴퓨터구조 연구실 선임연구원. 주관심분야는 병렬처리컴퓨터 구조, 병렬 I/O 시스템, 시스템 성능분석



林 基 郁(正會員)

1950년 8월 22일생. 1977년 인하대학교 공과대학 전자공학과 졸업. 1986년 한양대학교 대학원 전자계산학 석사. 1994년 인하대학교 대학원 전자계산학 박사. 1977년 ~ 1983년 한국전자기술연구소 선임연구원. 1983년 ~ 1988년 한국전자통신연구소 시스템소프트웨어 연구실장. 1988년 ~ 1989년 미 캘리포니아 주립대학(Irvine) 방문연구원. 1989년 ~ 1997년 한국전자통신연구원, 시스템연구부장. 1989년 ~ 1997년 주전산기(타이컴) Ⅲ, Ⅳ 개발 사업책임자. 1997년 ~ 현재 정보통신연구관리단 전보기술전문위원. 주관심분야는 소프트웨어 아키텍쳐, 실시간데이타베이스 시스템, 컴퓨터시스템구조

朴 圭 晧(正會員)

1950년 10월 19일생. 1973년 서울대학교 전자공학과 졸업. 1975년 한국과학기술원 전기 및 전자공학과(공학석사). 1983년 프랑스 파리 11대학(공학박사). 1975년 ~ 1978년 동양정밀 개발과장. 1983년 ~ 현재 한국과학기술원 전기 및 전자공학과 교수. 주관심분야는 병렬처리, 컴퓨터 구조, 컴퓨터 비젼