LETTER

A Quantum-Inspired Evolutionary Computing Algorithm for Disk Allocation Method

Kyung-Ho KIM[†], **Joo-Young HWANG**[†], **Kuk-Hyun HAN**[††], **Jong-Hwan KIM**[††], ***Nonmembers***, ***and*** **Kyu-Ho PARK**[†], ***Regular Member***

SUMMARY Based on a Quantum-inspired Evolutionary Algorithm (QEA), a new disk allocation method is proposed for distributing buckets of a binary cartesian product file among unrestricted number of disks to maximize concurrent disk I/O. It manages the probability distribution matrix to represent the qualities of the genes. Determining the excellent genes quickly makes the proposed method have faster convergence than DAGA. It gives better solutions and 3.2 – 11.3 times faster convergence than DAGA.

key words: *partial match queries, optimal disk allocation, quantum-inspired evolutionary algorithm*

1. Introduction

Today, large databases are used for application areas such as web servers, spatial databases and large knowledge bases. Large databases require very large amount of data which is difficult to store and access efficiently. Distributing a large file onto multiple disks and accessing them in parallel make it possible to enhance the performance of the large databases.

This paper proposes an efficient disk allocation method for partial match queries on large binary cartesian product files. The disk allocation problem is to distribute the buckets of multi-attribute files among multiple disks with the purpose of minimizing the average response time of all partial match queries.

The problem of finding optimal bucket distribution is known to be NP-hard [1] and heuristic approaches have been proposed [1]–[4]. In [4], it is shown that DAGA gives the best solution among the heuristic approaches if the number of disks is not restricted. DAGA is based on a genetic algorithm. In the genetic algorithm, evolution status is represented by a population which consists of encoded solutions (chromosomes). The population evolves by simple operations such as reproduction, crossover and mutation of the solutions [5]. The excellent genes remain in the solutions selected based on the law of the survival of the fittest. The genetic algorithm is so general that it can be applied to various problems. However, its converging time is too long due to the control based on the chromosomes not on the genes.

In [6], Han and Kim proposed a Quantum-inspired Evolutionary Algorithm (QEA) whose convergence is faster than that of the conventional genetic algorithms. It is an evolutionary algorithm which is similar to the genetic algorithm. Its evolution status representation and evolution process are more efficient than those of the genetic algorithm. The evolution status of QEA is represented by a probability distribution matrix which represents the qualities of the genes. A population is sampled based on the probability distribution matrix and is evaluated. By using an insight to a given problem, excellent genes are determined and their probabilities are increased, which makes the converging time of QEA faster than that of the genetic algorithms.

In this paper, we propose QEA-based Disk allocation Method (QDM) of buckets in a binary cartesian product file among multiple disks to minimize the average response time of all the partial match queries on the file. QDM gives better solutions than DAGA and QDM is 3.2 – 11.3 times faster than DAGA for generating the solutions of equal quality.

2. Problem Definition

To describe the disk allocation problem, we introduce some necessary definitions.

Let $\{X_1, X_2, \cdots, X_n\}$ be a set of attributes. Each attribute X_i is associated with a domain, denoted D_i. For a given number, k, each domain D_i is partitioned into k disjoint subsets, $D_{i1}, D_{i2}, \cdots, D_{ik}$. A file is a finite subset of the cartesian product $D_1 \times \cdots \times D_n$. An element of a file is called a record. A k-ary cartesian product file is a file whose records are stored in k^n buckets such that all records in every bucket are in $D_{1j_1} \times \cdots \times D_{nj_n}$, where $1 \leq j_i \leq k$ for $1 \leq i \leq n$.

In case of a binary cartesian product file, each bucket is denoted by a binary string $[j_1 j_2 \cdots j_n]$, where $j_1, j_2, \cdots, j_n$ are bits. An n-attribute binary cartesian

Manuscript received May 20, 2002.
Manuscript revised September 4, 2002.

[†]The authors are with the Computer Engineering Research Laboratory, Department of Electrical Engineering and Computer Science [Division of Electrical Engineering], Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong Yuseong-gu, Daejon, 305–701, Republic of Korea.

[††]The authors are with the Robot Intelligence Technology Laboratory, Department of Electrical Engineering and Computer Science [Division of Electrical Engineering], Korea Advanced Institute of Science and Technology, 373-1 Guseong-dong Yuseong-gu, Daejon, 305–701, Republic of Korea.

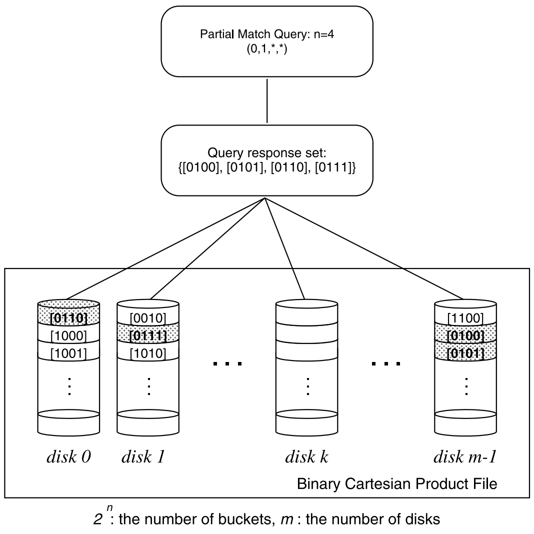

Fig. 1 Disk allocation problem: Example.

product file consists of 2^n buckets. A partial match query is a query of the form $q : (a_1, a_2, \cdots, a_n)$, where a_i is 0, 1, or $*$ (unspecified). The response set of a query q, denoted as $R(q)$, is the set of buckets that qualify for q. The response time of a query q is defined as the maximum among $N_0^q, N_1^q, \ldots, N_{m-1}^q$, where N_i^q is the number of qualifying buckets on disk i for the query q.

An example of 4-attribute binary cartesian product file is given in Fig. 1. For a query $q = (0, 1, *, *)$, the response set, $R(q)$, is $\{[0100], [0101], [0110], [0111]\}$. The bucket [0110] is allocated to disk 1, the bucket [0111] to disk 2, and the two buckets [0100] and [0101] to disk $m - 1$. Those qualifying buckets will be accessed from the multiple disks and the response time is 2. If all of the qualifying buckets are allocated to different disks, the response time will be reduced to 1. The problem is stated as follows.

Problem Definition

Given an n-attribute binary cartesian product file and m disks, allocate all buckets onto m disks to minimize the average response time for all partial match queries.

3. Related Work: DAGA

In DAGA, the 2^n buckets are allocated to the 2^n positions that are mapped to the m disks in a round-robin manner. A solution consists of 2^n genes where the i-th gene represents the bucket allocated to the i-th position. An example of DAGA's solution for $n = 3$ and $m = 3$ is given in Table 1. In solution 1, buckets 5, 0, 6 are allocated to disk 0, buckets 4, 7, 2 to disk 1 and buckets 3,1 to disk 2.

Initial population consisting of N_p solutions is created, the quality of all the solutions are evaluated, and the next generation is generated using the genetic evolution process including reproduction, crossover, and mutation. The reproduction is to select N_p solutions randomly in the current population. The solutions with higher fitness values are more likely to be selected than the solutions with lower fitness. The crossover operation is to exchange the genes of two solutions. The mutation operation is to modify the genes of a solution randomly. The procedure described above is repeated until a termination condition is met.

The quality is measured at the level of solution in DAGA. Therefore, bad genes remain in the solution although the fitness of the solution is high. If the quality is measured at the level of genes, the excellent genes are determined quickly, so the convergence is accelerated. The proposed QEA-based disk allocation method manages the probability distribution matrix to represent the qualities of the genes. Determining the excellent genes quickly makes QDM have faster convergence than DAGA.

Table 1 Solution examples in DAGA when $n = 3$ and $m = 3$.

Position	0	1	2	3	4	5	6	7
Solution 1	5	4	3	0	7	1	6	2

Table 2 Probability distribution matrix of QDM where $\Gamma_{i,j}$ is the probability for the i-th bucket to be allocated to the j-th disk.

Disk	Bucket			
	0	1	$\cdots$	$2^n - 1$
0	$\Gamma_{0,0}$	$\Gamma_{1,0}$	$\cdots$	$\Gamma_{2^n-1,0}$
1	$\Gamma_{0,1}$	$\Gamma_{1,1}$	$\cdots$	$\Gamma_{2^n-1,1}$
$\vdots$		$\vdots$		
$m-1$	$\Gamma_{0,m-1}$	$\Gamma_{1,m-1}$	$\cdots$	$\Gamma_{2^n-1,m-1}$

4. QEA-Based Disk Allocation Method

In QDM, the evolution status is represented by a $2^n \times m$ probability distribution matrix as in Table 2 where (i, j) entry, $\Gamma_{i,j}$, is the probability for the i-th bucket to be allocated to the j-th disk. Initially, all the probabilities are set to $\frac{1}{m}$. A solution of QDM consists of 2^n genes where the i-th gene represents the disk to which the i-th bucket is allocated. The disk for a bucket is randomly selected based on the probability distribution matrix. For evenly distribution of the buckets on the disks, the order of the buckets to allocate is randomized and the maximum number of the buckets allocated to a disk is restricted to $\lceil \frac{2^n}{m} \rceil$.

A solution, S, has the fitness value, $f(S)$, defined as $1/T_{avg}(S)$ where $T_{avg}(S)$ is the average response time of S for partial match queries. For an n-attributed binary cartesian product file, there exist 3^n queries, but the response time is always 1 for the queries with all the specified bits and $\lceil \frac{2^n}{m} \rceil$ for the query with all the unspecified bits. Thus, the number of queries to consider is $(3^n - 2^n - 1)$ and the fitness function is defined

Table 3 Solution examples in QDM when $n = 3$ and $m = 3$.

Bucket	0	1	2	3	4	5	6	7	f
Solution 1	0	2	1	2	1	0	0	1	0.72
Solution 2	0	2	2	2	1	1	0	0	0.62

Algorithm 1 QDM Procedure

```
Initializing probability distribution matrix
Generating the solutions
Evaluation
Storing the best solution to $\overline{S}$
while not termination-condition do
    Generating the solutions
    Evaluation
    Updating probability distribution matrix
    Storing the best solution to $\overline{S}$
end while
```

as Eq. (1).

$$\begin{aligned} f(S) &= (T_{avg}(S))^{-1} \\ &= \left(\sum_{r=1}^{3^n-2^n-1} p_r \cdot \max\{N_0^{q_r}|_S, N_1^{q_r}|_S, \cdots, N_{m-1}^{q_r}|_S\} \right)^{-1}, \end{aligned} \tag{1}$$

where $N_j^{q_r}|_S$ is the number of qualifying buckets on the j-th disk for the partial query q_r in the case of S and p_r is the access probability for q_r. In the case of evenly distributed query load, p_r for all r is $(3^n - 2^n - 1)^{-1}$.

Table 3 shows the solution examples when $n = 3$ and $m = 3$. In solution 1, buckets 0, 5, 6 are allocated to disk 0, buckets 2, 4, 7 to disk 1 and buckets 1, 3 to disk 2, and its fitness is 0.72.

The overall process of QDM is described in Algorithm 1. At each iteration, N_p solutions are generated based on the probability distribution matrix, the fitness values of the solutions are evaluated by the fitness function, f, and the probability distribution matrix is updated. At the k-th iteration, the probability distribution matrix is updated by comparing the solutions of the k-th iteration with the solution, $\overline{S}$ which is the best solution until the $(k-1)$-th iteration. The gene of a solution is *heterogeneous* if its value is different from that of the corresponding gene of $\overline{S}$. When the i-th gene of a solution is heterogeneous and its value is s_i, Γ_{i,s_i} is decreased (increased) if the fitness of the solution is lower (higher) than that of $\overline{S}$.

For example, assuming that solution 1 in Table 3 is $\overline{S}$ with the fitness value of 0.72, the probability distribution matrix for solution 2 in Table 3 with the fitness value of 0.62 is updated as follows. The heterogeneous genes of the solution are the 2nd, the 5th and the 7th genes. Since the fitness of solution 2 is lower than that of $\overline{S}$, the probabilities corresponding to the values of the heterogeneous genes ($\Gamma_{2,2}$, $\Gamma_{5,1}$, $\Gamma_{7,0}$) are decreased and the 2nd, the 5th and the 7th column vectors are normalized. The amount of the decrease for the probability is proportional to the difference between the fitness values of $\overline{S}$ and S, $\triangle f$, and inversely proportional to the number of heterogeneous genes, $\triangle d$.

Table 4 Probability distribution matrix after 50 iterations when $n = 3$ and $m = 3$.

Disk	Bucket							
	0	1	2	3	4	5	6	7
0	0.85	0.07	0.06	0.08	0.02	0.92	0.92	0.08
1	0.05	0.22	0.89	0.08	0.75	0.03	0.02	0.89
2	0.1	0.71	0.05	0.84	0.23	0.05	0.06	0.03

Algorithm 2 Updating probability distribution matrix for a solution, S

```
$\triangle f = f(\overline{S}) - f(S)$
$\triangle d$ = the number of heterogeneous genes which are genes of $S$ different from the genes of $\overline{S}$
if $\triangle f \leq 0$ then
    for $i = 0$ to $2^n-1$ do
        if $s_i \neq \overline{s}_i$ then
            $\Gamma_{i,j} \leftarrow U(i, s_i, \mu_1)$
        end if
    end for
else
    for $i = 0$ to $2^n-1$ do
        if $s_i \neq \overline{s}_i$ then
            $\Gamma_{i,j} \leftarrow U(i, s_i, -\mu_2 \cdot \frac{\triangle f}{\triangle d})$
        end if
    end for
end if
Normalizing probability distribution matrix to be $\sum_{j=0}^{m-1}\Gamma_{i,j} = 1$ for all $i$
```

The algorithm of updating the probability distribution matrix for a solution, S, is described in Algorithm 2 where s_i and $\overline{s}_i$ are the values of the i-th gene in S and $\overline{S}$, respectively. The update function, U, is defined as Eq. (2), where a positive μ returns the increased $\Gamma_{i,j}$ and negative μ returns the decreased $\Gamma_{i,j}$. Higher $|\mu|$ leads to more change of $\Gamma_{i,j}$. In the algorithm, μ_1 and μ_2 are constant values obtained experimentally.

$$U(i,j,\mu) = \begin{cases} \Gamma_{i,j} + (1-\Gamma_{i,j}) \times \mu, & \text{if } 0 \leq \mu \leq 1, \\ \Gamma_{i,j} - \Gamma_{i,j} \times |\mu|, & \text{if } -1 \leq \mu < 0. \end{cases} \tag{2}$$

Table 4 shows the probability distribution matrix at the 50th iteration when $n = 3$ and $m = 3$. μ_1 and μ_2 are 0.2 and 0.1, respectively. $\overline{S}$ at the 50th iteration is {0,2,1,2,1,0,0,1} which is near to the known optimal solution, {0,1,1,2,2,0,0,1}. The probabilities corresponding to the 1st gene and the 4th gene are not converged yet and the other genes are almost converged to the optimal values.

5. Experiments

The performance of QDM is compared with that of DAGA. All parameters of DAGA are adjusted as described in [4]. For QDM, the coefficients of the probability distribution matrix update function in Algorithm

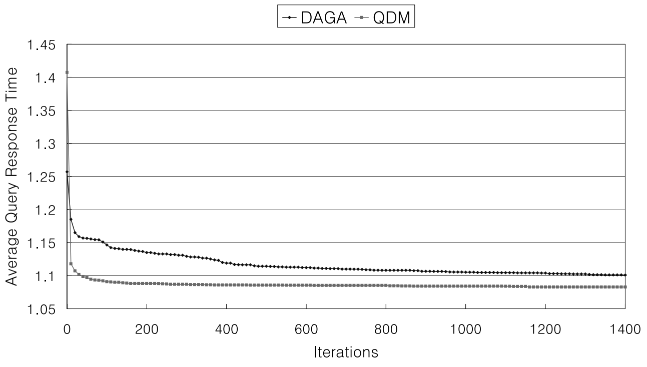

Fig. 2 Mean values of T_{avg}s with respect to iteration when $n = 5$ and $m = 12$.

2, μ_1 and μ_2, are adjusted to be 0.1 and 0.05, respectively. The numbers of sample solutions, N_p, for both methods are 50, 100 and 100 for $n = 5$, 6 and 7, respectively. Experiments are done for the various combinations of the number of disks and the number of attributes. The number of attributes, n, is varied from 5 to 7 and the number of disks, m, is varied from 4 to 16.

Two experiments are performed. In the first experiment, QDM and DAGA are configured to terminate after a specific number of iterations. To compare the convergence behaviors of the methods, the qualities of the solutions at the same number of iterations are compared. The second experiment compares the speeds of the methods to obtain the solutions with the same quality. Both methods are configured to terminate when their solutions satisfy that the average query response time is less than a specific target value.

In the first experiment, QDM and DAGA terminate after the same iterations which are 3,000, 6,000 and 6,000 iterations for $n = 5$, 6 and 7, respectively. The experiments show that QDM converges more quickly than DAGA. The average query response times of the best solutions found during the intermediate iterations are plotted in Fig. 2 which shows only the case of $n = 5$ and $m = 12$. As the number of iterations increases, the average query response time of QDM decreases more steeply than that of DAGA. The best solution found during the iterations is output as the final solution.

Both methods are run 30 times for each combination of the number of disks and the number of attributes. All the minimum average query response times of QDM are equal to or less than those of DAGA. The means and deviations of QDM are also less than those of DAGA. For illustration, those values at the 3000th iteration for the cases of $n = 5$ and $m = 4, 6, \ldots, 16$ are shown in Table 5.

In the second experiment, the number of iterations to obtain the solutions with target average response times are measured[†]. The target average response times are adjusted to be the mean values of the final solutions of DAGA obtained in the first experiment. The numbers of iterations are plotted in Fig. 3 for the cases of $n = 5$ and $m = 4, 6, \ldots, 16$. QDM requires 3.2 – 11.3 times smaller numbers of iterations than DAGA to find the solutions with the average query response times less than the same target values.

Table 5 Minimum, mean and deviation of query response time for $n = 5$ after 3000 iterations.

No. of Disks	*Minimum*		*Mean*		*Deviation*	
	T_{QDM}	T_{DAGA}	T_{QDM}	T_{DAGA}	σ_{QDM}	σ_{DAGA}
4	1.4095	1.4095	1.4619	1.5406	0.036	0.081
6	1.3286	1.3286	1.3431	1.3557	0.009	0.011
8	1.0857	1.1048	1.1240	1.1974	0.026	0.033
10	1.0857	1.0857	1.0917	1.1269	0.010	0.018
12	1.0714	1.0714	1.0833	1.0923	0.006	0.008
14	1.0619	1.0667	1.0690	1.0754	0.003	0.004
16	1.0000	1.0095	1.0062	1.0436	0.004	0.066

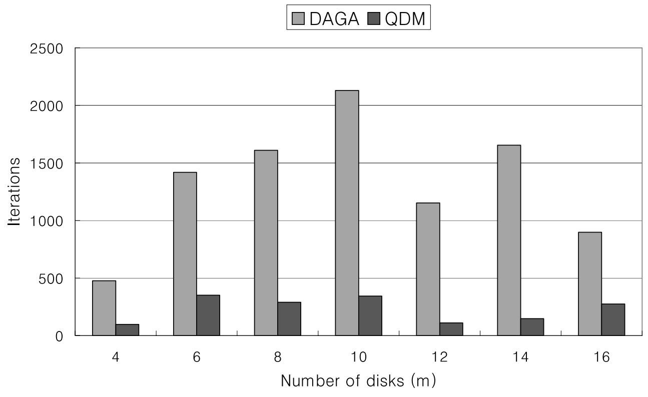

Fig. 3 The number of iterations of QDM and DAGA for generating the target solutions for $n = 5$.

6. Conclusion

The disk allocation problem is to distribute buckets of a multi-attribute file among multiple disks with the purpose of minimizing the average response time of all partial match queries. The previous best method, DAGA is based on a genetic algorithm and its converging time is too long [4]. In this paper, we proposed a QEA-based disk allocation method called QDM. It manages the probability distribution matrix to represent the qualities of the genes and evaluates the qualities of the genes from the sampled solutions. Determining the excellent genes quickly makes QDM have faster convergence than DAGA. Our experiments show that the average query

[†]The run times of each iteration for both QDM and DAGA are the same because more than 90% of the computing time is spent in calculating the fitness values and the algorithms' overhead is very small. So, the number of iterations is used for comparing the run times of the algorithms.

response times of QDM are equal to or less than those of DAGA and the convergence of QDM is 3.2 – 11.3 times faster than that of DAGA.

Acknowledgment

This work has been supported by National Research Laboratory Project of the Korea Ministry of Science and Technology.

References

[1] Y.Y. Sung, "Parallel searching for binary cartesian product files," Proc. 1985 ACM Computer Science Conference, pp.163–172, 1985.

[2] K.A.S. Abdel-Ghaffar, "Optimal disk allocation for partial match queries," ACM Trans. Database Systems, vol.18, no.1, pp.132–156, 1993.

[3] M. Kim and S. Pramanik, "Optimal file distribution for partial match retrieval," Proc. 1988 ACM International Conference on Management of Data, pp.173–182, 1988.

[4] D.Y. Ahn and K.H. Park, "Disk allocation methods using genetic algorithm," IEICE Trans. Inf. & Syst., vol.E82-D, no.1, pp.291–300, Jan. 1999.

[5] D. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[6] K.-H. Han and J.-H. Kim, "Quantum-inspired evolutionary algorithm for a class of combinatorial optimization," IEEE Trans. Evol. Comput., vol.6, no.6, pp.580–593, 2002.