DNA Algorithms of Implementing Biomolecular Databases on a Biological Computer

Weng-Long Chang* and Athanasios V. Vasilakos

Abstract—**In this paper, DNA algorithms are proposed to perform eight operations of relational algebra (calculus), which include** *Cartesian product, union, set difference, selection, projection, intersection, join,* **and** *division,* **on biomolecular relational databases.**

Index Terms—**Biomolecular relational algebra, biomolecular relational databases, DNA computing.**

I. Introduction

The GENETIC information of cellular organisms is encoded by DNA (deoxyribonucleic acid) in [1]–[3]. DNA includes polymer chains which are commonly regarded as DNA strands. Each strand may be made of a sequence of nucleotides, or bases, attached to a sugar-phosphate "backbone." The four DNA nucleotides are adenine, guanine, cytosine and thymine, commonly abbreviated to A, G, C, and T, respectively. The classical double helix of DNA is formed when two separate single strands bond. Bonding occurs by the pairwise attraction of bases: A bonds with T and G bonds with C in [1]–[3]. Double-stranded DNA may be denatured into single strands by heating the solution to a temperature determined by the composition of the strand in [1]–[3].

From [1]–[3], storing information in molecules of DNA allows for an information density of approximately 1 bit per cubic nm (nanometer) and a dramatic improvement over existing storage media. Relational database system in [4] is the most popular one. One interesting open question is asking whether relational database system in [4] can be constructed by means of biological operations and DNA strands or not. Our motivation is to find the answer of the interesting open question.

Our major contributions in this journal paper are as follows.

- We demonstrate that biological operations can be applied to construct biomolecular databases where data records in relational tables are encoded as DNA strands.
- We propose DNA-based algorithms to complete eight operations of relational algebra (calculus), which contain *Cartesian product*, *union*, *set difference*, *selection, projection, intersection, join*, and *division*.
- We analyze complexity, strengths and weaknesses of constructing biomolecular relational databases.
- We theoretically show that constructing biomolecular databases on a molecular computer is a very feasible task.

Manuscript received May 16, 2014; revised July 21, 2014; accepted October 03, 2014. Date of publication October 17, 2014; date of current version February 27, 2015. This work was supported by National Science Foundation of Republic of China under Grants NSC-102-2221-E-151-031- and NSC-102-2622-E-151-013-CC3. *Asterisk indicates corresponding author.*

*W.-L. Chang is with the Department of Computer Science and Information Engineering, National Kaohsiung University of Applied Sciences, Kaohsiung 807, Taiwan (e-mail: changwl@cc.kuas.edu.tw).

A. V. Vasilakos is with the National Technical University of Athens, Greece (e-mail: vasilako@ath.forthnet.gr).

Digital Object Identifier 10.1109/TNB.2014.2363470

II. Computational Model

Definition 2–1: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall\, x_d \in \{0,1\}$ for $1 \le d \le n\}$ and a bit x_j, the biomolecular operation "Append-Head" appends x_j onto the head of every element in set X. The formal representation is written as Append-Head $(X, x_j) = \{x_j x_n x_{n-1} \ldots x_2 x_1 \mid \forall\, x_d \in \{0,1\}$ for $1 \le d \le n$ and $x_j \in \{0,1\}\}$.

Definition 2–2: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$ and a bit x_j, the biomolecular operation, "Append-Tail," appends x_j onto the end of every element in set X. The formal representation is written as Append-Tail $(X, x_j) = \{x_n x_{n-1} \ldots x_2 x_1 x_j \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n$ and $x_j \in \{0,1\}\}$.

Definition 2–3: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$, the biomolecular operation "Discard (X)" sets X to be an empty set and can be represented as "$X = \varnothing$."

Definition 2–4: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$, the biomolecular operation "Amplify $(X, \{X_i\})$" creates a number of identical copies X_i of set X, and then "Discard (X)."

Definition 2–5: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$ and a bit, x_j, if the value of x_j is equal to one, then the biomolecular *extract* operation creates two new sets, $+(X, x_j^1) = \{x_n x_{n-1} \ldots x_j^1 \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \ne j \le n\}$ and $-(X, x_j^1) = \{x_n x_{n-1} \ldots x_j^0 \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \ne j \le n\}$. Otherwise, it produces another two new sets, $+(X, x_j^0) = \{x_n x_{n-1} \ldots x_j^0 \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \ne j \le n\}$ and $-(X, x_j^0) = \{x_n x_{n-1} \ldots x_j^1 \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \ne j \le n\}$.

Definition 2–6: Given m sets $X_1 \ldots X_m$, the biomolecular *merge* operation, $\cup(X_1, \ldots, X_m) = X_1 \cup \cdots \cup X_m$.

Definition 2–7: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$, the biomolecular operation "Detect (X)" returns *true* if $X \ne \varnothing$. Otherwise, it returns *false*.

Definition 2–8: Given set $X = \{x_n x_{n-1} \ldots x_2 x_1 \mid \forall x_d \in \{0,1\}$ for $1 \le d \le n\}$, the biomolecular operation "Read (X)" describes any element in X.

III. Constructing Biomolecular Databases

A. Introduction to a Relational View of Data

Given sets S_1, S_2, …, S_n (not necessarily distinct), R is a relation on these n sets if it is a set of n-tuples, each of which has its first element from S_1, its second element from S_2, and so on [4]. We shall refer to S_j as the jth domain of R. As defined above, R is said to have degree n. Relations of degree n is called as n-ary. From [4], an n-ary relation R has the five properties: 1) each row represents an n-tuple of R, 2) the ordering of rows is immaterial, 3) all rows are distinct, 4) the ordering of columns is significant, and 5) the significance of each column is partially conveyed by labeling it with the name of the corresponding domain.

B. DNA Algorithms for the Cartesian Product on Biomolecular Databases

The *Cartesian product* (or cross-product or just product) of n sets, S_1, S_2, $\ldots S_n$, is the set of pairs that can be formed by choosing the first element of the pair to be any element of S_1, the second element of the pair to be any element of S_2, and so on [4]. Assume that L_k is the number of bits for the value of each element in S_k to $1 \leq k \leq n$ and that R is an n-ary relation and has m elements. It is assumed that R is equal to $\{(r_{i,1}, \ldots r_{i,n}) | r_i, k \in S_k$ for $1 \leq k \leq n$ and $1 \leq i \leq m\}$ and that the value encoding $r_{i,k}$ in R can be represented as a binary number, $v_{i,k,1} \ldots V_{i,k,L_k}$ for $1 \leq k \leq n$ and $1 \leq i \leq m$. The bits $v_{i,k,1}$ and v_{i,k,L_k} represent the first bit and the last bit for $r_{i,k}$, respectively. From [1]–[3], to $1 \leq j \leq L_k$ for every bit $v_{i,k,j}$ that encodes the j^{th} bit of the k^{th} field of the i^{th} element in an n-ary relation, two *distinct* DNA strands (sequences) are designed to minimize the possibility of unwanted binding and their length is θ base pairs. One represents the value "0" for $v_{i,k,j}$ and the other represents the value "1" for $v_{i,k,j}$. For the sake of convenience in our presentation, it is assumed that ${v_{i,k,j}}^1$ denotes the value of $v_{i,k,j}$ to be 1, ${v_{i,k,j}}^0$ defines the value of $v_{i,k,j}$ to be 0, and $v_{i,k,j}$ defines the value of $v_{i,k,j}$ to be 0 or 1. DNA algorithms that are used to implement a relational algebra (calculus), the *Cartesian product*, for constructing a biomolecular database R are, respectively, proposed in Sections III-B1 and III-B2.

1) A DNA Algorithm of Inserting One Record Into Biomolecular Databases: The following algorithm, Insert (T_{80}, i), is applied to insert one record into a biomolecular database R denoted in Section III-B. In Insert (T_{80}, i), tube T_{80} is the first parameter and is initially set to an empty tube. The second parameter, i, is used to represent the i^{th} record that will be inserted into the biomolecular database R.

```
Procedure Insert $(T_{80}, i)$
1) For $k = 1$ to $n$
    2) For $j = 1$ to $L_k$
        2a) Append-Tail $(T_{80}, v_{i,k,j})$.
    EndFor
EndFor
EndProcedure
```

Lemma 3–1: One record in a biomolecular database R can be constructed with a DNA strand from the algorithm Insert (T_{80}, i).

Proof: It consists of one nested loop. The outer loop is applied to insert one record (including n fields) into a biomolecular database R. The inner loop is employed to construct each field of one record in R. Each time, Step 2a) is used to append a DNA sequence, representing the value 0 or 1 for $v_{i,k,j}$, onto the end of every strand in tube T_{80}. This is to say that the value 0 or 1 to the j^{th} bit in the k^{th} field of the i^{th} record in R appears in tube T_{80}. After repeating the execution of Step 2a), it finally produces tube T_{80} that consists of a DNA sequence with $(\theta \times (\sum_{k=1}^{n} L_k))$ base pairs, representing one record in R. ■

2) A DNA Algorithm of Implementing the Cartesian Product on Biomolecular Databases: The following algorithm, CartesianProduct (T_0, m), is used to construct a biomolecular database R denoted in Section III-B. In CartesianProduct (T_0, m), tube T_0 is the first parameter and is initially set to an empty tube. The second parameter, m, is employed to represent the number of records in the biomolecular database R.

```
Procedure CartesianProduct $(T_0, m)$
1) For $i = 1$ to $m$
    1a) Insert $(T_{80}, i)$.
    1b) $T_0 = \cup(T_0, T_{80})$.
EndFor
EndProcedure
```

Lemma 3–2: A biomolecular database R can be constructed with DNA strands from the algorithm, CartesianProduct (T_0, m).

Proof: A single loop is used to insert m records into a biomolecular database R. Each time, Step 1a) is applied to call the procedure, Insert (T_{80}, i), to insert one record (including n fields) into a biomolecular database R. This is to say that the i^{th} record in R appears in tube T_{80}. Next, Step 2) is applied to pour tube T_{80} into tube T_0. This implies that the i^{th} record in R appears in tube T_0 and tube T_{80} becomes an empty tube. After repeating the execution of Steps 1a) and 1b), it finally produces tube T_0 that consists of m DNA sequences, representing m records in R. ■

C. DNA Algorithms for Set Operations on Biomolecular Databases

The three most common operations on sets are *union*, *intersection*, and *difference*. The contents of Definition 3–1 through Definition 3–3 are directly cited from [4] and are used to explain how these operations perform their functions on arbitrary sets X and Y.

Definition 3–1: $X \cup Y$, the *union* of X and Y, is the set of elements that are in X or Y or both. An element appears only once in the union even if it is present in both X and Y.

Definition 3–2: $X \cap Y$, the *intersection* of X and Y, is the set of elements that are in both X and Y.

Definition 3–3: $X - Y$, the *difference* of X and Y, is the set of elements that are in X but not in Y. Note that $X - Y$ is different from $Y - X$; the latter is the set of elements that are in Y but not in X.

When we apply these operations above to n-ary relations, we need to put some conditions on X and Y. The first condition is that X and Y must have identical sets of columns, and the domain for each column must be the same in X and Y. The second condition is that before we compute the set-theoretic union, intersection, or difference of sets of tuples, the columns of X and Y must be ordered so that their order is the same for both relations. DNA algorithms for performing these operations are, respectively, proposed in Sections III-C1, III-C2, and III-C3.

1) A DNA Algorithm for the Union Operator on Biomolecular Databases: Assume that X and Y are n-ary relations and have p elements and q elements, respectively. Suppose also that X and Y are, respectively, equal to $\{(r_{i,1}, \dots r_{i,n}) | r_{i,k} \in S_k$ for $1 \le k \le n$ and $1 \le i \le p\}$ and $\{(r_{i,1}, \dots r_{i,n}) | r_{i,k} \in S_k$ for $1 \le k \le n$ and $1 \le i \le q\}$. After the two DNA algorithms, CartesianProduct (T_1, p) and CartesianProduct (T_2, q), are called and are performed, tube T_1 consists of p DNA sequences representing p records in X and tube T_2 includes q DNA sequences representing q records in Y. The following DNA algorithm is used to perform $X \cup Y$. Notations used in the following DNA algorithm appear in Section III-B.

In Section III-B, a binary number $v_{i,k,j}$ is used to encode the j^{th} bit of the k^{th} field of the i^{th} element in an n-ary relation and its value is zero or one. When we design the DNA-based algorithm, Union (T_1, T_2, T_3, p), in advance we do not know that the value of $v_{i,k,j}$ is zero or one. For the convenience of our presentation, the *second* parameter of the *extract* operation in Step 7a) of Union (T_1, T_2, T_3, p) is set to $v_{i,k,j}$. If its value is given as *one*, then Step 7a) in Union (T_1, T_2, T_3, p) is to implement $T_{22} = +(T_{22},\ {v_{i,k,j}}^1)$ and $T_{22}^{OFF} = -(T_{22},\ {v_{i,k,j}}^1)$. Otherwise, Step 7a) in Union (T_1, T_2, T_3, p) is to implement $T_{22} = +(T_{22},\ {v_{i,k,j}}^0)$ and $T_{22}^{OFF} = -(T_{22},\ {v_{i,k,j}}^0)$.

```
Procedure Union $(T_1, T_2, T_3, p)$
1) Amplify $(T_1, T_{11}, T_{12})$.
2) Amplify $(T_2, T_{21}, T_{22})$.
3) $T_1 = \cup(T_1, T_{11})$.
4) $T_2 = \cup(T_2, T_{21})$.
5) For $i = 1$ to $p$
    6) For $k = 1$ to $n$
        7) For $j = 1$ to $L_k$
            7a) $T_{22} = +(T_{22}, v_{i,k,j})$ and
            ${T_{22}}^{OFF} = -(T_{22}, v_{i,k,j})$.
            7b) ${T_{22}}^{ON} = \cup({T_{22}}^{ON}, {T_{22}}^{OFF})$.
        EndFor
    EndFor
        7c) Discard $(T_{22})$.
        7d) $T_{22} = \cup(T_{22}, T_{22}^{ON})$.
    EndFor
8) $T_3 = \cup(T_{12}, T_{22})$.
9) Read $(T_3)$.
EndProcedure
```

Lemma 3–3: The union operator on two n-ary relations can be performed with DNA strands from the algorithm, Union (T_1, T_2, T_3, p).

Proof: Please refer to the proof of Lemma 3–1 and Lemma 3–2. ■

2) A DNA Algorithm for the Intersection Operator on Biomolecular Databases: Assume that X and Y were denoted in Section III-C1, tube T_1 consists of p DNA sequences representing p records in X, and tube T_2 includes q DNA sequences representing q records in Y. The following DNA algorithm is used to perform $X \cap Y$. Notations used in the following DNA algorithm appear in Section III-B.

```
Procedure Intersection $(T_1, T_2, T_4, p)$
1) Amplify $(T_2, T_{21}, T_{22})$.
2) $T_2 = \cup(T_2, T_{21})$.
3) For $i = 1$ to $p$
4) For $k = 1$ to $n$
5) For $j = 1$ to $L_k$
            5a) $T_{22} = +(T_{22}, v_{i,k,j})$ and
            ${T_{22}}^{OFF} = -(T_{22}, v_{i,k,j})$.
            5b) ${T_{22}}^{ON} = \cup({T_{22}}^{ON}, {T_{22}}^{OFF})$.
        EndFor
    EndFor
    5c) $T_4 = \cup(T_4, T_{22})$.
    5d) $T_{22} = \cup(T_{22}, {T_{22}}^{ON})$.
  EndFor
6) Discard $(T_{22})$.
7) Read $(T_4)$.
EndProcedure
```

Lemma 3–4: The intersection operator on two n-ary relations can be performed with DNA strands from the algorithm, Intersection (T_1, T_2, T_4, p).

Proof: Please refer to the proof of Lemma 3–1 and Lemma 3–2. ■

3) A DNA Algorithm for the Difference Operator in Biomolecular Databases: Suppose that X and Y were defined in Section III-C-IV, tube T_1 consists of p DNA sequences representing p records in X, and tube T_2 includes q DNA sequences representing q records in Y. The following DNA algorithm is used to perform $X - Y$. Notations used in the following DNA algorithm appear in Section III-B.

```
Procedure Difference $(T_1, T_2, T_5, q)$
1) Amplify $(T_1, T_{11}, T_{12})$.
2) Amplify $(T_2, T_{21}, T_{22})$.
3) $T_1 = \cup(T_1, T_{11})$.
4) $T_2 = \cup(T_2, T_{21})$.
5) For $i = 1$ to $q$
6) For $k = 1$ to $n$
7) For $j = 1$ to $L_k$
            7a) $T_{12} = +(T_{12}, v_{i,k,j})$ and
            ${T_{12}}^{\mathrm{OFF}} = -(T_{12}, v_{i,k,j})$.
            7b) ${T_{12}}^{\mathrm{ON}} = \cup({T_{12}}^{ON}, {T_{12}}^{OFF})$.
            7c) $T_{22} = +(T_{22}, v_{i,k,j})$ and
            ${T_{22}}^{OFF} = -(T_{22}, v_{i,k,j})$.
            7d) ${T_{22}}^{ON} = \cup({T_{22}}^{ON}, {T_{22}}^{OFF})$.
        EndFor
    EndFor
    5a) If (Detect $(T_{22})$ = "yes") then
            5b) Discard $(T_{12})$.
    EndIf
    5c) $T_{12} = \cup(T_{12}, {T_{12}}^{ON})$.
    5d) $T_{22} = \cup(T_{22}, {T_{22}}^{ON})$.
  EndFor
8) $T_5 = \cup(T_5, T_{12})$.
9) Read $(T_5)$.
EndProcedure
```

Lemma 3–5: The difference operator on two n-ary relations can be performed with DNA strands from the algorithm, Difference (T_1, T_2, T_5, q).

Proof: Please refer to the proof of Lemma 3–1 through Lemma 3–2. ■

D. DNA Algorithms for the Projection Operator in Biomolecular Databases

From an n-ary relation R (denoted in Section III-B), the *projection* operator in [4] is applied to produce a new relation that has only some of R's columns. The projection operator on R is denoted as $\pi_{S_1,S_2,\ldots,S_n}$ (R). The value of expression $\pi_{S_1,S_2,\ldots,S_n}$ (R) is a relation that is equal to $\{(r_{i,b},\ldots,r_{i,a}) | r_{i,k} \in S_k$ for $1 \leq k \leq n$, $1 \leq b \leq n$, $1 \leq a \leq n$, $1 \leq i \leq m$ and each element is distinct $\}$. DNA algorithms for completing the expression $\pi_{S_1,S_2,\ldots,S_n}$ (R) are, respectively, proposed in Sections III-D1 and III-D2.

1) A DNA Algorithm of the Projection Operator on an N-ary Relation: The DNA algorithm, Projection (T_0, T_6, m, c), is applied to perform the expression $\pi_{S_1,S_2,\ldots,S_n}$ (R), and notations used in the DNA algorithms are denoted in Section III-B. In Projection (T_0, T_6, m, c), tube T_0 is the first parameter and DNA strands in tube T_0 are applied to represent m elements in R. Tube T_6 is the second parameter and is set to an empty tube. The third parameter, m, is applied to represent the number of elements in R. The fourth parameter, c, is used to represent the number of specified columns for R. For the convenience of our presentation, in the expression $\pi_{S_1,S_2,\ldots,S_n}$ (R), we assign one number k to each specified column S_k in R. In Projection (T_0, T_6, m, c), Steps 3), 4), and 5) are a nested loop and are applied to extract the values of the specified columns for R and eliminate duplicates. Because the range of the value for index d at each iteration of Step 4) is from one through c, its corresponding specified column is different. Therefore, for the convenience of our presentation, it is assumed that the specific d^{th} column for R corresponds to the k^{th} domain so that the notation $v_{i,k,j}$ in Projection (T_0, T_6, m, c) is consistent with that notation $v_{i,k,j}$ denoted in Section III-B.

```
Procedure Projection $(T_0, T_6, m, c)$
1) Amplify $(T_0, T_7, T_8)$.
2) $T_0 = \cup(T_0, T_7)$.
3) For $i = 1$ to $m$
4) For $d = 1$ to $c$
5) For $j = 1$ to $L_k$
            5a) $T_8 = +(T_8, v_{i,k,j})$ and $T_8^{OFF} = -(T_8, v_{i,k,j})$, where the specific $d^{th}$ column for $R$ corresponds to the $k^{th}$ domain.
            5b) $T_8^{ON} = \cup(T_8^{ON}, T_8^{OFF})$.
      EndFor
      6) If (Detect $(T_8)$ = "yes") then
            7) For $j = 1$ to $L_k$
                  7a) Append-Tail $(T_9, v_{i,k,j})$, where the specific $d^{th}$ column for $R$ corresponds to the $k^{th}$ domain.
            EndFor
         EndIf
      8) $T_8 = \cup(T_8, T_8^{ON})$.
      EndFor
      9) If (Detect $(T_9)$ = "yes") then
            10) If (Detect $(T_6)$ = "yes") then
                  11) JudgeDistinctElement $(T_6, T_9, T_6^{>}, T_6^{=}, T_6^{<}, i, c)$.
                  12) If (Detect $(T_6^{=})$ = "no") then
                        13) $T_6 = \cup(T_6, T_9)$.
                  Else
                        14) Discard $(T_9)$.
                  EndIf
            Else
            15) $T_6 = \cup(T_6, T_9)$.
         EndIf
   EndFor
   16) Read $(T_6)$.
EndProcedure
```

Lemma 3–6: The projection operator on an n-ary relation can be performed with DNA strands from the algorithm, Projection (T_0, T_6, m, c).

Proof: DNA strands in tube T_0 are applied to represent m elements in R. Step 1) is used to amplify tube T_0 and to generate two new tubes, T_7 and T_8, which are copies of T_0 and tube T_0 becomes empty. Step 2) is then used to pour tube T_7 into tube T_0. This is to say that DNA strands representing m elements in R are still reserved in tube T_0. From Step 2), the property for no change of elements in R is satisfied in the processing of expression $\pi_{S_1,S_2,\ldots,S_n}$ (R). Steps 3), 4), and 5) are a nested loop and are applied to extract the values of the specified columns for R and eliminate duplicates.

On the execution of Step 5a), it applies the *extract* operation to form two tubes, T_8 and T_8^{OFF}. DNA strands in tube T_8 represent the values that are equal to the value of $v_{i,k,j}$. The values encoded by DNA strands in tube T_8^{OFF} are not equal to the value of $v_{i,k,j}$. Next, each time, Step 5b) uses the *merge* operation to pour tube T_8^{OFF} into tube T_8^{ON}. After repeating the execution of Steps 5a) and 5b) until the value of the loop variable j reaches L_k, tube T_8 contains DNA strands values that are the i^{th} row of the specified columns in R, and tube T_8^{ON} includes DNA strands encoding values that are not the i^{th} row of the specified columns in R.

Step 6) is then applied to detect whether tube T_8 is not empty. If it returns a "yes," then Steps 7) and 7a) are executed. Step 7) is a single loop and is employed to generate values for the i^{th} row of the k^{th} specified column in R. Each time, Step 7a) is used to append a DNA sequence, representing the value 0 or 1 for $v_{i,k,j}$, into tube T_9. This is to say that the value 0 or 1 to the j^{th} bit in the i^{th} row of the k^{th} specified column in R appears in tube T_9. After repeating the execution of Step 7a) until the value of the loop variable j reaches L_k, DNA strands encoding values for the i^{th} row of the k^{th} specified column in R are appended into tube T_9. Next, on the execution of Step 8), it applies the *merge* operation to pour tube T_8^{ON} into tube T_8. This is to say that DNA strands encoding values for other rows of the specified columns in R are in tube T_8. After repeating to execute until the value of the loop variable d reaches c, DNA strands encoding values for the i^{th} row of the specified columns in R are appended into tube T_9.

Step 9) is then applied to detect whether tube T_9 is not empty. If it returns a "yes," then Steps 10) through 15) are executed. Otherwise, nothing is done. Step 10) is used to detect whether tube T_6 is not empty. If it returns a "yes," then Steps 11) through 14) are executed. Step 11) is then employed to call the algorithm, JudgeDistinctElement $(T_6, T_9, T_6^{>}, T_6^{=}, T_6^{<}, i, c)$, to produce three new tubes, $T_6^{>}$, $T_6^{=}$, and $T_6^{<}$. Tube $T_6^{>}$ includes DNA strands with the compared result of greater than, tube $T_6^{=}$ contains DNA strands with the compared result of equal to, and tube $T_6^<$ consists of DNA strands with the compared result of less than. Step 12) is used to detect whether tube T_6 is not empty. If it returns a "no," then Step 13) is employed to pour tube T_9 into tube T_6. Otherwise, Step 14) is employed to discard tube T_9. This implies that DNA strands encoding duplicates are removed. If Step 10) returns a "no," this is to say that tube T_6 is empty, Step 15) is applied to pour tube T_9 into tube T_6. After repeating to execute until the value of the loop variable i reaches m, DNA strands in tube T_6 encode values of the specified columns in R and duplicates in R are removed. For reading the answer(s) that satisfy the *projection* operator for an n-ary relation R, Step 9) uses the *read* operation to display the answer(s). ■

2) A DNA Algorithm of Eliminating Duplicates of the Projection Operator on an N-ary Relation: The following DNA algorithm, JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$, is applied to eliminate duplicates of the expression $\pi_{S_1,S_2,\ldots,S_n}(R)$, and notations used in the following DNA algorithms are denoted in Section III-B. The algorithm, JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$, is called by the algorithm Projection (T_0, T_6, m, c). In JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$, DNA strands in tube T_6 encode the previous records (from the *first* record through $(i-1)th$ record) of the specified columns in R and duplicates of them are removed. DNA strands in tube T_9 encode the i^{th} record of the specified column in R. Tubes $T_6{}^>$, $T_6{}^=$ and $T_6{}^<$ are initially set to empty tubes. The sixth parameter, i, is applied to represent that record which will be processed is the i^{th} record in R. The seventh parameter, c, is used to represent the number of specified columns for R. In JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$, Steps 3) and 4) are a nested loop and are applied to eliminate duplicated records of the specified columns in R. Because the range of the value for index d at each iteration of Step 4) is from one through c, its corresponding specified column is different. Hence, for the convenience of our presentation, it is assumed that the specific d^{th} column for R corresponds to the k^{th} domain so that the notation $v_{i,k,j}$ in JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$ is consistent with that notation $v_{i,k,j}$ denoted in Section III-B.

```
Procedure JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$.
1) Amplify $(T_6, T_6{}^{ON}, T_6{}^{OFF})$.
2) $T_6 = \cup(T_6, T_6{}^{ON})$.
3) For $d = 1$ to $c$
4) For $j = 1$ to $L_k$
        4a) $T_9{}^{ON} = +(T_9, v_{i,k,j}{}^1)$ and
            $T_9{}^{OFF} = -(T_9, v_{i,k,j}{}^1)$, where the
            specific $d^{th}$ column for $R$ corresponds to the
            $k^{th}$ domain.
        4b) $T_7{}^{ON} = +(T_6{}^{OFF}, v_{i,k,j}{}^1)$ and
            $T_7{}^{OFF} = -(T_6{}^{OFF}, v_{i,k,j}{}^1)$.
        4c) If (Detect ( $T_9{}^{ON}$ = "yes") then
                4d) $T_6{}^= = \cup(T_6{}^=, T_7{}^{ON})$.
                4e) $T_6{}^< = \cup(T_6{}^<, T_7{}^{OFF})$.
                4f) $T_9 = \cup(T_9, T_9{}^{ON})$.
            Else
                4g) $T_6{}^> = \cup(T_6{}^>, T_7{}^{ON})$.
                4h) $T_6{}^= = \cup(T_6{}^=, T_7{}^{OFF})$.
                4i) $T_9 = \cup(T_9, T_9{}^{OFF})$.
            EndIf
        4j) $T_6{}^{OFF} = \cup(T_6{}^{OFF}, T_9{}^=)$.
    EndFor
  EndFor
EndProcedure
```

Lemma 3–7: Duplicates of the projection operator on an n-ary relation can be eliminated with DNA strands from the algorithm, JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$.

Proof: Step 1) is applied to amplify tube T_6 and to generate two new tubes, $T_6{}^{ON}$ and $T_6{}^{OFF}$, which are copies of T_6 and tube T_6 becomes empty. Step 2) is then employed to pour tube $T_6{}^{ON}$ into tube T_6. This is to say that DNA strands representing values of the specified columns in R are still reserved in tube T_6. Steps 3) and 4) are a nested loop and are applied to eliminate duplicated values of the specified columns in R. On the execution of Step 4a), it applies the *extract* operation to form two test tubes, $T_9{}^{ON}$ and $T_9{}^{OFF}$. DNA strands in tube $T_9{}^{ON}$ have $v_{i,k,j} = 1$ and DNA strands in tube $T_9{}^{OFF}$ have $v_{i,k,j} = 0$. Next, each time, Step 4b) uses the *extract* operation to form two test tubes, $T_7{}^{ON}$ and $T_7{}^{OFF}$. DNA strands in tube $T_7{}^{ON}$ have $v_{i,k,j} = 1$ and DNA strands in tube $T_7{}^{OFF}$ have $v_{i,k,j} = 0$. On the execution of Step 4c), it employs the *detection* operation to detect whether tube $T_9{}^{ON}$ is not empty. If it returns a "yes," then Steps 4d) through 4f) are executed. Each time, Steps 4d) through 4f) use three *merge* operations to pour tubes $T_7{}^{ON}$, $T_7{}^{OFF}$, and $T_9{}^{ON}$, respectively, into tubes $T_6{}^=$, $T_6{}^<$, and T_9. If it returns a "no," then Step 4g), Step 4h) and Step 4i) are executed. Each time, Steps 4g) through 4i) apply also three *merge* operations to pour tubes $T_7{}^{ON}$, $T_7{}^{OFF}$, and $T_9{}^{OFF}$, respectively, into tubes $T_6{}^>$, $T_6{}^=$, and T_9.

On the execution of Step 4j), it employs the *merge* operation to pour tube $T_9{}^=$ into tube $T_6{}^{OFF}$. After repeating to execute Steps 4a) through 4j) until the value of the loop variable d reaches c, it finally produces tubes $T_6{}^>$, $T_6{}^=$, and $T_6{}^<$. DNA strands in tube $T_6{}^>$ have the result of greater than (">"), DNA strands in tube $T_6{}^=$ have the result of equal to ("="), and DNA strands in tube $T_6{}^<$ have the result of less than ("<"). Therefore, it is inferred that duplicates of the projection operator on an n-ary relation R can be eliminated with DNA strands from the algorithm, JudgeDistinctElement $(T_6, T_9, T_6{}^>, T_6{}^=, T_6{}^<, i, c)$. ■

E. DNA Algorithms for the Selection Operator in Biomolecular Databases

From an n-ary relation R (denoted in Section III-B), the *selection* operator from [4] is used to produce a new relation with a subset of R's tuples. The tuples in the resulting relation are those that satisfy some selected condition P that involves the columns of R. The selection operator on R is denoted as $\sigma_P(R)$. The selected condition P is expressed as $D\theta E$, where D is a column of R or a constant value, E is also a column of R or a *constant* value, and θ is any element in $\{=, >, <, \neq, \geq, \leq\}$. It is assumed that a column of R in D or E is the k^{th} column for $1 \leq k \leq n$ and L_k is the number of bits for the length of its domain. It is also assumed that D can be represented as $v_{i,k,j}$ denoted in Section III-B. Similarly, for ease in our presentation, it is supposed that E can be represented as a binary number $e_1 \cdots e_{L_k}$. The bits e_1 and e_{L_k} represent the first bit and the last bit for E, respectively. For every bit e_j to $1 \leq j \leq L_k$, the same DNA strands encoding $v_{i,k,j}$ are also used to encode it. One represents the value "0" for e_j and the other represents the value "1" for e_j. For the sake of convenience in our presentation, it is assumed that ${e_j}^1$ denotes the value of e_j to be 1, ${e_j}^0$ defines the value of e_j to be 0, and e_j defines the value of e_j to be 0 or 1. The value of expression $\sigma_P(R)$ is a relation that is equal to $\{(r_{i,1}, \ldots, r_{i,n}) | r_{i,k} \in S_k$ for $1 \leq k \leq n, 1 \leq i \leq m$ and the selected condition P is satisfied $\}$.

The following DNA algorithm, Selection $(T_0, {T_{16}}^{>}, {T_{16}}^{=}, {T_{16}}^{<}, {T_{16}}^{\neq}, {T_{16}}^{\geq}, {T_{16}}^{\leq}, D, E)$, is applied to complete the expression $\sigma_P(R)$, and notations used in Selection $(T_0, {T_{16}}^{>}, {T_{16}}^{=}, {T_{16}}^{<}, {T_{16}}^{\neq}, {T_{16}}^{\geq}, {T_{16}}^{\leq}, D, E)$ are denoted in Section III-B. Tube T_0 is first parameter and DNA strands in tube T_0 are applied to represent m elements in R. Tubes ${T_{16}}^{>}$, ${T_{16}}^{=}$, ${T_{16}}^{<}$, ${T_{16}}^{\neq}$, ${T_{16}}^{\geq}$, and ${T_{16}}^{\leq}$ are initially set to empty tubes. The purpose of tube ${T_{16}}^{>}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of greater than $(\theta = ' >')$. The purpose of tube ${T_{16}}^{=}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of equal to $(\theta = ' = ')$. The purpose of tube ${T_{16}}^{<}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of less than $(\theta = ' < ')$. The purpose of tube ${T_{16}}^{>=}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of greater than or equal to $(\theta = ' \geq ')$. The purpose of tube ${T_{16}}^{<=}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of less than or equal to $(\theta = ' \leq ')$. The purpose of tube ${T_{16}}^{\neq}$ is to store DNA strands encoding those elements in R have a result that satisfies the condition of not equal to $(\theta = ' \neq ')$.

```
Procedure Selection
$(T_0, {T_{16}}^{>}, {T_{16}}^{=}, {T_{16}}^{<}, {T_{16}}^{\neq}, {T_{16}}^{\geq}, {T_{16}}^{\leq}, D, E)$
1) Amplify $(T_0, T_{13}, T_{14})$.
2) $T_0 = \cup(T_0, T_{13})$.
3) For $j = 1$ to $L_k$
        3a) Append-Tail $(T_{15}, e_j)$.
    EndFor
4) For $j = 1$ to $L_k$
        4a) ${T_{15}}^{ON} = +(T_{15}, {e_j}^1)$ and ${T_{15}}^{OFF} = -(T_{15}, {e_j}^1)$.
        4b) ${T_{14}}^{ON} = +(T_{14}, {v_{i,k,j}}^1)$ and ${T_{14}}^{OFF} = -(T_{14}, {v_{i,k,j}}^1)$.
        4c) If (Detect $({T_{15}}^{ON}$ = "yes") then
            4d) ${T_9}^{=} = \cup({T_9}^{=}, {T_{14}}^{ON})$.
            4e) ${T_9}^{<} = \cup({T_9}^{<}, {T_{14}}^{OFF})$.
            4f) $T_{15} = \cup(T_{15}, {T_{15}}^{ON})$.
          Else
            4g) ${T_9}^{>} = \cup({T_9}^{>}, {T_{14}}^{ON})$.
            4h) ${T_9}^{=} = \cup({T_9}^{=}, {T_{14}}^{OFF})$.
            4i) $T_{15} = \cup(T_{15}, {T_{15}}^{OFF})$.
          EndIf
        4j) $T_{14} = \cup(T_{14}, {T_9}^{=})$.
    EndFor
5) ${T_9}^{=} = \cup({T_9}^{=}, T_{14})$.
6) Amplify $({T_9}^{>}, {T_{16}}^{>}, {T_{17}}^{>})$.
7) Amplify $({T_9}^{=}, {T_{16}}^{=}, {T_{17}}^{=})$.
8) Amplify $({T_9}^{<}, {T_{16}}^{<}, {T_{17}}^{<})$.
9) Amplify $({T_{17}}^{>}, {T_{18}}^{>}, {T_{19}}^{>})$.
10) Amplify $({T_{17}}^{=}, {T_{18}}^{=}, {T_{19}}^{=})$.
11) Amplify $({T_{17}}^{<}, {T_{18}}^{<}, {T_{19}}^{<})$.
12) ${T_{16}}^{>=} = \cup({T_{18}}^{>}, {T_{18}}^{=})$.
13) ${T_{16}}^{<=} = \cup({T_{18}}^{<}, {T_{19}}^{=})$.
14) ${T_{16}}^{\neq} = \cup({T_{19}}^{>}, {T_{19}}^{<})$.
15) In light of the selected condition that is one of $=, >, <, \neq, \geq$ and $\leq$, the read operation is used to read the answer(s) from the corresponding tube of tubes ${T_{16}}^{>}$, ${T_{16}}^{=}$, ${T_{16}}^{<}$, ${T_{16}}^{\neq}$, ${T_{16}}^{\geq}$ and ${T_{16}}^{\leq}$.
EndProcedure
```

Lemma 3–8: The selection operator on an n-ary relation can be performed with DNA strands from the algorithm, Selection $(T_0, {T_{16}}^{>}, {T_{16}}^{=}, {T_{16}}^{<}, {T_{16}}^{\neq}, {T_{16}}^{\geq}, {T_{16}}^{\leq}, D, E)$.

Proof: Please refer to the proof of Lemma 3–6 through Lemma 3–7. ■

F. DNA Algorithms for the Theta-Join Operator in Biomolecular Databases

Assume that $R1$ and $R2$ are n-ary relations and have p elements and q elements, respectively. The *theta-join* operator, cited from [4], is applied to produce, from the *Cartesian product* of $R1$ and $R2$, a new n-ary relation. The tuples in the resulting n-ary relation are those that satisfy some selected condition P that involves the columns of $R1$ and $R2$. The theta-join operator on $R1$ and $R2$ is denoted as $R1 \infty_P R2$, where the selected condition P is denoted in Section III-E. From [4], expression $R1 \infty_P R2$ is actually equal to expression $\sigma_P(R1 \times R2)$. This is to say that the theta-join operator on $R1$ and $R2$ can be performed through the *Cartesian product* and the selection operator. The following DNA algorithms are employed to perform the expression $\sigma_P(R1 \times R2)$, and notations used in the following DNA algorithms are denoted in Section III-B.

```
Procedure CartesianProductTwoRelations $(T_{51}, T_{52})$
1) For $i = 1$ to $q$
2) For $k = 1$ to $n$
3) For $j = 1$ to $L_k$
            3a) ${T_{52}}^{ON} = +(T_{52}, {v_{i,k,j}}^1)$ and ${T_{52}}^{OFF} = -(T_{52}, {v_{i,k,j}}^1)$.
            3b) If (Detect $({T_{52}}^{ON}$ = "yes") then
                    3c) Append-Tail $(T_{51}, {v_{i,k,j}}^1)$.
                EndIf
            3d) If (Detect $({T_{52}}^{OFF}$ = "yes") then
                    3e) Append-Tail $(T_{51}, {v_{i,k,j}}^0)$.
                EndIf
            3f) $T_{52} = \cup({T_{52}}^{ON}, {T_{52}}^{OFF})$.
        EndFor
    EndFor
  EndFor
```

Lemma 3–9: The Cartesian product on two n-ary relations can be performed with DNA strands from the algorithm, CartesianProductTwoRelations (T_{51}, T_{52}).

Proof: Please refer to Lemma 3–6 through Lemma 3–7. ■

```
Procedure Theta-join $(T_{50})$
1) CartesianProduct $(T_{53}, p)$.
2) CartesianProduct $(T_{54}, q)$.
3) Amplify $(T_{53}, T_{51}, T_{55})$.
4) Amplify $(T_{54}, T_{52}, T_{56})$.
```

```
5) $T_{53} = \cup(T_{53}, T_{55})$.
6) $T_{54} = \cup(T_{54}, T_{56})$.
7) CartesianProductTwoRelations $(T_{51}, T_{52})$.
8) Selection $(T_{51}, T_{51}{}^{>}, T_{51}{}^{=}, T_{51}{}^{<}, T_{51}{}^{\neq}, T_{51}{}^{\geq}, T_{51}{}^{\leq}, D, E)$.
9) $T_{50} = \cup(T_{50}, T_{51})$.
10) Read $(T_{50})$.
EndProcedure
```

Lemma 3–10: The theta-join operator on two n-ary relations can be performed with DNA strands from the algorithm, Theta-join (T_{50}).

Proof: Please refer to Lemma 3–6 through Lemma 3–7. ■

G. DNA Algorithms for the Division Operator in Biomolecular Databases

Assume that relations $R3$ and $R4$ have columns $(A1, \ldots, Aw, B1, \ldots Bz)$ and $(B1, \ldots, Bz)$, respectively. Columns $B1, \ldots, Bz$ are common to the two relations, $R3$ additionally has columns $A1, \ldots, Aw$ and $R4$ has no other columns. Moreover, suppose that the domain of every column comes from S_k (denoted in Section III-B) for $1 \leq k \leq n$ and the corresponding columns (i.e., columns with the same name) are defined on the same domain. Assume that relations $R3$ and $R4$ have p elements and q elements, respectively. The expression of the division operator on relations $R3$ and $R4$ is denoted as $R3 \div R4$, where relations $R3$ and $R4$ represent the dividend and the divisor, respectively. From [4], expression $R3 \div R4$ is actually equal to $\pi_{A1,\ldots,Aw}(R3) - \pi_{A1,\ldots,Aw}((\pi_{A1,\ldots,Aw}(R3) \times R4) - R3)$. This implies that the division operator on relations $R3$ and $R4$ can be accomplished through the projection operator, difference operator, and the Cartesian product. The following DNA algorithms are employed to perform the expression $\pi_{A1,\ldots,Aw}(R3) - \pi_{A1,\ldots,Aw}((\pi_{A1,\ldots,Aw}(R3) \times R4) - R3)$, and notations used in the following DNA algorithms are denoted in Section III-B.

```
Procedure Division $(T_{60})$
1) CartesianProduct $(T_{63}, p)$.
2) CartesianProduct $(T_{64}, q)$.
3) Amplify $(T_{63}, T_{67}, T_{65})$.
4) Amplify $(T_{64}, T_{68}, T_{66})$.
5) $T_{63} = \cup(T_{63}, T_{65})$.
6) $T_{64} = \cup(T_{64}, T_{66})$.
7) Projection $(T_{67}, T_{61}, p, w)$.
8) CartesianProductTwoRelations $(T_{61}, T_{66})$.
9) Difference $(T_{61}, T_{67}, T_{69}, p)$.
10) Projection $(T_{69}, T_{70}, p^{*}q, w)$.
11) Projection $(T_{67}, T_{71}, p, w)$.
12) Difference $(T_{71}, T_{70}, T_{60}, p^{*}q)$.
13) Read $(T_{60})$.
EndProcedure
```

Lemma 3–11: The division operator on relations $R3$ and $R4$ can be implemented with DNA strands from the algorithm, Division (T_{60}).

Proof: Please refer to Lemma 3–6 through Lemma 3–7. ■

H. Index Technology and Primary Key in Biomolecular Databases

For an n-ary relation R denoted in Section III-A, if the values of a column or combination of columns for any two rows are different, then the column or combination of the columns is called a *primary* key [4]. An *index* is usually defined on a single field of a file, called an indexing field. The index typically stores each value of the index field along with a list of pointers to all disk blocks that contain a record with that field value. The values in the index are ordered so that we can do a binary search on the index [4]. The following DNA algorithm is applied to construct an n-ary relation R denoted in Section III-A with a primary key $(S_1, \ldots, S_d)$, where $S_1, \ldots, S_d$ are all its domains and assume that d is the number of columns for the primary key. The notations in the following DNA algorithm are denoted in Section III-B.

```
Procedure PrimaryKeyDetect $(T_0)$
(0) For $i = 1$ to $m$
1) If (Detect $(T_0)$ == "no") Then
        1a) Insert $(T_{80}, i)$.
        1b) $T_0 = \cup(T_0, T_{80})$.
    Else
        2) For $k = 1$ to $n$
            3) For $j = 1$ to $L_k$
                3a) Append-Tail $(T_{82}, v_{i,k,j})$.
            EndFor
        EndFor
        4) For $k = 1$ to $d$
            5) For $j = 1$ to $L_k$
                5a) $T_{82}{}^{ON} = +(T_{82}, v_{i,k,j})$ and
                $T_{82}{}^{OFF} = -(T_{82}, v_{i,k,j})$.
                5b) $T_0{}^{ON} = +(T_0, v_{i,k,j})$ and
                $T_0{}^{OFF} = -(T_0, v_{i,k,j})$.
                5c) If (Detect $(T_{82}{}^{ON})$ == "yes") Then
                5d) $T_{83}{}^{=} = \cup(T_0{}^{ON}, T_{83}{}^{=})$ and
                $T_{83}{}^{\neq} = \cup(T_0{}^{OFF}, T_{83}{}^{\neq})$.
                    Else
                5e) $T_{83}{}^{=} = \cup(T_0{}^{OFF}, T_{83}{}^{=})$ and
                $T_{83}{}^{\neq} = \cup(T_0{}^{ON}, T_{83}{}^{\neq})$.
                    EndIf
                5f) $T_{82} = \cup(T_{82}{}^{ON}, T_0{}^{OFF})$
                and $T_0 = \cup(T_0, T_{83}{}^{=})$ and
                $T_{84} = \cup(T_{84}, T_{83}{}^{\neq})$.
            EndFor
        EndFor
        6) If (Detect $(T_0)$ == "no") Then
                6a) Insert $(T_{80}, i)$.
                6b) $T_0 = \cup(T_0, T_{80}, T_{84})$
            Else
                6c) Terminate the algorithm because input
                data are duplicated.
            EndIf
        EndIf
EndFor
EndProcedure
```

Lemma 3–12: A biomolecular database R with a primary key $(S_1, \ldots, S_d)$ can be constructed with DNA strands from the algorithm, PrimaryKeyDetect (T_0).

Proof: Please refer to Lemma 3–1 through Lemma 3–2.

IV. Discussions on Strengths and Weaknesses of Constructing DNA Relational Databases

From Lemma 3–1 through Lemma 3–12, one DNA strand with $(\theta \times (\sum_{k=1}^{n} L_k))$ base pairs is used to encode one element (record) in an n-ary relation R with m records (elements). If for a record in R its length is 4096 bytes that is equal to 32 768 bits, then we need to design one new DNA strand with $(\theta \times 32\,768)$ base pairs. From a biological standpoint, all sequences generated to represent 32 768 bits must be checked to ensure that the DNA strands that they encode do not form unwanted secondary structures with one another (i.e., strands remain separate at all times, and only bind together when this is required). The biggest challenge of constructing a biomolecular relational database is actually to the problem of strand design that has been addressed at length to minimize the possibility of unwanted binding.

With current biotechnology, the time for each operation is at least one second. Realistically, steps like gel electrophoresis take much longer, but for the sake of argument say each biological operation takes one second. It is assumed that the value of m is equal to 10^{10}. From CartesianProduct (T_0, m), constructing an n-ary relation with 10^{10} elements needs to take 10^{10} seconds which are about 317 years. Because $10^{10} \times 10^{10}$ elements are processed by Union (T_1, T_2, T_3, p), Intersection (T_1, T_2, T_4, p), Difference (T_1, T_2, T_5, q), Projection (T_0, T_6, m, c), Theta-join (T_{50}) and Division (T_{60}), it also takes about 10^{10} seconds to obtain the results. Even if an electronic supercomputer is used to construct 10^{10} elements and to complete those functions to $10^{10} \times 10^{10}$ elements, it is impossible to complete them in minutes or hours. If the value of m is equal to 10^{20}, then for a molecular computer and an electronic supercomputer constructing an n-ary relation with 10^{20} elements is a very difficult task and completing those functions on $10^{20} \times 10^{20}$ elements is also a very difficult task.

References

[1] L. M. Adleman, "Molecular computation of solutions to combinatorial problems," *Science*, vol. 266, no. 11, pp. 1021–1024, 1994.

[2] M. Amos, *Theoretical and Experimental DNA Computation*. New York: Springer, 2005.

[3] W.-L. Chang and A. V. Vasilakos, *Molecular Computing: Towards a Novel Computing Architecture for Complex Problem Solving*. New York: Springer, 2014.

[4] E. F. Codd, "A relational model of data for large shared data banks," *Commun. ACM*, vol. 13, no. 6, pp. 377–387, 1970.

Weng-Long Chang received the Ph.D. degree in computer science and Information Engineering from National Cheng Kung University, Taiwan, in 1999. He is current Professor at the Department of Computer Science and Information Engineering in National Kaohsiung University of Applied Sciences, Taiwan. His researching interests include biological algorithms, quantum algorithms, quantum-molecular algorithms, data structures and algorithms, and languages and compilers for parallel computing.

Athanasios V. Vasilakos is recently Professor at Kuwait University. He served or is serving as an Editor for many technical journals, such as the IEEE Transactions on Network and Service Management; IEEE Transactions on Cloud Computing; IEEE Transactions on Information Forensics And Security; IEEE Transactions On Cybernetics; IEEE Transactions on NanoBioscience; IEEE Transactions on Information Technology in Biomedicine; ACM *Transactions on Autonomous and Adaptive Systems*; the IEEE Journal on Selected Areas in Communications. He is also General Chair of the European Alliances for Innovation (http://www.eai.eu).