A Tissue P System and a DNA Microfluidic Device for Solving the Shortest Common Superstring Problem

Lucas LEDESMA, Daniel MANRIQUE
Alfonso RODRÍGUEZ-PATÓN, Andrés SILVA

Universidad Politécnica de Madrid
Facultad de Informática
Campus de Montegancedo s/n
Boadilla del Monte – 28660 Madrid, Spain
E-mail: `arpaton@fi.upm.es`

Abstract. This paper describes a tissue P system for solving the Shortest Common Superstring Problem in linear time. This tissue P system is well suited for parallel and distributed implementation using a microfluidic device working with DNA strands. The tP system is not based on the usual brute force generate/test technique applied in DNA computing, but builds the space solution gradually. The possible solutions/superstrings are build step by step through the parallel distributed combination of strings using the *overlapping concatenation* operation. Moreover, the DNA microfluidic device solves the problem autonomously, without the need of external control or manipulation.

1 Introduction

A microfluidic device, microflow reactor or, alternatively, "lab-on-a-chip" (LOC) are different names of micro devices composed basically of microchannels and microchambers. These are passive fluidic elements, formed in the planar layer of a chip substrate, which serve only to confine liquids to small cavities. Interconnection of channels allows the formation of networks along which liquids can be transported from one location of the device to another, where controlled biochemical reactions take place in a shorter time and more accurately than in conventional laboratories. There are also 3D microfluidic systems with a limited number of layers. See [18, 16] for an in-depth study of microflow devices.

The process of miniaturization and automation in analytical chemistry was an issue first addressed in the early 1980s, but the design (year 1992) of the device called "Capillary electrophoresis on a chip" [10] was the first important miniaturized chemical analysis system. Though still in its infancy, the interest in this technology has grown explosively over the last few years.

Microelectromechanical (MEMS) technologies are now being widely applied to microflow devices to fabricate microchannels in different substrate materials, integrate electrical function into microfluidic devices, and develop valves, pumps, and sensors. A thorough review of many different microfluidic devices for nucleic acid analysis (for example, chemical amplification, hybridization, separation, and detection) is presented in [16].

These microfluidic devices can implement a dataflow-like architecture for processing DNA (see [6] and [12]) and could be a good support for the distributed biomolecular computing model called tissue P systems (or tP systems for short) [11]. The underlying computational structure of tP systems are graphs or networks of connected processors that could easily be translated to microchambers (cells or processors) connected with microchannels.

There are several previous works on DNA computing using microfluidic systems [6, 12, 4, 8, 9]. One [8] describes the design of a linear time DNA algorithm for the Hamiltonian Path Problem (HPP) suited for parallel implementation using a microfluidic system (this bioalgorithm shares some features with the algorithm for the Shortest Common Superstring Problem presented in this paper). This algorithm [8] builds the space solution gradually. The possible solutions/paths are built step by step by exploring the graph according to a breadth-first search so that only the paths that represent permutations of the set of vertices, and which, therefore, do not have repeated vertices (a vertex is only added to a path if this vertex is not already present) are extended. This simple distributed DNA algorithm has only two operations: concatenation (append) and sequence separation (filter). The HPP is resolved autonomously by the system, without the need for external control or manipulation. In another paper, Gehani and Reif [6] study the potential of biomolecular microflow computation, describe methods to efficiently route strands between chambers, and determine theoretical lower bounds on the quantities of DNA and the time needed to solve a problem in the microflow biomolecular model. Two other works [12, 4] solve the Maximum Clique Problem with microfluidic devices. This is an NP-complete problem. McCaskill [12] takes a brute-force approach codifying every possible subgraph in a DNA strand. The algorithm uses the so-called Selection Transfer Modules (STM) to retain all possible cliques of the graph. The second step of McCaskill's algorithm is a sorting procedure to determine the maximum clique. By contrast, Chiu *et al.* [4] describes a novel approach that uses neither DNA strands nor selection procedures. Subgraphs and edges of the graph are hard codified with wells and reservoirs, respectively, connected by channels and containing fluorescent beads. The readout is a measure of the fluorescence intensities associated with each subgraph. The weakness of this latter approach is an exponential increase in the hardware needed with the number of vertices of the graph. Finally, Livstone and Landweber [9] propose the application of microreactors to implement Boolean functions AND and OR (connecting microreactors in series and in parallel) but without finding a possible implementation for the NOT function.

1.1 The Problem

The problem chosen for our work is the Shortest Common Superstring Problem (SCSP). The input for this problem is a finite set of n strings x_i and an integer K. The output of the problem is "yes", if there exists a superstring of length at most K and "no" otherwise. A superstring over the given set is a string that contains all the strings in the set as substrings (a formal definition of the problem is given later in section 2).

There are several reasons for the choice of the SCSP. First, the problem is NP-complete [5]. This means that large instances of the SCSP cannot be solved efficiently by electronic computers because running time escalates exponentially as problem size grows. Second, being strong related to DNA sequencing methods like SBH (sequencing by hybridization), this problem is of biological interest. The aim of the SBH technique is to reconstruct the sequence of a long DNA strand from the knowledge of the sequence of its short overlapping substrings or fragments. Microarray chips technology was first developed [3] for this purpose. The computational problem often used to model this process of DNA fragment assembly is SCSP. A more detailed information about the relation of SBH to SCSP is given in [1], and a novel and promising application of SBH to resequencing is described in [15]. Third, SCSP is also applied in data compression schemes.

2 The Shortest Common Superstring Problem

In this section we give a formal definition of the problem.

Input: Alphabet Σ, finite set $S = \{x_1, x_2, \dots, x_n\}$, $n \geq 1$, of strings from Σ^*, and a positive integer K.

Question: Is there a string $w \in \Sigma^*$ with length $|w| \leq K$ such that each string $x_i \in S$, $1 \leq i \leq n$, is a substring of w, i.e., for all i, $1 \leq i \leq n$, $w = u_i x_i v_i$, where u_i, $v_i \in \Sigma^*$?

The problem remains *NP-complete* even if the cardinality of Σ is 2.

3 Previous Work

The only previous solution to the Shortest Common Superstring Problem in the DNA computing field is [7]. This DNA algorithm follows a brute-force approach:

Step 1. Encode all the strings $\{x_i, x_2, \dots, x_n\}$ of the set S in DNA strands.

Step 2. Generate all possible DNA strands w of length k between $max\{|x_i|, 1 \leq i \leq n\}$ and K using T, G, A residues only.

Step 3. Let x_1 be a string of S. For the string population generated in *Step* 2, select only those strands that contain x_1 as a substring. From the newly obtained string population, select only those strings that contain $x_2 \in S$ as substring, etc. Repeat this step for each $x_i \in S$, $1 \leq i \leq n$.

Step 4. If, after step 3, there is any strand w remaining (which means that w contains all $x_i \in S, 1 \leq i \leq n$, as substrings), say "yes", otherwise say "no".

This algorithm builds $\sum_{k=max\{|x_i|\}}^{k=K} 3^k$ different DNA strands encoding candidate solutions and then retains only those strings that contain all the x_i as substrings. For example, for a set $S = \{x_1, x_2, \dots, x_n\}$ where $max\{|x_i|\} = 4$ and for $K = 25$ we would need to generate $3^4 + 3^5 + \ldots + 3^{25} = 1.27 \times 10^{12}$ DNA strands. And for the same set, if we choose $K = 37$, we would need to generate $3^4 + 3^5 + \ldots + 3^{37} = 6.75 \times 10^{17}$ DNA strands. This exponential increase in molecular resources places an obvious in practice upper limit on this approach. Remember that around 10^{18} DNA strands can be handled in a test tube.

4 A Tissue P system for the SCSP

4.1 Overlapping Concatenation

Before presenting the tP system, we explain the notation used in this paper. Given two strings x and y in Σ^*, the overlapping concatenation between them, denoted by $x \odot y$, will be described as follows:

$$x \odot y = \begin{cases} x, & \text{if } x = x'_1 y x'_2, \text{ where } x'_1 \text{ and } x'_2 \in \Sigma^*, \\ y, & \text{if } y = y'_1 x y'_2, \text{ where } y'_1 \text{ and } y'_2 \in \Sigma^*, \\ x'uy', & \text{where } x = x'u,\ y = uy' \text{ and } |u| \text{ is the longest overlap.} \end{cases}$$

From the above definition it follows that: for every x, $x \odot x = x$ and for every x, y, $|x \odot y| = |x| + |y| - |u| = |x \cdot y| - |u|$. If $u = \lambda$, then the overlapping concatenation is the usual concatenation: $x \odot y = x \cdot y$.

- Example 1: Given $\Sigma_1 = \{a, b, c, d, r\}$, and the strings $x = abraca$, $y = cadabra \in \Sigma_1^*$, $x \odot y = x'uy' = abra\underline{ca}dabra$, where $x' = abra$, $y' = dabra$ and$u = ca$.
- Example 2: Given $\Sigma_1 = \{a, b, c, d, r\}$, and the strings $x = abracadabra$, $y = acada \in \Sigma_1^*$, $x \odot y = x'_1 y x'_2 = abr\underline{acada}bra$, where $x'_1 = abr$, $x'_2 = bra$.

The operation $\odot$ is extended naturally to the languages L_1 and L_2:

$$L_1 \odot L_2 = \{x \odot y \ |x \in L_1 \wedge y \in L_2\}.$$

The overlapping concatenation operation described above models the parallel DNA overlapping assembly operation. This operation has been used previously in DNA computing (see [13]) to generate DNA data pools and in combinatorial chemistry (see [17]). Here it will be used to implement the proposed tP system using DNA microfluidic devices.

4.2 Tissue P Systems

We briefly and informally review the bioinspired computational model called tP system. A detailed description is given in [11] and in [14].

A tP system is a network of finite automata-like processors, dealing with multisets of symbols, according to local states (available in a finite number for each "cell"), communicating through these symbols, along channels ("axons") specified in advance. Each cell has a state from a given finite set and can process multisets of *objects*, represented by symbols of a given alphabet. The standard evolution rules are in the form $sM \to s'M'$, where s, s' are states and M, M' are multisets of symbols. We can apply such a rule to only one occurrence of M (that is, in a sequential *minimal* way), or to all possible occurrences of M (a *parallel* way), or, moreover, we can apply a maximal package of rules of the form $sM_i \to s'M'_i, 1 \leq i \leq k$, that is, involving the same states s, s', which can be applied to the current multiset (the *maximal* mode). Some of the elements of M' may be marked with the indication "go", and this means that they have to immediately leave the cell and pass to the cells to which there are direct links through synapses. This communication (transfer of symbol-objects) can be done in a *replicative* manner (the same symbol is sent to all adjacent cells), or in a *non-replicative* manner; in the second case, we can send all the symbols to just one adjacent cell, or we can distribute them non-deterministically.

One way to use such a computing device is to start from a given initial configuration and to let the system proceed until it reaches a halting configuration and to associate a result with this configuration. A halting computation is a computation that ends in a configuration where no rule in any cell can be used. In these *generative* tP systems the output will be defined by sending symbols out of the system. To this end, one cell will be designated as the output cell.

Within this model, we present here a tP system (working in a maximal mode and using the replicative communication mode) to solve the Shortest Common Superstring Problem. As we already mentioned, this problem involves determining whether or not, for a given set of strings $S = \{x_1, x_2, \ldots, x_n\}$ from Σ^* and a positive integer K, there is a string w of length $|w| \leq K$ containing all the strings in S as substrings. Our tP system differs from the general version of a tP system as to two features: (1) our tP system deals with strings (not symbols) that evolve through the $\odot$ operation, and (2) our tP system does not have a single output cell (strings can be sent out from every cell of our tP system). Not only does the tP system presented solves the SCSP but it can also be used to find the minimum value of K for which the problem has an "yes" solution.

This tP system has one cell σ_i associated with each string x_i of S and the cells are fully connected. Starting from each x_i, overlapping concatenations of the strings in S grow simultaneously in all cells, and in parallel. Strings $z \odot x_i$ are produced in each cell σ_i and are communicated to the other cells. Strings z are the inputs of the cells and x_i are the string associated with each cell σ_i. It is easy to see that, after $n-1$ steps, the strings built by the system contain as substrings the x_i, for all $1 \leq i \leq n$. Moreover, after $n-1$ steps, the system has built $n!$ superstrings z where:

1. Each x_i, $1 \leq n \leq n$, appears only once in each z,
2. the $n!$ superstrings z correspond to the $n!$ parallel combinations of $x_i \in S$ using the overlapping concatenation $\odot$,
3. each superstring z is sent out of the system only if its length is $|z| \leq K$.

Formally, the tP system proposed to solve the Shortest Common Superstring Problem for a set S of n strings $x_i \in \Sigma^*$ and a value K is

$$\Pi = (O, \sigma_1, \ldots, \sigma_n, syn),$$

where:

1. $O = \{[z; l] \mid z \in \Sigma^*, 1 \leq l \leq n\}$
2. $syn = \{(i, j) \in \{1, 2, \ldots, n\} \times \{1, 2, \ldots, n\} \mid i \neq j\}$ is the set of channels among cells; each cell is connected to every other cells.
3. $\sigma_1, \sigma_2, \ldots, \sigma_n$ are cells of form
   σ_i = ({s}, s, [x_i;1], P_i) for each $i = 1, 2, \ldots, n$

   P_i **:** $s[z; l] \to s[z \odot x_i; l+1]_{go}$ such that $z \in \Sigma^*$, $|z|_{x_i} = 0, \quad 1 \leq l \leq n-2,$
   $s[z, n-1] \to s[z \odot x_i; n]_{out}, \quad$ if $|z \odot x_i| \leq K$.

The candidate superstrings z grow simultaneously in all cells of Π, because of the *max* mode of using the rules (each cell has only one state, hence all rules can be used at the same time). Moreover, the rule $s[z, n-1] \to s[z \odot x_i; n]_{out}$, if $|z \odot x_i| \leq K$, can only work after $n-2$ steps and only the superstrings of length less or equal to K are sent out of the system at step $n-1$. It then suffices to look at the tP system after step $n-1$, where, if any string is sent out of the system, then the answer to the problem will be "yes"; "no" otherwise. The shortest common superstrings will be the strings sent out of the system. Furthermore, the system can be executed for different values of K between $max\{|x_i|\}$ and $\sum |x_i|$, $1 \leq i \leq n$, to find the minimum K for which there exists a superstring over S.

5 A DNA Algorithm Using a Microfluidic Device

The tP system described above can be easily translated to a parallel and distributed DNA algorithm to be implemented in a microfluidic system. Remember that we have a set of strings $S = \{x_1, x_2, \ldots, x_n\}$ from Σ^* and an integer K, and we want to determine whether or not there is a superstring w with length $|w| \leq K$. Here we assume without loss of generality that no string $x_i \in S$ is a substring of any other $x_j \in S$. Our algorithm finds the solution to the SCSP in linear time. First, we describe the overall operation of the algorithm.

The proposed algorithm constructs the $n!$ superstrings of S corresponding to the $n!$ combinations of $x_i \in S$ using the overlapping concatenation. For this purpose, the hardware of the microfluidic system consists of n nodes n_i. In each node n_i, in each time unit, the overlapping concatenation of the string x_i associated with the node n_i and the input string z is obtained, $z \odot x_i$, and then communicated to the others nodes of the microsystem.

Each node n_i consists of three chambers E_i, C_i, and F_i. Chamber E_i is used to extend the input strings z when there is overlapping between z and the x_i associated to the node n_i. Chamber C_i is used to append the string x_i to the input string z when no overlapping is present. And the last chamber F_i is used: (1) to eliminate the input strings z that have already substring x_i and (2) to route the input strings z to the corresponding concatenation C_i or extension E_i chamber.

Thus, in each node, in each step, the string associated with that node is combined with the input string using the overlapping concatenation and then communicated to the other nodes. In step $n - 1$, the system has assembled $n!$ superstrings corresponding to the $n!$ possible combinations of x_i described above.

We now present the algorithm:

- *Coding*: Each string x_i is encoded with a different short single DNA strand. Each of these single strands must be carefully selected to avoid mismatch hybridizations throughout the bioalgorithm run (see [2] for DNA strands design criteria).

- In parallel, in $(n - 1)$ steps, starting from each string in S the algorithm grows $n!$ superstrings. The $x_i \in S$ are combined using the overlapping concatenation $\odot$ in all possible ways (that is, the $n!$ ways corresponding to the $n!$ permutations of n).

- Time $t = n$. Read the outputs of the system. At time $t = n$, the $n!$ strings assembled by the algorithm will be superstrings. Then it merely remains to choose the shortest strings and determine whether its length is less than or equal to K, in which case the answer to the problem is "yes", otherwise the answer is "no".

Now we propose a microfluidic system to implement the DNA algorithm. The *hardware* of our system for a set S with n strings is composed of $3n+1$ chambers: n filtering chambers F_i, n append/concatenation chambers C_i, n extension chambers E_i, and one chamber L to determine the length of the superstrings.

Chamber F_i. Input: DNA strands z from chambers C_j and E_j with $1 \leq i, j \leq n$ and $j \neq i$. Function: Separate the input strands into three groups: (1) the first group is composed of those input strands that do not hybridize to the strand associated with $\overline{x_i}$ (the strand associated with $\overline{x_i}$ is the complementary strand of the strand associated with x_i); (2) the second group is composed of those input strands that hybridize partially on the left extreme of the strand associated with $\overline{x_i}$; (3) and the third group is composed of those input strands that fully hybridize to the strand associated with $\overline{x_i}$. Only the strands in the first two groups (1) and (2) will be valid outputs of the chamber. Those outputs will be called $output_1$ and $output_2$ respectively. The third group of strands is discarded or removed.

Chambers C_i. Input: DNA strands z from the $output_1$ of F_i. Function: Appends the substrand associated with x_i to the right of each strand z in its input. Output: $z \odot x_i = z \cdot x_i$ (remember that, in this case, the $\odot$ operation is the usual concatenation, because there is no overlapping between z and x_i).

Chambers E_i. Input: DNA strands z from the $output_2$ of F_i. Function: Extends every strand z in its input to produce $z \odot x_i$, using the parallel overlapping assembly operation with $\overline{x_i}$ as extension pattern. Output: $z \odot x_i$.

Chamber L. Inputs: DNA strands z representing superstrings of S sent out of the C_i and E_i chambers in step $n-1$. Function: determine the length of the different z.

Pattern of connectivity (layout). The $output_1$ of each chamber F_i is connected to the input of the associated E_i. The $output_2$ of each chamber F_i is connected to the input of the associated C_i. For every $1 \leq i, j \leq n$ and $i \neq j$, there is a channel from the outputs of chambers C_i and E_i to the input of every chamber F_j; and there is a pump that forces the flow of the liquid from C_i and E_i to F_j.

For simplicity's sake, we group the chambers by their subindexes in nodes. Thus, $node_i$ will be composed of chambers F_i, C_i, and E_i. The input of $node_i$ will be the input of chamber F_i and the output of $node_i$ will be the union of the outputs of E_i and C_i.

Implementation. It is beyond the scope of this paper to give more details of the possible implementation of these microsystems. We merely indicate that the filtering, append, and extend operations are widely used in DNA computing and a detailed description of the microfluidic devices is given in [6, 16].

Working (dynamics) of the system. We assume that the operations of $node_i$ (filter and extension or append) take one time unit.

- Step 0: $(t = 0)$ *(pre-loading).*
  - Put enough copies of the strand associated with $\overline{x_i}$ into each chamber F_i.
  - Put enough copies of the strand associated with x_i, and enough copies of the auxiliary strands and enzymes to allow concatenation into each chamber C_i.
  - Put enough copies of the strand associated with $\overline{x_i}$ and enough enzymes and nucleotides to allow the extension into each chamber E_i.
- Steps 1 to $(n-1)$: (from $t = 1$ to $t = (n-1)$).
  - *Computations:* For all $1 \leq i \leq n$, in parallel, a filtering operation in all F_i, an append operation in all C_i, and an extension operation in all E_i are performed in each step.
  - *Movement* (pumping) of strands from step t to step $t+1$.

$$Input(F_{j_{(t+1)}}) = \bigcup_i (Output(C_{i_{(t)}}) \cup Output(E_{i_{(t)}}))$$

for all $1 \leq i, j \leq n$, and $i \neq j$. The inlet of chamber F_j in time $t+1$ is the union of the outlets of chambers C_i and E_i in time t for all $1 \leq i, j \leq n$, and $i \neq j$. Remember that we can group the chambers in nodes, thus

$$Input(node_{j_{(t+1)}}) = \bigcup_i Output(node_{i_{(t)}})$$

for all $1 \leq i, j \leq n$, and $i \neq j$. The inlet of the $node_j$ in time $t+1$ is the union of the outlets of nodes $node_i$ in time t.

- *Step* $t = n$ *(readout)*: After step $(n-1)$ collects the strands in the output of chambers C_i and E_i and determines their lengths. The length of the shortest output strands determines the minimum value of K for which there exists a superstring of S. The lengths of the strands can be calculated by gel electrophoresis with chamber L.

5.1 Example

We give an example of the execution of the algorithm for a set with four strings $S = \{x_1 = ABRAC, x_2 = ADABR, x_3 = DABRA, x_4 = RACAD\}$ constructed from the alphabet $\Sigma = \{A, B, R, C, D\}$. Table 1 shows the outputs of each node in each step. After time $t = 3 = n-1$, the strands in the output of all nodes have to be collected. These strands codify the 24 superstrings corresponding to the $24 = 4!$ possible ways of combining through overlapping concatenation the original x_i strings. The numbers that appear in front of each string are shown to ease the understanding how each string is assembled. These numbers show the order of combination of the x_i; for example, in time $t = 2$, the output of $node_1$ contains the string $321|DABRADABRAC = x_3 \odot x_2 \odot x_1 = DABRA \odot ADABR \odot ABRAC$. The last chamber of our microfluidic device is used to determine the length of the strands using gel electrophoresis. The strings in bold are the shortest common superstring, and their length is the minimum K for which there exists a superstring of S. In this example $K = 9$ is the minimum value of K for which the SCSP has a "yes" solution.

The number of DNA strands encoding candidate solutions constructed with our algorithm is $n! = 4! = 24$. To solve this example using the algorithm proposed in [7] approximately

$$\sum_{k=max\{|x_i|\}=5}^{k=K=20} 3^k \approx 5.2 \times 10^{10}$$

candidate solutions ($K = |ABRAC| + |ADABR| + |DABRA| + |RACAD| = 20$) would need to be generated.

6 Final Remarks

It was shown in [11] that tP systems are a useful computational model for solving complex graph-related problems. This paper presents a new application of tP systems outside the graph-related problem domain with interest in the field of genomics and in data compression. The new tP system and its implementation with a DNA microfluidic system solves in linear time the Shortest Common Superstring Problem (SCSP).

The algorithm for the SCSP runs autonomously, in linear time, in parallel, and without manual external intervention. There is only one previous work in the DNA computing field solving the SCSP problem [7]. Our algorithm does not follow the Adleman's style of

	$t=0$	$t=1$	$t=2$	$t=3$
$node_1$	$1\|ABRAC$	$21\|ADABRAC$	$321\|DABRADABRAC$	$4321\|RACADABRAC$
		$31\|DABRAC$	$421\|RACADADABRAC$	$3421\|DABRACADABR$
		$41\|RACADABRAC$	$231\|ADABRAC$	$4231\|RACADADABRAC$
			$431\|RACADABRAC$	$2431\|\mathbf{ADABRACAD}$
			$241\|ADABRACAD$	$3241\|DABRADABRACAD$
			$341\|DABRACAD$	$2341\|\mathbf{ADABRACAD}$
$node_2$	$2\|ADABR$	$12\|ABRACADABR$	$312\|DABRACADABR$	$4312\|RACADABRAC$
		$32\|DABRADABR$	$412\|RACADABRAC$	$3412\|DABRACADABR$
		$42\|RACADADABR$	$132\|ABRACDABRADABR$	$4132\|RACADABRAC$
			$432\|RACADABRA$	$1423\|ABRACADABRA$
			$142\|ABRACADABR$	$3142\|DABRACADABR$
			$342\|DABRACADABR$	$1342\|ABRACDABRACADABR$
$node_3$	$3\|DABRA$	$13\|ABRACDABRA$	$213\|ADABRAC$	$4213\|RACADADABRAC$
		$23\|ADABRA$	$413\|RACADABRAC$	$2413\|\mathbf{ADABRACAD}$
		$43\|RACADABRA$	$123\|ABRACADABRA$	$4123\|RACADABRAC$
			$423\|RACADADABRA$	$1423\|ABRACADABRA$
			$143\|ABRACADABRA$	$2143\|\mathbf{ADABRACAD}$
			$243\|ADABRACAD$	$1243\|ABRACADABRA$
$node_4$	$4\|RACAD$	$14\|ABRACAD$	$214\|ADABRACAD$	$3214\|DABRADABRACAD$
		$24\|ADABRACAD$	$314\|DABRACAD$	$2314\|\mathbf{ADABRACAD}$
		$34\|DABRACAD$	$124\|ABRACADABR$	$3124\|DABRACADABR$
			$324\|DABRADABRACAD$	$1324\|ABRACDABRADABRACAD$
			$134\|ABRACDABRACAD$	$2134\|\mathbf{ADABRACAD}$
			$234\|ADABRACAD$	$1234\|ABRACADABRA$

Table 1: Running of the algorithm. Output of the nodes at each step. The numbers that appear in front of each string are shown to ease the understanding of how each string is constructed. These numbers show the order of combination of the x_i. The strings in time $t = 3 = n - 1$ are the $24 = n! = 4!$ superstrings generated by the system. Strings in bold are the shortest common superstrings, and their minimum length ($K = 9$) is the minimum value of K for which there exists a superstring of S.

DNA computation (generate the entire solution space and sequentially filter the unfeasible solutions) used in this previous work. The tP system, and the respective implementation with a DNA bioalgorithm using a microfluidic device proposed in this paper, follows a constructive problem-solving strategy avoiding the generation of unfeasible solutions. Our bioalgorithm only generates feasible solutions of the SCSP prunning to a great extent the number of DNA strands used.

Microfluidic systems looks like an interesting and promising future support for many distributed DNA computing models (shift from Adleman/Lipton manual wet test tubes to DNA computing on surfaces to microflow DNA computing), and its full potential (the underlying computational paths could be a graph with cycles allowing, for example, the iterative construction and selection of solutions) needs to be thoroughly examined.

Acknowledgements. The work of the first author was supported by a grant of AECI (Agencia Española de Cooperación Internacional). The third authors was partially supported by Ministerio de Ciencia y Tecnología under project TIC2002-04220-C03-03, cofinanced by FEDER funds.

References

[1] J. Blażewicz and M. Kasprzak. Complexity of DNA sequencing by hybridization. *Theoretical Computer Science*, 290 (2003), 1459–1473.

[2] A. Brenneman and A. Condon. Strand design for bio-molecular computation, (Survey paper), *Theoretical Computer Science*, 287 (2002), 39–58.

[3] A. Caviani Pease, D. Solas, E.J. Sullivan, M.T. Cronin, C.P. Holmes, and S.P. Fodor. Light-generated oligonucleotide arrays for rapid DNA sequence analysis, *Proc. Nat. Acad. Sci.* U.S.A., 91 11 (May 24, 1994), 5022–5026.

[4] D.T. Chiu, E. Pezzoli, H. Wu, A.D. Stroock, and G. M. Whitesides. Using three-dimensional microfluidic networks for solving computationally hard problems. *Proc. Nat. Acad. Sci.* U.S.A., 98 6 (March 13, 2001), 2961–2966.

[5] J. Gallant, D. Maier, and J. Storer. On finding minimal length superstrings. *Journal of Computer and Systems Science*, 20, 1 (1980), 50–58.

[6] A. Gehani and J.H. Reif. Microflow bio-molecular computation. *Biosystems*, 52, 1-3 (1999), 197–216.

[7] G. Gloor, L. Kari, M. Gaasenbeek, and S. Yu. Towards a DNA solution to the shortest common superstring problem. In *4th Int. Meeting on DNA-Based Computing*, Baltimore, Penns., June, 1998. Also, *Proceedings of IEEE'98 International Joint Symposia on Intelligence and Systems*, Rockville, MD, (May 1998), 140–145.

[8] L.F. Ledesma, J. Pazos, and A. Rodríguez-Patón. A DNA algorithm for the Hamiltonian Path problem using microfluidic systems. *Aspects of Molecular Computing - Essays Dedicated to Tom Head on the Occasion of His 70th Birthday. Lecture Notes in Computer Science* 2950, Springer (2004), 289–296.

[9] M.S. Livstone and L.F. Landweber: Mathematical considerations in the design of microreactor-based DNA computers. In *DNA Computing, 9th International Workshop*

on DNA Based Computers, DNA9, Madison, WI, USA, June 1-3, 2003, revised papers. *Lecture Notes in Computer Science* 2943, Springer (2004), 180–189.

[10] A. Manz, D.J. Harrison, E.M.J. Verpoorte, J.C. Fettinger, A. Paulus, H. Ludi, and H.M. Widmer. Planar chips technology for miniaturization and integration of separation techniques into monitoring systems: "Capillary electrophoresis on a chip". *J. Chromatogr.*, 593 (1992), 253–258.

[11] C. Martín-Vide, Gh. Păun, J. Pazos, and A. Rodríguez-Patón. Tissue P systems. *Theoretical Computer Science*, 296, 2 (2003), 295–326.

[12] J.S. McCaskill. Optically programming DNA computing in microflow reactors. *Biosystems*, 59, 2 (2001), 125–138.

[13] Q. Ouyang, P. D. Kaplan, S. Liu, and A. Libchaber. DNA solution of the maximal clique problem. *Science*, 278 (1997), 446–449.

[14] Gh. Păun. *Membrane Computing. An Introduction.* Springer, Berlin, 2002.

[15] I. Pe'er, N. Arbili, and R. Shamir. A computational method for resequencing long DNA targets by universal oligonucleotide arrays. *Proc. Nat. Acad. Sci.* U.S.A., 99, 24 (November 26, 2002), 15492–15496.

[16] P.R. Selvaganapathy, E.T. Carlen, and C.H. Mastrangelo. Recent Progress in Microfluidic Devices for Nucleic Acid and Antibody Assays. *Proceedings of the IEEE*, 91 6 (2003), 954–973.

[17] W.P.C. Stemmer. The Evolution of Molecular Computation. *Science*, 270 (1995) 1510–1510.

[18] E. Verpoorte and N.F. De Rooij. Microfluidics Meets MEMS. *Proceedings of the IEEE*, 91 6 (2003), 930–953.