Solutions to Computational Problems through Gene Assembly *

Artiom Alhazov[2], Ion Petre[1,2], Vladimir Rogojin[2]

[1] Academy of Finland
[2] Computational Biomodelling Laboratory
Turku Center for Computer Science, FIN-20520 Turku, Finland
aalhazov@abo.fi,ipetre@abo.fi,vrogojin@abo.fi

Abstract. Gene assembly in ciliates is an impressive computational process. Ciliates have a unique way of storing their genetic information in two fundamentally different forms within their two types of nuclei. Micronuclear genes are broken into blocks (called MDSs), with MDSs shuffled and separated by non-coding material; some of the MDSs may even be inverted. During gene assembly, all MDSs are sorted in the correct order to yield the transcription-able macronuclear gene. Based on the intramolecular model for gene assembly, we prove in this paper that gene assembly may be used in principle to solve computational problems. We prove that any given instance of the hamiltonian path problem may be encoded in a suitable way in the form of an 'artificial' gene so that gene assembly is successful on that gene-like pattern if and only if the given problem has an affirmative answer.

1 Introduction

Ciliates are unicellular organisms existing for over a billion years, forming a group of thousands of species. A common feature they share is that their cell contains two kinds of nuclei that have different functionality - micronuclei act as germline nuclei and macronuclei act as the somatic nuclei.

During the sexual reproduction, the macronuclei are destroyed and one haploid micronucleus is transformed into a macronucleus. The gene operations have a definite computational flavor: some DNA segments (internally eliminated sequences, IES) are eliminated, others (macronuclear destined sequences, MDS) are reordered; some MDSs are also inverted. The process is driven by splicing on specific sequences on the ends of MDSs, called pointers: the end of each MDS matches the beginning of the MDS that should follow it in the assembled gene. Two main models exist for the gene assembly, one intermolecular, see [13, 14] and one intramolecular, see [8, 18]. In this article we consider the latter one.

In 1994 a famous experiment of L. Adleman took place giving an example how biological processes can be interpreted as computing (a small instance of

* A. Alhazov (artiom@math.md) and V. Rogojin are on leave of absence from Institute of Mathematics and Computer Science of Academy of Sciences of Moldova, Chisinau MD-2028 Moldova.

hamiltonian path problem, HPP was represented by DNA molecules and solved by molecular biology tools). The current paper considers replacing such DNA operations as annealing by ciliate operations, therefore, we speak about ciliate-based computing.

Indeed, what ciliates do in gene assembly is sorting, inversion and excision of DNA sequences. Therefore, our strategy is to encode an arbitrary instance of HPP into a *hypothetical* micronuclear gene, assemble the gene using the intramolecular model, and filter the result of the assembly to get the answer to HPP, if there is any.

This is a novel approach to DNA computing, using a model for gene assembly in ciliates. Although the computational flavor of ciliates has been shown previously in[13–15] where the Turing universality of various assembly models was proved, this is the first attempt at using (in principle) gene assembly for solving mathematical problems. If ever implemented in living cells, the solution potentially has the advantage that the cell itself implements many steps of the procedure, including selecting the resulting substring and its replication. It is important to underline that we only propose here a conceptual (theoretical) approach to ciliate-based computing. We only briefly discuss some issues related to potential experimental implementations of our approach in Section 8.

2 Definitions

For an alphabet Σ we denote by Σ^* the set of all finite strings over Σ. We denote the empty string by Λ. For strings u, v over Σ we say that u is a *substring* of v, denoted $u \leq v$, if $v = xuy$, for some strings x, y. Let $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$ be complement symbols of Σ; we call $u \in (\Sigma \cup \overline{\Sigma})^*$ a *signed string*. The complement operation is extended to signed strings by $\overline{\overline{a}} = a$, $a \in \Sigma$ and $\overline{a_1 a_2 \cdots a_k} = \overline{a_k} \cdots \overline{a_2}\ \overline{a_1}$, $a_i \in \Sigma \cup \overline{\Sigma}$, $1 \leq i \leq k$.

We call a (directed) *graph* a tuple $G = (V, E)$, where V is a finite set of nodes, and $E \subseteq \{(p, q) \mid p, q \in V\}$ is a set of edges. A sequence $q_1 q_2 \cdots q_k$ of nodes $q_i \in V$, $1 \leq i \leq k$ is called a path if $(q_i, q_{i+1}) \in E$, $1 \leq i \leq k-1$.

The *hamiltonian Path Problem* for a directed graph $G = (V, E)$, given the initial node p and a final node q is the problem of deciding whether G has an acyclic path from p to q containing all nodes of the graph (it is implicit in this definition that all nodes are visited only once). Such a path is called *hamiltonian*. The hamiltonian path problem is a known NP-complete problem, see [16].

For some results we need also the following graph construction, which we call *bipartite transformation*: given a graph $G = (V, E)$ we construct a graph $bi(G) = G' = (V', E')$ where $V' = \{p, p' \mid p \in V\}$ and $E' = \{(p', q) \mid (p, q) \in E\} \cup \{(p, p') \mid p \in V\}$. In other words, we split each node p in two nodes p and p' connected by an edge, and replace the edges (p, q) of the original graph by (p', q). This gives us a bipartite graph where every edge (p, q) in G corresponds to a path $pp'q$ in G'.

Consider a graph $G = (V, E)$ with $V = \{p_1, \cdots, p_n\}$ and the hamiltonian path problem from p_1 to p_n. Due to the technical reasons, throughout the paper

we use the following transformation: add two nodes $b, e \notin V$ to G and look for paths from b to e in graph $ext(G, p_1, p_n) = G'' = (V \cup \{b, e\}, E \cup \{(b, p_1), (p_n, e)\})$. Notice that the edge from b is unique and so is the edge to e, and there are no edges to b and no edges from e. Clearly, u is a path in G from p_1 to p_n if and only if bue is a path in G' from b to e. Therefore, this HPP is equivalent to the original one.

Example 1. For the graph $G_1 = (V_1 = \{1, 2, 3\}, E_1 = \{(2, 1), (3, 1), (3, 2)\})$ we illustrate in Figure 1 the graph $ext(G_1, 3, 1)$.

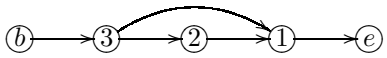

Fig. 1. A hamiltonian path in $ext(G_1, 3, 1)$ is $b321e$.

3 Gene assembly

The following three molecular operations are postulated in the intramolecular model to explain the gene assembly process, see [8] and [18]:

- *Loop, Direct Repeat* (ld) is applied on a pair of directly repeating pointers in the molecule. The molecule folds on itself to form a loop so that recombination is facilitated on the two occurrences of that pointer. As a result, the part of the molecule between repeating pointers is excised from the molecule in the form of circular molecule, while the parts from both sides of the excised molecule splice together;
- *Hairpin, Inverted Repeat* (hi) is applied on a pair of pointers, where one is an inverted repeat of the other one. The molecule is folded as a hairpin to facilitate recombination on those pointers. As a result of the operation, the part of the molecule flanked by the repeating pointers is inverted;
- *Double Loop, Alternating Direct Repeat* (dlad) is applicable to the overlapping direct repetitions of pointers, i.e., if we have a molecule of the form $\cdots p \cdots q \cdots p \cdots q \cdots$. The molecule folds to form a double loop so that a double recombination on p and q is facilitated. As a result, the parts of the molecule between the first and the second occurrences of p and q are exchanged.

A sequence of nucleotides is considered to act as a pointer only when placed at the border of an MDS and an IES. Note that after applying an operation on a certain pointer, that pointer remains as a sequence of nucleotides in the molecule, but ceases to participate in other operations, because it does not reside anymore on the border between an IES and an MDS.

We represent each molecule through its sequence of MDSs. In turn, we represent each MDS through its incoming and outgoing pointers, as well as through the sequence of pointers incorporated in the MDS as a result of applying previous operations. To formalize this definition, let $\Sigma_P = \{p_1, p_2, \cdots, p_n\}$ be the set of pointers. Then we represent an MDS by a triple $M = (p, u, q)$ where $p, q \in \Sigma_P$ are called *incoming* and *outgoing* pointers, respectively, and $u \in \Sigma_P^*$ is the *content*. We say that the length of M is $|M| = |puq|$. Let us denote by Σ_M the set $\{(p, u, q) \mid p, q \in \Sigma_P,\ u \in \Sigma_P^*\}$ of all MDSs. The complement of an MDS (p, u, q) is $(\overline{q}, \overline{u}, \overline{p})$ and $\overline{\Sigma_M} = \{\overline{M} \mid M \in \Sigma_M\}$. Finally, we call *descriptors* the strings from the set $S = (\Sigma_P \cup \overline{\Sigma_P} \cup \Sigma_M \cup \overline{\Sigma_M})^*$.

It is important to note that we consider in this paper descriptors in which pointers may have an arbitrary number of occurrences. Although in any successful assembly only two such occurrences are actually used, this multiplicity is the foundation of our ciliate-based search algorithm for a solution to HPP: choosing non-deterministically various occurrences of a given pointer in the assembly yield the detection of various paths in the given graph.

We formalize the ld, hi, and dlad operations as rewriting rules on descriptors as shown bellow. Note that all rewriting rules are non-deterministic: in general, for a given input, a rule may be applied in several ways, leading to different results. We assume here a non-deterministic computing paradigm: a descriptor may be assembled successfully if there exists a sequence of rules leading to its assembly, see bellow for formal details.

1 $\psi_1(q, u, p)\psi_2(p, v, r)\psi_3 \Rightarrow^{\mathsf{ld}_p} \psi_1(q, upv, r)\psi_3$;

2.1 $\psi_1(p, u, q)\psi_2(\overline{p}, \overline{v}, \overline{r})\psi_3 \Rightarrow^{\mathsf{hi}_p} \psi_1 p\overline{\psi_2}(\overline{q}, \overline{u}\ \overline{p}\ \overline{v}, \overline{r})\psi_3$;

2.2 $\psi_1(q, u, p)\psi_2(\overline{r}, \overline{v}, \overline{p})\psi_3 \Rightarrow^{\mathsf{hi}_p} \psi_1(q, upv, r)\overline{\psi_2}\ \overline{p}\psi_3$;

3.1 $\psi_1(p, u_1, r_1)\psi_2(q, u_2, r_2)\psi_3(r_3, u_3, p)\psi_4(r_4, u_4, q)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1 p\psi_4(r_4, u_4 q u_2, r_2)\psi_3(r_3, u_3 p u_1, r_1)\psi_2 q\psi_5$;

3.2 $\psi_1(p, u_1, r_1)\psi_2(r_2, u_2, q)\psi_3(r_3, u_3, p)\psi_4(q, u_4, r_4)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1 p\psi_4 q\psi_3(r_3, u_3 p u_1, r_1)\psi_2(r_2, u_2 q u_4, r_4)\psi_5$;

3.3 $\psi_1(r_1, u_1, p)\psi_2(q, u_2, r_2)\psi_3(p, u_3, r_3)\psi_4(r_4, u_4, q)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1(r_1, u_1 p u_3, r_3)\psi_4(r_4, u_4 q u_2, r_2)\psi_3 p\psi_2 q\psi_5$;

3.4 $\psi_1(r_1, u_1, p)\psi_2(r_2, u_2, q)\psi_3(p, u_3, r_3)\psi_4(q, u_4, r_4)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1(r_1, u_1 p u_3, r_3)\psi_4 q\psi_3 p\psi_2(r_2, u_2 q u_4, r_4)\psi_5$;

3.1′ $\psi_1(p, u_1, r_1)\psi_2(q, u_2, p)\psi_4(r_4, u_4, q)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1 p\psi_4(r_4, u_4 q u_2 p u_1, r_1)\psi_2 q\psi_5$;

3.2′ $\psi_1(p, u_1, q)\psi_3(r_3, u_3, p)\psi_4(q, u_4, r_4)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1 p\psi_4 q\psi_3(r_3, u_3 p u_1 q u_4, r_4)\psi_5$;

3.3′ $\psi_1(r_1, u_1, p)\psi_2(q, u_2, r_2)\psi_3(p, u_3, q)\psi_5 \Rightarrow^{\mathsf{dlad}_{p,q}}$
$\psi_1(r_1, u_1 p u_3 q u_2, r_2)\psi_3 p\psi_2 q\psi_5$;

where ψ_i are descriptors, q, r, r_i are pointers, u, v, u_i are sequences of pointers.

Consider also the operation cut on descriptors defined in the following way: $\mathsf{cut}(\psi_1(b,u,e)\psi_2) = (b,u,e)$. We call an MDS (b,u,e) a *successful assembly* of the descriptor ψ if $(b,u,e) = \mathsf{cut}(\phi_1(\phi_2(\cdots\phi_k(\psi)\cdots)))$, with $\phi_1,\cdots,\phi_k$ being some ld, hi, or dlad rules.

For a descriptor ψ we denote by $\mathcal{L}_{\mathsf{ld}}(\psi)$ ($\mathcal{L}_{\mathsf{hi}}(\psi)$, $\mathcal{L}_{\mathsf{dlad}}(\psi)$) the set of all MDSs assembled successfully from ψ using only ld-rules (hi, dlad, respectively).

For more details about the formalization of the gene structure and the intramolecular operations we refer to [4], [5], [6], [7], [9], [10], [11], [20], [21], as well as to the monograph [3].

4 Computing through gene assembly

Our principle of computing through gene assembly is the following: given a (mathematical) problem, we encode its input into a descriptor as defined in the previous section in such a way that the problem has a solution if and only if the associated descriptor has a successful assembly with certain properties. Moreover, the result of the assembly encodes the solution to the problem.

As our computational problem of choice we consider in this paper the hamiltonian path problem (HPP): given a directed graph $G = (V,E)$ and two nodes $p,q \in V$ one needs to decide whether or not G has a hamiltonian path from p to q. To solve the problem through the gene assembly by ld only, hi only or dlad only, we encode the set of edges of graph $G' = ext(G,p,q)$ into certain descriptors ψ_G^{ld}, ψ_G^{hi} or ψ_G^{dlad} respectively. Also, we encode any path v in G' through an MDS M_v using a construction described bellow. We prove then in each case that the graph G contains a hamiltonian path u if and only if descriptors ψ_G^{ld}, ψ_G^{hi} and ψ_G^{dlad} can be successfully assembled to descriptors containing MDS M_{bue}.

Let $G = (V,E)$ be a directed graph and $f = (p',q') \in E$ an edge of G. We then associate to f the MDS $M_f = (p',\Lambda,q')$. In general, for a set of edges $\{(q_1,q_2),(q_2,q_3),\cdots(q_{k-1},q_k)\}$ of G, we encode the path $u = q_1q_2\cdots q_{k-1}q_k$ of G through the MDS $M_u = (q_1, q_2\cdots q_{k-1}, q_k)$.

We say that *a node r appears in an MDS* (p,u,q) if symbol r appears in the string puq.

5 Computing using ld only

In this section we consider a (theoretical) solution to the hamiltonian path problem through gene assembly with ld only to be used throughout the assembly.

Let $G = (V,E)$ be a directed graph with $V = \{p_1,p_2,\cdots,p_n\}$, $n > 0$, and consider the hamiltonian path problem with p_1 as the starting node and p_n as the ending node. We reduce it to the same problem for graph $G' = ext(G,p_1,p_n)$, starting node b and ending node e.

We say that *a descriptor ψ_G is associated to G* if it is of the form

$$\psi_G^{\mathsf{ld}} = (b,\Lambda,p_1)\alpha_G^{n-1}(p_n,\Lambda,e), \text{ where } \alpha_G = \prod_{(p,q)\in E}(p,\Lambda,q) \text{ is a descriptor}$$

encoding all edges of G. Note that in general there are many descriptors associated to G, depending on the order in which the edges are encoded in α_G. As far as our solution to HPP is concerned, we may freely choose any of them.

Example 2. Consider the graph G_1 from Example 1. Then $\psi^{\mathsf{ld}}_{G_1} = (b, \Lambda, 3)\ (2, \Lambda, 1)\ (3, \Lambda, 1)(3, \Lambda, 2)\ (2, \Lambda, 1)(3, \Lambda, 1)(3, \Lambda, 2)(1, \Lambda, e)$ is associated to G_1, and $\mathcal{L}_{\mathsf{ld}}(\psi^{\mathsf{ld}}_{G_1}) = \{(b, 321, e), (b, 31, e)\}$.

Using the encoding presented above, we can now prove that gene assembly solves the HPP problem. Let G be a directed graph and consider the hamiltonian path problem from b to e for $G' = ext(G, p_1, p_n)$. Let ψ^{ld}_G be an arbitrary descriptor associated to G. Then the following results hold.

Lemma 1. *Any successfully assembled MDS $M \in \mathcal{L}_{\mathsf{ld}}(\psi^{\mathsf{ld}}_G)$ is associated to a path from b to e in G'.*

Lemma 2. *For every acyclic path u from b to e in G', $M_u \in \mathcal{L}_{\mathsf{ld}}(\psi^{\mathsf{ld}}_G)$.*

Theorem 1. *The hamiltonian path problem for graph $G = (V, E)$ and nodes p_1, p_n has an affirmative answer if and only if there exists $M \in \mathcal{L}_{\mathsf{ld}}(\psi^{\mathsf{ld}}_G)$ where all nodes appear and $|M| = |V| + 2$. In this case, M is an encoding of a hamiltonian path of $G' = ext(G, p_1, p_n)$ from b to e.*

6 Computing using hi only

In this section we consider a (theoretical) solution to the HPP problem through the gene assembly by hi operation only.

Consider a directed graph $G = (V, E)$ with $V = \{p_1, \cdots, p_n\}$, $n > 0$ and the hamiltonian path problem with p_1 as the starting node and p_n as the ending node. We solve an equivalent HPP for $G' = ext(G, p_1, p_n)$ from b to e instead.

We say that *a descriptor* ψ^{hi}_G is associated to G if it is of the form

$$\psi^{\mathsf{hi}}_G = (b, \Lambda, p_1) \prod_{(p,q) \in E \cup \{(p_n, e)\}} g_{p,q}, \text{ where } g_{p,q} = (x, \Lambda, y)(p, \Lambda, q)(\overline{z}, \Lambda, \overline{y})$$

is a descriptor encoding an edge (p, q). The order of the descriptors $g_{p,q}$ in ψ^{hi}_G is not important.

Example 3. Consider the graph G_1 from Example 1. Then $\psi^{\mathsf{hi}}_{G_1} = (b, \Lambda, 3)\ (x, \Lambda, y)\ (2, \Lambda, 1)(\overline{z}, \Lambda, \overline{y})\ (x, \Lambda, y)(3, \Lambda, 1)(\overline{z}, \Lambda, \overline{y})(x, \Lambda, y)(3, \Lambda, 2)(\overline{z}, \Lambda, \overline{y})(x, \Lambda, y)(1, \Lambda, e)\ (\overline{z}, \Lambda, \overline{y})$ is associated to G_1. The corresponding successful assemblies are $\mathcal{L}_{\mathsf{hi}}(\psi^{\mathsf{hi}}_{G_1}) = \{(b, 321, e), (b, 31, e)\}$.

Using the encoding presented above, we can now prove that hi-operations solve the HPP problem. Consider a directed graph G and the hamiltonian path problem from b to e for $G' = ext(G, p_1, p_n)$. Then the following results hold.

Lemma 3. *Any successfully assembled MDS $M \in \mathcal{L}_{\mathsf{hi}}(\psi_G^{\mathsf{hi}})$ is associated to a path from b to e in G'.*

We omit the proof, since its idea is similar to that from Lemma 1.

Lemma 4. *For every path u from b to e in G' without repeating edges, there exists an MDS $M_u \in \mathcal{L}_{\mathsf{hi}}(\psi_G^{\mathsf{hi}})$.*

Theorem 2. *The hamiltonian path problem for graph $G = (V, E)$ and nodes p_1, p_n has an affirmative answer if and only if there exists MDS $M \in \mathcal{L}_{\mathsf{hi}}(\psi_G^{\mathsf{hi}})$ where all nodes appear and $|M| = |V| + 2$. In this case, M is an encoding of a hamiltonian path of $G' = ext(G, p_1, p_n)$ from b to e.*

Consider the bipartite transformation $bi(G)$ applied to graph G. In this case, for any node p the edge (p, p') is encoded only once, so only acyclic paths are assembled. Therefore, the following corollary holds.

Corollary 1. *The following statements are equivalent: (a) The hamiltonian path problem for graph $G = (V, E)$ and nodes p_1, p_n has an affirmative answer; (b) there exists $M \in \mathcal{L}_{\mathsf{hi}}(\psi_{bi(G)}^{\mathsf{hi}})$ where all nodes appear (c) there exists $M \in \mathcal{L}_{\mathsf{hi}}(\psi_{bi(G)}^{\mathsf{hi}})$ with $|M| = 2|V| + 2$. Moreover, the MDS M from (b) and (c) is an encoding of a hamiltonian path of $ext(bi(G), p_1, p'_n)$ from b to e.*

7 Computing using dlad only

We now consider a (theoretical) solution to the HPP problem through the gene assembly using only dlad operation.

Consider a directed graph $G = (V, E)$ with $V = \{p_1, \cdots, p_n\}$, $n > 0$ and the hamiltonian path problem in $G' = ext(G, p_1, p_n)$ from b to e.

We say that *a descriptor ψ_G^{dlad} is associated to G* if it is of the form

$$\psi_G^{\mathsf{dlad}} = g_{b,p_1} \left(\prod_{(p,q) \in E \cup \{(p_n,e)\}} g_{p,q} \right) (r, x)^{|E|}, \text{ where } g_{p,q} = (p, q)(x, y)$$

is a descriptor encoding an edge (p, q); we may choose any order of encoding edges (with the exception g_{b,p_1} must be the first) in ψ_G^{dlad}.

Example 4. Consider the graph G_1 from Example 1. Then $\psi_{G_1}^{\mathsf{dlad}} = (b, \Lambda, 3)$ $(x, \Lambda, y)(2, \Lambda, 1)(x, \Lambda, y)$ $(3, \Lambda, 1)(x, \Lambda, y)(3, \Lambda, 2)(x, \Lambda, y)(1, \Lambda, e)(x, \Lambda, y)(r, \Lambda, x)$ $(r, \Lambda, x)(r, \Lambda, x)(r, \Lambda, x)(r, \Lambda, x)$ is associated to G_1. The successful assemblies are $\mathcal{L}_{\mathsf{dlad}}(\psi_{G_1}^{\mathsf{dlad}}) = \{(b, 321, e), (b, 31, e)\}$.

Equipped with this encoding, we now prove that dlad solves the HPP problem. Consider a directed graph G and the hamiltonian path problem in G' from b to e. The following results hold.

Lemma 5. *Any successfully assembled MDS $M \in \mathcal{L}_{\mathsf{dlad}}(\psi_G^{\mathsf{dlad}})$ is associated to a path from b to e.*

We omit the proof, since its idea is again similar to that in Lemma 1.

Lemma 6. *For any path u without repeating edges, exists MDS $M_u \in \mathcal{L}_{\mathsf{dlad}}(\psi_G^{\mathsf{dlad}})$.*

Theorem 3. *The hamiltonian path problem for graph $G = (V, E)$ and nodes p_1, p_n has an affirmative answer if and only if there exists MDS $M \in \mathcal{L}_{\mathsf{dlad}}(\psi_G^{\mathsf{dlad}})$ where all nodes appear and $|M| = |V| + 2$. In this case, M is an encoding of a hamiltonian path of $G' = ext(G, p_1, p_n)$ from b to e.*

Consider the bipartite transformation $bi(G)$ applied to graph G. In this case, for any node p the edge (p, p') is encoded only once, so only acyclic paths are assembled. Therefore, the following corollary holds.

Corollary 2. *The following statements are equivalent: (a) The hamiltonian path problem for graph $G = (V, E)$ and nodes p_1, p_n has an affirmative answer; (b) there exists $M \in \mathcal{L}_{\mathsf{dlad}}(\psi_{bi(G)}^{\mathsf{dlad}})$ where all nodes appear; (c) there exists $M \in \mathcal{L}_{\mathsf{dlad}}(\psi_{bi(G)}^{\mathsf{dlad}})$ with $|M| = 2|V| + 2$. Moreover, the MDS M from (b) and (c) is an encoding of a hamiltonian path of $ext(bi(G), p_1, p'_n)$ from b to e.*

8 Discussion

It has been observed many times in the literature that gene assembly in ciliates has a definite computational flavor. Two mathematical models were proposed to model gene assembly as a computational process transforming one structure into another one. Moreover, it has been shown that both models are Turing universal: assuming that a Turing machine may be encoded in the form of an artificial gene of high enough length and present in a high enough number of copies, then the Turing machine may be simulated through gene assembly, see [13–15]. The approach that we take in this paper is different. Given a mathematical problem such as HPP, we ask the question how to encode the problem into a gene pattern such that solving the problem is equivalent with assembling the gene. Using each of the three operations ld, hi, and dlad, we show that the construction is indeed possible, at least theoretically. It is important to underline here the connection with the computational principle in the celebrated experiment of Adleman [1]. While in [1], one encodes the given graph into a set of molecules that recombine among themselves to yield in principle the encodings of all paths through the graph, we encode our graph into a set of sequences that are placed in an arbitrary order on a chromosome-like molecule. This molecule may be assembled in many possible ways; in fact, the encodings of all paths of a certain length may be assembled in this way. Although a micronuclear gene is presented in several copies in a ciliate, it remains to be tested experimentally if a ciliate would assemble two or more identical copies of our artificial gene into several

different forms. Answering this question would clarify the scale of a prototype experiment to test our approach, in terms of the number of ciliates required.

Some recent results of [2] and [22] suggest that RNA-template could be used to control and direct gene assembly. Based on this, one may attempt to implement our ciliate-based solutions to HPP. For example, one may inject templates to indicate all possibilities in which two MDSs may recombine. The amount of such templates would thus be at most quadratic in the number of MDSs used by our encoding. Clearly, this can only be validated through laboratory experiments.

Acknowledgments

I. Petre gratefully acknowledges support by Academy of Finland, project 108421, A. Alhazov and V. Rogojin gratefully acknowledge support by Academy of Finland, project 203667.

References

1. Adleman, L.M., Molecular computation of solutions to combinatorial problems. *Science* **226** (1994), 1021–1024.
2. Angeleska, A., Jonoska, N., Saito, M., and Landweber, L.F., RNA-Template Guided DNA Assembly. In: M. Garzon and H. Yan (eds.) Preliminary Proceedings on DNA13 meeting, University of Memphis, Memphis (2007), 364.
3. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., *Computation in Living Cells: Gene Assembly in Ciliates*, Springer (2003).
4. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., Formal systems for gene assembly in ciliates. *Theoret. Comput. Sci.* **292** (2003) 199–219.
5. Ehrenfeucht, A., Harju, T., Petre, I., and Rozenberg, G., Characterizing the micronuclear gene patterns in ciliates. *Theory of Comput. Syst.* **35** (2002) 501–519.
6. Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., String and graph reduction systems for gene assembly in ciliates. *Math. Structures Comput. Sci.* **12** (2001) 113–134.
7. Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., Circularity and other invariants of gene assembly in ciliates. In: M. Ito, Gh. Păun and S. Yu (eds.) *Words, semigroups, and transductions*, World Scientific, Singapore, (2001), 81–97.
8. Ehrenfeucht, A., Prescott, D. M., and Rozenberg, G., Computational aspects of gene (un)scrambling in ciliates. In: L. F. Landweber, E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin, Heidelberg, New York (2001), 216–256.
9. Harju, T., Petre, I., Li, C. and Rozenberg, G., Parallelism in gene assembly. In: C. Ferretti, G. Mauri, C. Zandron (Eds.) *Proceedings of the 10th International Meeting on DNA-based computers, DNA10*, Springer, *Lecture Notes in Computer Science* **3384**, (2005), 138–148.
10. Harju, T., Petre, I., and Rozenberg, G., Gene assembly in ciliates: molecular operations. In: Gh. Păun, G. Rozenberg, A.Salomaa (Eds.) *Current Trends in Theoretical Computer Science*, (2004).
11. Harju, T., Petre, I., and Rozenberg, G., Gene assembly in ciliates: formal frameworks. In: Gh. Păun, G. Rozenberg, A.Salomaa (Eds.) *Current Trends in Theoretical Computer Science*, (2004).

12. Kari, L., and Landweber, L. F., Computational power of gene rearrangement. In: E. Winfree and D. K. Gifford (eds.) *Proceedings of DNA Bases Computers, V* American Mathematical Society (1999) 207–216.
13. Landweber, L. F., and Kari, L., The evolution of cellular computing: Nature's solution to a computational problem. In: *Proceedings of the 4th DIMACS Meeting on DNA-Based Computers*, Philadelphia, PA (1998), 3–15.
14. Landweber, L. F., and Kari, L., Universal molecular computation in ciliates. In: L. F. Landweber and E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York (2002).
15. Onolt-Ishdorj, T., Petre, I., and Rogojin, V., Computational Power of Intramolecular Gene Assembly, *Computability in Europe 2007*, submitted.
16. Papadimitriou, C.H., *Computational Complexity*. Addison-Wesley, (1994).
17. Prescott, D. M., The DNA of ciliated protozoa. *Microbiol. Rev.* **58**(2) (1994) 233–267.
18. Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260.
19. Petre, I., Invariants of gene assembly in stichotrichous ciliates. *IT*, Oldenbourg Wissenschftsverlag, **3** (2006), 161–167.
20. Prescott, D. M., and Rozenberg, G., How ciliates manipulate their own DNA – A splendid example of natural computing. *Natural Computing* **1** (2002) 165–183.
21. Prescott, D. M., and Rozenberg, G., Encrypted genes and their reassembly in ciliates. In: M. Amos (ed.) *Cellular Computing*, Oxford University Press, Oxford (2003).
22. Vijayan, V., Nowacki, M., Zhou, Y., Doak, T., and Landweber, L., Programming a Ciliate Computer: Template-Guided In Vivo DNA Rearrangements in Oxytricha. In: M. Garzon and H. Yan (eds.) Preliminary Proceedings on DNA13 meeting, University of Memphis, Memphis (2007), 172.