Complexity Measures for Gene Assembly

Tero Harju[1], Chang Li[2], Ion Petre[2,3], and Grzegorz Rozenberg[4,5]

[1] Department of Mathematics, University of Turku, Turku Center for Computer Science, FIN-20014 Turku, Finland `email:harju@utu.fi`
[2] Department of Computer Science, Åbo Akademi University, Turku Center for Computer Science, FIN-20520 Turku, Finland `email:lchang@abo.fi`
[3] Academy of Finland `email:ipetre@abo.fi`
[4] Leiden Institute for Advanced Computer Science, Leiden University, 2333 CA Leiden, the Netherlands `email:rozenber@liacs.nl`
[5] Department of Computer Science, University of Colorado, Boulder, Co 80309-0347, USA

Abstract. The process of gene assembly in ciliates is a fascinating example of programmed DNA manipulations in living cells. We describe in this paper four measures of complexity for this process, based on: (a) the types of operations used in the assembly, (b) the number of operations used in the assembly, (c) the length of the molecular folds involved, and (d) the length of the shortest possible parallel assembly for that gene.

"One of the oldest forms of life on Earth has been revealed as a natural born computer programmer."
BBC, September 10, 2001.

1 Introduction

Ciliates are very old eukaryotic unicellular organisms that have developed an unusual way of organizing their genome. Each cell has two types of functionally different nuclei - the *macronucleus* is the somatic nucleus, while the *micronucleus* is the germline nucleus. Depending on the species each type of nuclei may be present in many copies in each cell.

The macronuclear genes are very short molecules, ranging in the S.nova organisms between 200bp and 3700bp, with an average of 2200 bp in length, see [22], [19], [3], [4]. Incidentally, these are the shortest DNA molecules known in Nature, see [20]! The micronuclear genome however is organized on very long chromosomes (about 120 chromosomes, each with about 10^7 bp in S.nova, see [19]), with coding sequences occupying as little as 2 - 5% of the genome, see, e.g., [3]. During the process of sexual reproduction, ciliates destroy the old macronuclei and transform a micronucleus into a new macronucleus. Ciliates thus have to identify precisely the genetic material and splice it out from the chromosomes. The complexity of the process is given by the fundamentally different organization of the micronuclear and the macronuclear genomes. The structure is particularly complex in a family of ciliates called *Stichotrichs* – we concentrate in this paper on this family.

The macronuclear gene is a contiguous DNA sequence, with genes generally placed on their own very short DNA molecules. The same gene in the micronucleus is broken into pieces called *MDSs (macronuclear destined sequences)* that are separated by noncoding blocks called *IESs (internally eliminated sequences)*. Moreover, the order of MDSs is shuffled, with some of the MDSs being inverted. Here is where the challenge of gene assembly lies: ciliates have to identify correctly more than 100 000 MDSs in their genome, see [20], assemble them together in the orthodox order, and eliminate all IESs. We refer to [12], [19], [23] for more details on ciliates and gene assembly.

A hint on how ciliates achieve gene assembly is given by the structure of MDSs. It turns out that ciliates organize their genomic data as *linked lists* in the style used in computer science, see [19]. A short sequence in the end of each MDS is repeated identically in the beginning of the MDS that should follow it in the orthodox order, thus serving as a computer science-like pointer. Moreover, the first MDS starts with a special beginning marker, while the last MDS ends with a special ending marker. It is currently believed that ciliates splice together their MDSs on the common pointers to assemble the gene. There are two main models for gene assembly, see [16], [17] and [8], [21], that both agree on this generic mechanism.

We concentrate in this paper on the *intramolecular* model of [8], [21]. The model consists of three molecular operations: ld, hi, and dlad. In each of these operations, the molecule folds on itself so that two or more pointers get aligned and through recombination two or more MDSs get combined into a bigger composite MDS. The process continues until all MDSs have been assembled.

First operation: ld . In the operation *(loop, direct repeat)-excision*, or ld for short, a pair of pointers flanking an IES guides the excision of the IES as a circular molecule, as illustrated in Fig. 1. The DNA molecule folds on itself so that the two pointers can get aligned, after which the IES is excised through recombination. As a result, two MDSs get joined and form a bigger coding block. It is crucial to note that the excised molecule does not contain any coding blocks and so, it is not required to participate anymore in the gene assembly process.

Second operation: hi . The operation *(hairpin, inverted repeat)-excision/reinsertion*, or hi for short, is applicable to a molecule containing a pair of pointers where one pointer is the inversion of the other. This is illustrated in Fig. 2. The molecule folds on itself forming a hairpin so that the two copies of the pointer can get aligned with the same polarity, thus facilitating the recombination. Through recombination, the sequence between the two occurrences of the pointer is inverted. One may also note that as a result of applying hi, two MDSs are joined together into a bigger coding block, while two IESs are joined together into a bigger noncoding block.

Third operation: dlad . The operation *(double loop,alternating direct repeat)-excision/reinsertion*, or dlad for short applies to a DNA molecule containing two pairs of pointers where the segments encompassed by the pairs of pointers

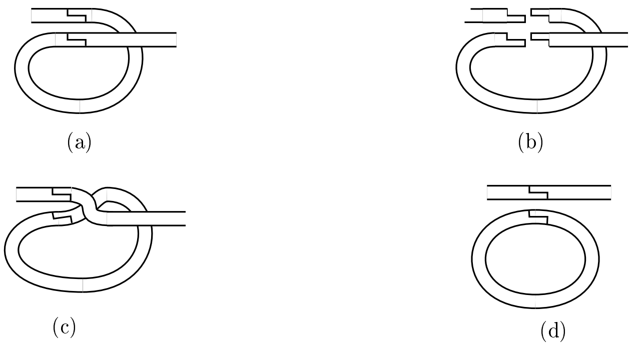


Fig. 1. Illustration of the ld-rule.

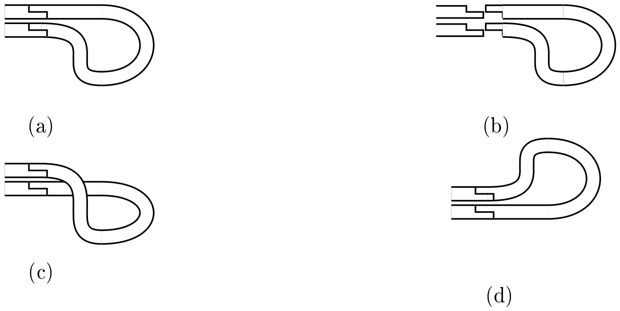


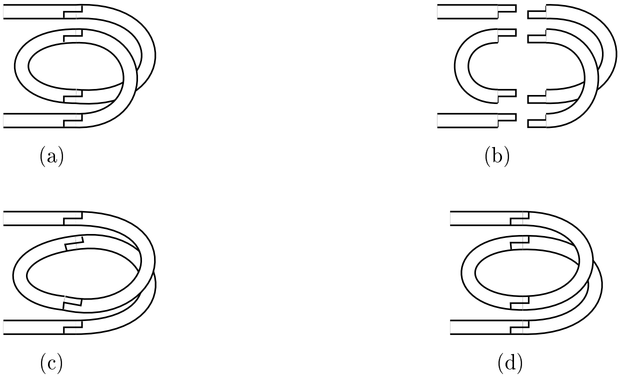
Fig. 2. Illustration of the hi-rule.

overlap with each other. This is illustrated in Fig. 3. The molecule folds into two loops so that the two copies of the first pointer align with each other in one loop, and the two copies of the second pointer align with each other in the other loop. Thus, the molecule is in position for two recombinations. As a result of this double recombination, two sequences are translocated; several MDSs are joined together into bigger coding sequences, see [6] for details.

2 Definitions

We give in this section some basic notions concerning permutations, strings, and graphs.

Signed strings For a finite alphabet $\Sigma = \{a_1, \ldots, a_n\}$, we denote by Σ^* the free monoid generated by Σ and call any element of Σ^* a *string*. Let $\overline{\Sigma} = \{\overline{a}_1, \ldots, \overline{a}_n\}$, where $\Sigma \cap \overline{\Sigma} = \emptyset$. For $p, q \in \Sigma \cup \overline{\Sigma}$, we say that p, q have the same *signature* if either $p, q \in A$, or $p, q \in \overline{A}$ and we say that they have *different signatures* otherwise. For $p \in \Sigma$, we say that p is an *unsigned letter*, while for $p \in \overline{\Sigma}$, we say that p is a *signed letter*.


(a)

(b)

(c)

(d)

Fig. 3. Illustration of the dlad-rule.

We denote $\Sigma^{\maltese} = (\Sigma \cup \overline{\Sigma})^*$. For any $u \in \Sigma^{\maltese}$, $u = x_1 \dots x_k$, with $x_i \in \Sigma \cup \overline{\Sigma}$, for all $1 \leq i \leq k$, we denote $\|u\| = \|x_1\| \dots \|x_k\|$, where $\|a\| = \|\overline{a}\| = a$, for all $a \in \Sigma$. We also denote $\overline{u} = \overline{x}_k \dots \overline{x}_1$, where $\overline{\overline{a}} = a$, for all $a \in \Sigma$.

We say that $u \in \Sigma^{\maltese}$ is a *signed double occurrence string* if for any $p \in \Sigma$, u has either 0, or 2 occurrences from the set $\{p, \overline{p}\}$. In case u has two occurrences from the set $\{p, \overline{p}\}$, we say that p is a *positive letter* in u if the two occurrences have different signatures, and we say that is a *negative letter* in u if they have the same signature.

Let u be a signed double occurrence string. We say that letters p and q, $p \neq q$, *overlap* in u if $u = u_1 p u_2 q u_3 p u_4 q u_5$, for some $u_i \in \Sigma^{\maltese}$, $1 \leq i \leq 5$.

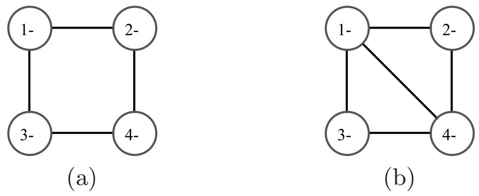
Signed permutations A *permutation* π over alphabet Σ is a bijection $\pi : \Sigma \to \Sigma$. Fixing the order relation $(a_1, a_2, \dots, a_m)$ over Σ, we often denote π as the string $\pi(a_1) \dots \pi(a_m) \in \Sigma^*$. A *signed permutation* over Σ is a string $\psi \in \Sigma^{\maltese}$, where $\|\psi\|$ is a permutation over Σ.

Signed graphs A *signed graph* is a triple $G = (V, E, \phi)$, with V a finite set of *vertices*, $E \subseteq V \times V$ the set of (undirected) *edges*, with the property that $(x, y) \in E$ if and only if $(y, x) \in E$, and $\phi : V \to \{+, -\}$ the *signature function*. We say that vertex $p \in V$ is positive if $\phi(p) = +$ and it is *negative* otherwise. For all $p \in V$, we denote by $N_G(p)$ the neighborhood of p in G.

For a signed graph $G = (V, E, \phi)$ and $p \in V$ we denote by $G \setminus \{p\}$ the graph induced by the set of vertices $V \setminus \{p\}$. We also denote by $\mathsf{loc}_p(G)$ the *local complementation* of G at p: $\mathsf{loc}_p(G) = (V, E', \phi')$, where $(x, y) \in E'$ if and only if $(x, y) \notin E$, for all $x, y \in N_G(p)$ and $(x, y) \in E'$ if and only if $(x, y) \in E$ otherwise. Also, $\phi'(x) = +$ if and only if $\phi(x) = -$, for all $x \in N_G(p)$ and $\phi'(x) = \phi(x)$, otherwise.

We denote by C_4 and D_4 the graphs illustrated in Fig. 4.

For any signed double occurrence string u over alphabet Σ, we may associate to u a signed graph $G_u = (V_u, E_u, \phi_u)$ in the following way: $V_u = \{p \in \Sigma \mid p \text{ or } \overline{p} \text{ occurs in } u\}$, $E_u = \{(p, q) \mid \ p \text{ and } q \text{ overlap in } u\}$, and $\phi_u(p) = +$ if and



(a)

(b)

Fig. 4. (a) The square C_4; (b) the diamond D_4.

only if p is a positive letter in u. The graph G_u is also called the *overlap graph* of u.

For $k \geq 2$ we will use throughout the paper the alphabets $\Sigma_k = \{1, \ldots, k\}$ and $\Delta_k = \{2, \ldots, k\}$.

3 Three models for gene assembly

The intramolecular model for gene assembly, [8], [21], has been formalized on several levels of abstraction. The structures of genes can be represented as: signed permutations, MDS descriptors, signed double occurrence strings, or signed overlap graphs. Consequently, the process of gene assembly can be formalized on strings, or reduction of graphs. As it turns out, all these levels of abstraction are equivalent as far as the modeling of gene assembly is concerned, see [6] for a detailed discussion on model forming. Nevertheless, different levels of abstraction prove more suitable (more elegant or technically simpler) than the others depending on the subject of the investigation.

In this paper we consider issues dealing with formalization of the gene assembly on the level of string permutation, signed double occurrence strings, and signed graphs. We present briefly these three abstraction levels and we refer to [6] for more details.

For any gene γ having k MDSs, $k \geq 1$, we may associate a signed permutation to γ in the following way: associate to the MDS M_i letter i, $1 \leq i \leq k$, and its inversion $\overline{M}_i$ the signed letter $\bar{i}$. Thus the signed permutation associated to the MDS sequence $M_3\overline{M}_1M_2$ is simply the signed permutation $3\bar{1}2$.

We may also associate a signed double occurrence string to any gene (more generally, to any sequence of MDSs), simply by writing its sequence of pointers. Thus, for a sequence of k MDSs, we associate to each MDS M_i, $2 \leq i \leq k-1$, the string consisting of its incoming and outgoing pointers: $i(i+1)$. To $\overline{M}_i$ we associate string $i(i+1)$. The first and the last MDS are special because they contain some markers, that will ignore in our string-based representation. Thus, to M_1 we associate string 2 and $\overline{M}_1$ string $\bar{2}$. Similarly, to M_k we associate string k and to $\overline{M}_k$ string $\bar{k}$, Consequently, to the MDS sequence $M_3\overline{M}_1M_2$ we associate string $3\bar{2}23$. Also, to the MDS sequence $M_2\bar{M}_4M_1M_3$ we associate string $23\bar{4}234$.

On a higher level of abstraction, we may associate a graph to a sequence of MDS in the following way. If U_γ is the string associated to gene γ, then G_γ is the signed overlap graph of U_γ, as defined in Section 2. Thus, the graph associated to the MDS sequence $M_3\overline{M}_1M_2$ consists of positive vertex 2, adjacent to negative vertex 3.

The molecular operation ld, hi, dlad are modeled as string rewriting rules as follows (without risk of confusion we use the notation ld, hi, dlad also for the string rules).

Let u be a signed double occurrence string over alphabet Δ_k.

1. For all $p \in \Delta_k \cup \overline{\Delta}_k$, ld_p is defined as follows:
$$\mathsf{ld}_p(uppv) = uv,$$
where $u, v \in \Delta_k^{\maltese}$.
2. For all $p \in \Delta_k \cup \overline{\Delta}_k$, hi_p is defined as follows:
$$\mathsf{hi}_p(upv\bar{p}w) = u\bar{v}w,$$
where $u, v \in \Delta_k^{\maltese}$.
3. For all $p \in \Delta_k \cup \overline{\Delta}_k$, $\mathsf{dlad}_{p,q}$ is defined as follows:
$$\mathsf{dlad}_{p,q}(u_1pu_2qu_3pu_4qu_5) = u_1u_4u_3u_2u_5,$$
where $u_i \in \Delta_k^{\maltese}$, for all $1 \le i \le 5$.

We say that a composition ϕ of ld, hi, and dlad operations is a *reduction strategy* for string u if $\phi(u) = \Lambda$.

Example 1. Let $u = 3\bar{5}26547367 2\bar{4}88$, then ld_8 is applicable to u: $\mathsf{ld}_8(u) = 3\bar{5}265473672\bar{4}$. Also, hi_4 and $\mathsf{dlad}_{5,6}$ are applicable to u: $\mathsf{hi}_4(u) = 3\bar{5}265\bar{2}\bar{7}\bar{6}\bar{3}788$, and $\mathsf{dlad}_{3,2}(u) = 676546\bar{5}\bar{4}88$.

The corresponding operations for signed graphs are defined as follows. Let $G = (V, E)$ be a signed graph.

1. For all $p \in V$, ld_p is applicable to G if and only if p is an isolated negative vertex in G. In this case, $\mathsf{ld}_p(G) = G \setminus \{p\}$.
2. For all $p \in V$, hi_p 4 is applicable to G if and only if p is an positive vertex in G. In this case, $\mathsf{hi}_p(G) = \mathsf{loc}_p(G) \setminus \{p\}$.
3. For all $p, q \in V$, $\mathsf{dlad}_{p,q}$ is applicable to G if and only if p and q are adjacent negative vertices in G. Then $\mathsf{dlad}_{p,q}(G) = (V \setminus \{p, q\}, E')$, where E' is obtained from E by complementing the edges that join vertices in $N_G(p)$ with vertices in $N_G(q)$. This means that $(x, y) \in (E' \setminus E) \cup (E \setminus E')$ if and only if
$$\begin{aligned} &x \in N_G(p) \setminus N_G(q) \text{ and } y \in N_G(q), \text{ or} \\ &x \in N_G(q) \cup N_G(q) \text{ and } y \in (N_G(p) \setminus N_G(q)) \cup (N_G(q) \setminus N_G(p)), \text{ or} \\ &x \in N_G(q) \setminus N_G(p) \text{ and } y \in N_G(p). \end{aligned}$$

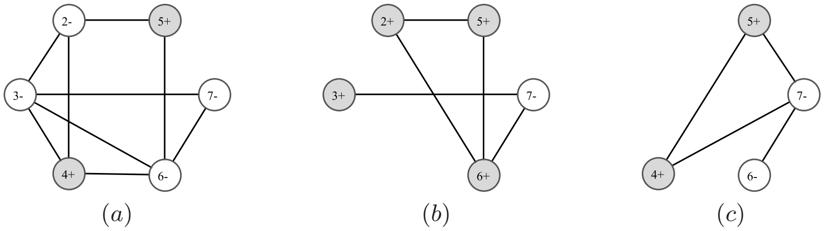


Fig. 5. (a) The graph G_u in Example 1; (b) $\mathsf{hi}_4(G_u)$; (c) $\mathsf{dlad}_{2,3}(G_u)$

We say that a composition ψ of ld, hi, and dlad operations is a *reduction strategy* for graph G if $\psi(G) = \emptyset$.

Example 2. The signed overlap graph associated to string u in Example 1 is illustrated in Fig. 5

Without risk of confusion, we denote both for the string rules and for the graph rules: $\mathsf{Ld} = \{\mathsf{ld}_p \mid p \geq 2\}$, $\mathsf{Hi} = \{\mathsf{hi}_p \mid p \geq 2\}$, $\mathsf{Dlad} = \{\mathsf{dlad}_{p,q} \mid p, q \geq 2, p \neq q\}$.

Note that the process of gene assembly, and all its formalizations, as sorting permutations, reducing strings, or reducing graphs, is non-deterministic. We illustrate this in the following example.

Example 3. Let $u = 562324573467$ be a double occurrence string. The u has at least two reduction strategies: $\phi_1 = \mathsf{ld}_7 \circ \mathsf{ld}_4 \circ \mathsf{dlad}_{5,6} \circ \mathsf{dlad}_{2,3}$ and $\phi_2 = \mathsf{ld}_5 \circ \mathsf{ld}_6 \circ \mathsf{dlad}_{2,3} \circ \mathsf{dlad}_{4,7}$. Indeed,

$$\phi_1(u) = (\mathsf{ld}_7 \circ \mathsf{ld}_4 \circ \mathsf{dlad}_{5,6})(56457467) = (\mathsf{ld}_7 \circ \mathsf{ld}_4)(7447) = \mathsf{ld}_7(77) = \Lambda,$$

$$\phi_2(u) = (\mathsf{ld}_5 \circ \mathsf{ld}_6 \circ \mathsf{dlad}_{2,3})(56232635) = (\mathsf{ld}_5 \circ \mathsf{ld}_6)(5665) = \mathsf{ld}_5(55) = \Lambda.$$

Note that the two strategies have the same number of ld operations, albeit applied on different pointers.

The following result, adapted from [6] provides an invariant for all sorting strategies of a given string.

Theorem 1 ([6]). *Let u be a signed double occurrence string and ϕ_1, ϕ_2 two reduction strategies for u. Then ϕ_1 and in ϕ_2 contain the same number of* ld *operations.*

4 First complexity measure: the minimal subset of operations sufficient for gene assembly

We introduce in this section our first measure of gene complexity in terms of the smallest set of (types of) operations that are capable to assemble the given gene.

Our formalism in this section will be that of signed double occurrence strings. Note that a similar presentation may also be done in terms of signed graphs, see[5].

The concept of (gene) complexity here is the following. For a given string x, consider assembly strategies φ for x, and take the set $S_\varphi \subseteq \{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$ of those types of operations that are used in φ. We say that S_φ is an *assembly set* for x.

Example 4. Note that a string may have several assembly sets. For instance, if $u = 2\bar{3}\bar{2}434$, then $\varphi_1(u) = \mathsf{dlad}_{3,4} \circ \mathsf{hi}_2$ is an assembly strategy for u: $\varphi_1 = \mathsf{dlad}_{3,4}(3434) = \Lambda$. Thus, $\{\mathsf{Hi}, \mathsf{Dlad}\}$ is an assembly set for u. However, $\{\mathsf{Hi}\}$ is also an assembly set for u. Indeed, $\varphi_2 = \mathsf{hi}_2 \circ \mathsf{hi}_4 \circ \mathsf{hi}_3$ is an assembly strategy for u: $\varphi_2(u) = (\mathsf{hi}_2 \circ \mathsf{hi}_4)(2\bar{4}24) = \mathsf{hi}_2(2\bar{2}) = \Lambda$.

We say that a set $S \subseteq \{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$ is a *minimal assembly set* for X, if for any $T \subseteq S$, where T is an assembly set for X, we have $T = S$.

As we will observe at the end of this section, a string X has a unique minimal assembly set. Anticipating this result, the following notion of complexity is well defined.

Definition 1. *The complexity $\mathcal{C}_1(X) \subseteq \{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$ of a signed double occurrence string X is the minimal assembly set of X.*

To prove the result announced above, we need to consider for every $S \subseteq \{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$, what are the strings with S as an assembly set. The first complete characterization was given in [5] in the case of realistic strings. The results were then extended to signed double occurrence strings in [1]; the characterizations in [1] are based in part on a notion of break point graphs. For simplicity, we only consider here the case of elementary strings and the approach in [5].

Theorem 2 ([5]). *Let u be an elementary string.*

(i) $\{\mathsf{Ld}\}$ *is an assembly set for u if and only if u has no overlap and no signed letters.*
(ii) $\{\mathsf{Ld}, \mathsf{Hi}\}$ *is an assembly set for u if and only if* $|u| \leq 2$ *or u contains at least one positive pointer.*
(iii) $\{\mathsf{Ld}, \mathsf{Dlad}\}$ *is an assembly set for u if and only if u has no signed letters.*
(iv) $\{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$ *is an assembly set for any signed double occurrence string.*

We omit in this paper the characterization of the strings with $\{\mathsf{Hi}\}$, $\{\mathsf{Dlad}\}$, or $\{\mathsf{Hi}, \mathsf{Dlad}\}$ as assembly sets. Somewhat technical characterization have been given for those case in[5].

Example 5. (a) The R_1 gene of S.nova is described by the MDS sequence $M_1 M_2 M_3 M_4 M_5 M_6$. Its associated string 2233445566 has $\{\mathsf{Ld}\}$ as an assembly set.
(b) String 2323 has $\{\mathsf{Dlad}\}$ as an assembly set.
(c) String $23\bar{2}3$ has $\{\mathsf{Hi}\}$ as an assembly set.

(d) String $2\bar{3}\bar{2}3$ has two assembly strategies: $\mathsf{ld}_3 \circ \mathsf{hi}_2$ and $\mathsf{ld}_2 \circ \mathsf{hi}_3$. Thus, it has $\{\mathsf{Ld}, \mathsf{Hi}\}$ as an assembly set.

(e) The α-TP gene of S.nova is described by the MDS sequence $M_1M_3M_5M_9 M_{11}M_2M_4M_6M_8M_{10}M_{12}M_{13}M_{14}$. Its associated string 2345691011122345 678910111213131414 has $\{\mathsf{Ld}, \mathsf{Dlad}\}$ as an assembly set.

(f) The actin I gene of S.nova is described by the MDS sequence $M_3M_4M_6M_5 M_7M_9\overline{M}_2M_1M_8$. Its associated string $3445675678 9\bar{3}\bar{2}289$ has $\{\mathsf{Ld}, \mathsf{Hi}, \mathsf{Dlad}\}$ as an assembly set.

We can now prove the following result.

Theorem 3. *Let $u \neq \Lambda$ be an elementary string and S_1, S_2 two minimal assembly sets for u. Then $S_1 = S_2$.*

Proof. Assume that there is an elementary string $u \neq \Lambda$ with two different minimal assembly sets S_1, S_2. Clearly, $S_1 \not\subseteq S_2$ and $S_2 \not\subseteq S_1$. We then have the following cases:

(i) $S_1 = \{\mathsf{Ld}\}$, $S_2 = \{\mathsf{Hi}\}$;
(ii) $S_1 = \{\mathsf{Ld}\}$, $S_2 = \{\mathsf{Dlad}\}$;
(iii) $S_1 = \{\mathsf{Ld}\}$, $S_2 = \{\mathsf{Hi}, \mathsf{Dlad}\}$;
(iv) $S_1 = \{\mathsf{Ld}\,\mathsf{Hi}\}$, $S_2 = \{\mathsf{Dlad}\}$;
(v) $S_1 = \{\mathsf{Ld}, \mathsf{Hi}\}$, $S_2 = \{\mathsf{Ld}, \mathsf{Dlad}\}$;
(vi) $S_1 = \{\mathsf{Ld}, \mathsf{Hi}\}$, $S_2 = \{\mathsf{Hi}, \mathsf{Dlad}\}$;
(vii) $S_1 = \{\mathsf{Ld}, \mathsf{Dlad}\}$, $S_2 = \{\mathsf{Hi}\}$;
(viii) $S_1 = \{\mathsf{Ld}, \mathsf{Dlad}\}$, $S_2 = \{\mathsf{Hi.Dlad}\}$;
(ix) $S_1 = \{\mathsf{Hi}\}$, $S_2 = \{\mathsf{Dlad}\}$.

In cases (i) -(iv) we obtain that u has two reduction strategy: one containing at least one Ld- operations, another containing none. This is a contradiction by Theorem 1. Using a similar argument, in case (vi)-(viii) it follows that S_1 is not a minimal assembly set for u, again a contradiction.

Finally, in case (v) and (ix) we obtain that $u = \Lambda$. Indeed, since $S_1 \supseteq \{\mathsf{Hi}\}$, u should have at least one signed letter. On the other hand, since $S_2 \not\supseteq \{\mathsf{Hi}\}$, u can not have any signed letter.

Corollary 1. *The complexity measure $\mathcal{C}_1$ is well defined.*

5 Second complexity measure: weights associated to the assembly operations

The concept of our second measure of complexity is straightforward: a gene is more “complex” than another if it requires more “effort” to be assembled. The simplest way to measure the “effort” required to assemble a given gene is through counting the number of operations required in the assembly.

Definition 2. *Let u be a signed double occurrence string and φ an assembly strategy for u. We denote by $\mathcal{C}_2^{(1)}(\varphi)$ the number of* ld, hi, *and* dlad *operations in φ. Then the complexity $\mathcal{C}_2^{(1)}(u)$ is defined as:*

$$\mathcal{C}_2^{(1)}(u) = min\{\mathcal{C}_2^{(1)}(\varphi) \mid \varphi \text{ is an assembly strategy for } u\}.$$

Example 6. Consider $u = 2\bar{3}\bar{2}434$ and φ_1, φ_2 are assembly strategies for u as in Example 4. Then $\mathcal{C}_2^{(1)}(\varphi_1) = 2$, while $\mathcal{C}_2^{(1)}(\varphi_2) = 3$. It is easy to see that $\mathcal{C}_2^{(1)}(u) = 2$.

Clearly, to find the complexity $\mathcal{C}_2^{(1)}(u)$ for a given string u, one needs to find the length of an assembly strategy φ for u using maximum number of dlad operations. Indeed, note that ld and hi operations reduce the length of the string by two, while dlad operations reduce the length of the string by four.

Finding the complexity $\mathcal{C}_2^{(1)}(u)$ is easy if $\mathcal{C}_1(u) \neq \{\mathsf{Hi}, \mathsf{Dlad}\}$. Indeed, based on Theorem 1, it is easy to see that in this case, for any two assembly strategy φ and ψ for u, we have $\mathcal{C}_2^{(1)}(\varphi) = \mathcal{C}_2^{(1)}(\psi)$. It is currently unknown how to compute $\mathcal{C}_2^{(1)}(u)$ if $\mathcal{C}_1(u) = \{\mathsf{Hi}, \mathsf{Dlad}\}$.

Considering the molecular model of the dlad operations, with a double fold and two simultaneous recombination, it may sometimes not be desirable to maximize the number of dlad operations as done when computing $\mathcal{C}_2^{(1)}(u)$. A different idea is to associate *weights* to each ld, hi and dlad operation and consequently, to any assembly strategy. Associating weights to the operations may be done in at least two ways: either by introducing a (fixed) weight for each type of operation, or through variable weights depending on the type of operation and the string to which the operation applies. We illustrate both ideas in the following.

Definition 3. *For any operation $f \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$, we define $\mathcal{C}_2^{(2)}(f)$ as follows: $\mathcal{C}_2^{(2)}(f) = c_1$, if $f \in \mathsf{Ld}$; $\mathcal{C}_2^{(2)}(f) = c_2$, if $f \in \mathsf{Hi}$; and $\mathcal{C}_2^{(2)}(f) = c_3$, if $f \in \mathsf{Dlad}$, where $c_1, c_2, c_3 \geq 0$. Then for a composition $\varphi = f_k \circ \cdots \circ f_1$, $f_i \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$, we let $\mathcal{C}_2^{(2)}(\varphi) = \sum_{i=1}^{k} \mathcal{C}_2^{(2)}(f_i)$.*

For a signed double occurrence string u, the complexity $\mathcal{C}_2^{(2)}(u)$ is defined as

$$\mathcal{C}_2^{(2)}(u) = min\{\mathcal{C}_2^{(2)}(\varphi) \mid \varphi \text{ is an assembly strategy for } u\}.$$

Example 7. Let $u = 2\bar{3}\bar{2}434$ and φ_1, φ_2 are assembly strategies for u as in Examples 4 and 6. If we define $\mathcal{C}_2^{(2)}(f) = 1$, for any $f \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$, then $\mathcal{C}_2^{(2)} = \mathcal{C}_2^{(1)}$: we only count the number of operations in each strategy. Consider now that $\mathcal{C}_2^{(2)}(f) = 0$ if $f \in \mathsf{Ld}$, and $\mathcal{C}_2^{(2)}(f) = 1$ if $f \in \mathsf{Hi}$, and $\mathcal{C}_2^{(2)}(f) = 3$ if $f \in \mathsf{Dlad}$, then $\mathcal{C}_2^{(2)}(\varphi_1) = 4$ and $\mathcal{C}_2^{(2)}(\varphi_2) = 3$.

A more refined measure of complexity may be introduced depending on the length of the strings "manipulated" by each operation: the length of the string inverted by hi_p, and the length of the strings translocated by $\mathsf{dlad}_{p,q}$. In the case of ld_p, the excised string is always the same, pp and so, for simplicity, we may associate it complexity zero. We formally define this complexity measure in the following.

Definition 4. *Let u be a signed double occurrence string.*

(i) For any operation ld_p applicable to u, we let $\mathcal{C}_2^{(3)}(\mathsf{ld}_p, u) = 0$.
(ii) For any operation hi_p applicable to u, we let $\mathcal{C}_2^{(3)}(\mathsf{hi}_p, u) = |u_2|$, where $u = u_1 p u_2 \overline{p} u_3$, for some strings u_1, u_2, u_3.
(iii)For any operation $\mathsf{dlad}_{p,q}$ applicable to u, we let $\mathcal{C}_2^{(3)}(\mathsf{dlad}_{p,q}, u) = |u_2|+|u_4|$, where $u = u_1 p u_2 q u_3 p u_4 q u_5$, for some strings u_1, u_2, u_3, u_4, u_5.

For an assembly strategy $\varphi = f_k \circ \cdots \circ f_1$ for u, $f_i \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$, we let: $\mathcal{C}_2^{(3)}(\varphi, u) = \sum_{i=1}^{k} \mathcal{C}_2^{(3)}(f_i, (f_{i-1} \circ \cdots f_1)(u))$. Then we define the complexity $\mathcal{C}_2^{(3)}(u)$ as

$$\mathcal{C}_2^{(3)}(u) = min\{\mathcal{C}_2^{(3)}(\varphi, u) \mid \varphi \textit{ is an assembly strategy for } u\}.$$

Example 8. Let $u = 344567567893\overline{3}\overline{2}289$ be the string associated to the gene actin I in S.nova. Then $\varphi_1 = \mathsf{ld}_6 \circ \mathsf{dlad}_{7,5} \circ \mathsf{ld}_4 \circ \mathsf{hi}_2 \circ \mathsf{hi}_8 \circ \mathsf{hi}_9 \circ \mathsf{hi}_3$ is an assembly strategy for u:

$$\begin{aligned}
u_1 &= \mathsf{hi}_3(u) = \overline{9}\overline{8}\overline{7}\overline{6}\overline{5}\overline{7}\overline{6}\overline{5}\overline{4}\overline{4}\overline{2}289,\\
u_2 &= \mathsf{hi}_9(u_1) = \overline{8}\overline{2}2445675678,\\
u_3 &= \mathsf{hi}_8(u_2) = \overline{7}\overline{6}\overline{5}\overline{7}\overline{6}\overline{5}\overline{4}\overline{4}\overline{2}2,\\
u_4 &= \mathsf{hi}_2(u_3) = \overline{7}\overline{6}\overline{5}\overline{7}\overline{6}\overline{5}\overline{4}\overline{4},\\
u_5 &= \mathsf{ld}_4(u_4) = \overline{7}\overline{6}\overline{5}\overline{7}\overline{6}\overline{5},\\
u_6 &= \mathsf{dlad}_{7,5}(u_5) = \overline{6}\overline{6},\\
u_7 &= \mathsf{ld}_6(u_6) = \Lambda.
\end{aligned}$$

Then $\mathcal{C}_2^{(3)}(\varphi_1, u) = \mathcal{C}_2^{(3)}(\mathsf{hi}_3, u) + \mathcal{C}_2^{(3)}(\mathsf{hi}_9, u_1) + \mathcal{C}_2^{(3)}(\mathsf{hi}_8, u_2) + \mathcal{C}_2^{(3)}(\mathsf{hi}_2, u_3) + \mathcal{C}_2^{(3)}(\mathsf{ld}_4, u_4) + \mathcal{C}_2^{(3)}(\mathsf{dlad}_{7,5}, u_5) + \mathcal{C}_2^{(3)}(\mathsf{ld}_6, u_6) = 10 + 12 + 10 + 0 + 0 + 2 + 0 = 34$.

Note that $\varphi_2 = \mathsf{hi}_3 \circ \mathsf{hi}_2 \circ \mathsf{dlad}_{8,9} \circ \mathsf{ld}_7 \circ \mathsf{dlad}_{5,6} \circ \mathsf{ld}_4$ is also an assembly strategy for u:

$$\begin{aligned}
v_1 &= \mathsf{ld}_4(u) = 356756789\overline{3}\overline{2}289,\\
v_2 &= \mathsf{dlad}_{5,6}(v_1) = 37789\overline{3}\overline{2}289,\\
v_3 &= \mathsf{ld}_7(v_2) = 389\overline{3}\overline{2}289,\\
v_4 &= \mathsf{dlad}_{8,9}(v_3) = 3\overline{3}\overline{2}2,\\
v_5 &= \mathsf{hi}_2(v_4) = 3\overline{3},\\
v_6 &= \mathsf{hi}_3(v_5) = \Lambda.
\end{aligned}$$

Then $\mathcal{C}_2^{(3)}(\varphi_2, u) = \mathcal{C}_2^{(3)}(\mathsf{ld}_4, u) + \mathcal{C}_2^{(3)}(\mathsf{dlad}_{5,6}, v_1) + \mathcal{C}_2^{(3)}(\mathsf{ld}_7, v_2) + \mathcal{C}_2^{(3)}(\mathsf{dlad}_{8,9}, v_3) + \mathcal{C}_2^{(3)}(\mathsf{hi}_2, v_4) + \mathcal{C}_2^{(3)}(\mathsf{hi}_3, v_5) = 0$.

A natural question here is: what are the strings with minimal $\mathcal{C}_2^{(3)}(u)$ complexity? We give a complete answer for realistic strings in the next section, where we discuss simple operations for gene assembly.

6 Third complexity measure: simple operations

As discussed also above, one way to introduce a complexity measure for gene assembly is by considering the length of the molecular folds involved in every step of the assembly. We consider in this section simple versions of ld, hi, and dlad where the operations can only be applied on the shortest possible folds. It is known that $\mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ is a complete model, in the sense that any gene (alternatively: signed permutation, string, or graph) may be assembled in this model, see [7]. It turns out that the simple operations are not complete: there are certain patterns that cannot be assembled through simple operations. Remarkably though, all known micronuclear gene sequences, see [2], can indeed be assembled through simple operations.

The molecular model for simple ld, hi, and dlad was introduced in [11]. Due to lack of space, we only give here a short intuitive presentation, followed by their formalization as rewriting rules for signed permutations.

The simple operations were modeled in [11] on the level of MDS descriptors, signed permutations, and signed double occurrence strings. We choose here the level of signed permutations, where several results are easiest and most elegant to present.

!!! As observed in Section 3, ld must always be simple – the excised sequences may never contain coding blocks for the assembly to succeed. In simple hi, one only inverts sequences containing *at most one MDS*. Similarly, in simple dlad, the two sequences that are translocated may contain altogether *at most one MDS*. We refer to [11] for details.

As noted in Section 3, when working on signed permutations, we ignore the ld operation and model gene assembly as a process of sorting a signed permutation rather than as a process of pointer elimination. Simple hi and dlad are modeled through the following operations for signed permutations.

1. For each $p \geq 1$, sh_p is defined as follows:

$$\mathsf{sh}_p(xp\ldots(p+i)(\overline{p+k})\ldots(\overline{p+i+1})y) = xp\ldots(p+i)(p+i+1)\ldots(p+k)y,$$
$$\mathsf{sh}_p(x(\overline{p+i})\ldots\overline{p}(p+i+1)\ldots(p+k)y) = xp\ldots(p+i)(p+i+1)\ldots(p+k)y,$$
$$\mathsf{sh}_p(x(p+i+1)\ldots(p+k)(\overline{p+i})\ldots\overline{p}) = x(\overline{p+k})\ldots(\overline{p+i+1})(\overline{p+i})\ldots\overline{p}y,$$
$$\mathsf{sh}_p(x(\overline{p+k})\ldots(\overline{p+i+1})p\ldots(p+i)y) = x(\overline{p+k})\ldots(\overline{p+i+1})(\overline{p+i})\ldots\overline{p}y,$$

   where $k > i \geq 0$ and x, y, z are signed strings over Σ_n. We denote $\mathsf{Sh} = \{\mathsf{sh}_i \mid 1 \leq i \leq n\}$.
2. For each p, $2 \leq p \leq n-1$, sd_p is defined as follows:

$$\mathsf{sd}_p(x\,p\ldots(p+i)\,y\,(p-1)\,(p+i+1)\,z) = xy(p-1)p\ldots(p+i)(p+i+1)z,$$
$$\mathsf{sd}_p(x\,(p-1)(p+i+1)yp\ldots(p+i)z) = x(p-1)p\ldots(p+i)(p+i+1)yz,$$

where $i \geq 0$ and x, y, z are signed strings over Σ_n. We also define $\mathsf{sd}_{\overline{p}}$ as follows:

$$\mathsf{sd}_{\overline{p}}(x(\overline{p+i+1})(\overline{p-1})y(\overline{p+i})\ldots\overline{p}z) = x\,(\overline{p+i+1})(\overline{p+i})\ldots\overline{p}(\overline{p-1})yz,$$
$$\mathsf{sd}_{\overline{p}}(x(\overline{p+i})\ldots\overline{p}y(\overline{p+i+1})(\overline{p-1})z) = xy(\overline{p+i+1})(\overline{p+i})\ldots\overline{p}(\overline{p-1})z,$$

where $i \geq 0$ and x, y, z are signed strings over Σ_n. We denote $\mathsf{Sd} = \{\mathsf{sd}_i, \mathsf{sd}_{\overline{i}} \mid 1 \leq i \leq n\}$.

We say that a signed permutation π over the set of integers $\{i, i+1, \ldots, i+l\}$ is *sortable* if there are operations $\phi_1, \ldots, \phi_k \in \mathsf{Sh} \cup \mathsf{Sd}$ such that $(\phi_k \circ \ldots \circ \phi_1)(\pi)$ is a sorted permutation. We say that π is *blocked* if neither an sh operation, nor an sd operation is applicable to π and π is not sorted.

Let $\phi = \phi_1 \circ \ldots \circ \phi_k$, $\phi_i \in \mathsf{Sh} \cup \mathsf{Sd}$, for all $1 \leq i \leq k$. We say that ϕ is a *strategy* for π if $\phi(\pi)$ is either sorted, or blocked. In the former case we say that ϕ is a *sorting strategy*, while in the latter case we say that ϕ is a *unsuccessful strategy* for π.

Example 9. Let $\pi = 2\,4\,3\,\overline{1}$ be a signed permutation. Then $(\mathsf{sh}_1 \circ \mathsf{sd}_3)(\pi) = \mathsf{sh}_1(2\,3\,4\,\overline{1}) = \overline{4}\,\overline{3}\,\overline{2}\,\overline{1}$, a sorted permutation.

One may introduce “elementary” versions of sh and sd, where only one letter is rewritten in every step, rather than strings as in sh and sd. We consider them as well and draw a comparison between the two models.

3. For each $p \geq 1$, eh_p is defined as follows:

$$\mathsf{eh}_p(x\,p(\overline{p+1})\,y) = x\,p\,(p+1)\,y, \qquad \mathsf{eh}_p(x\,(\overline{p+1})\,p\,y) = x\,(\overline{p+1})\,\overline{p}\,y,$$
$$\mathsf{eh}_p(x\,\overline{p}\,(p+1)\,y) = x\,p\,(p+1)\,y, \qquad \mathsf{eh}_p(x\,(p+1)\,\overline{p}\,y) = x\,(\overline{p+1})\,\overline{p}\,y,$$

where x, y are signed strings over Σ_n. We denote $\mathsf{Eh} = \{\mathsf{sh}_p \mid 1 \leq p \leq n\}$.

4. For each $p \geq 1$, $2 \leq p \leq n-1$, ed_p is defined as follows:

$$\mathsf{ed}_p(x\,p\,y\,(p-1)\,(p+1)\,z) = x\,y\,(p-1)\,p\,(p+1)\,z,$$
$$\mathsf{ed}_p(x\,(p-1)\,(p+1)\,y\,p\,z) = x\,(p-1)\,p\,(p+1)\,y\,z,$$
$$\mathsf{ed}_p(x\,(\overline{p+1})\,(\overline{p-1})\,y\,\overline{p}\,z) = x\,(\overline{p+1})\,\overline{p}\,(\overline{p-1})\,y\,z,$$
$$\mathsf{ed}_p(x\,\overline{p}\,y\,(\overline{p+1})\,(\overline{p-1})\,z) = x\,y\,(\overline{p+1})\,\overline{p}\,(\overline{p-1})\,z,$$

where x, y, z are signed strings over Σ_n. We denote $\mathsf{Ed} = \{\mathsf{sd}_p \mid 1 \leq p \leq n\}$.

Example 10. (a) Let $\pi = 3\,\overline{4}\,\overline{5}\,6\,\overline{1}\,2$. Then $(\mathsf{eh}_1 \circ \mathsf{eh}_6 \circ \mathsf{eh}_4 \circ \mathsf{eh}_3)(\pi) = 3\,4\,5\,6\,1\,2$ is a sorted permutation.
(b) Let $\pi' = 3\,4\,5\,6\,\overline{1}\,\overline{2}$. Then π' is not $\mathsf{Eh} \cup \mathsf{Ed}$-sortable. Indeed, no eh or ed operation is applicable to π'.

Lemma 1. *For any signed permutation π, if eh_p (ed_p, resp.) is applicable to π, for some p, then sh_p (sd_p, resp.) is also applicable to π and $\mathsf{eh}_p(\pi) = \mathsf{sh}_p(\pi)$ ($\mathsf{ed}_p(\pi) = \mathsf{sd}_p(\pi)$, resp.)*

Note that Lemma 1 does not hold in the reverse direction: if $\pi = 1\,4\,2\,3$, then $\mathsf{sd}_2(\pi) = 1\,2\,3\,4$, while ed_2 is not applicable to π.

As illustrated by the next example, it turns out that the $\mathsf{Eh} \cup \mathsf{Ed}$-model is *nondeterministic*.

Example 11. Let $\pi = 1\,3\,5\,2\,4$. Note that π has both sorting and non-sorting strategies in the elementary model. Indeed, $(\mathsf{ed}_2 \circ \mathsf{ed}_4)(\pi) = 1\,2\,3\,4\,5$, a sorted permutation. On the other hand, $\pi' = \mathsf{ed}_3(\pi) = 1\,5\,2\,3\,4$ is not sorted and no eh or ed operation is applicable to π'.

Due to nondeterminism, deciding whether a given permutation is Eh-, Ed-, or $\mathsf{Eh} \cup \mathsf{Ed}$-sortable is difficult. A complete answer may be found in [10], based on an involved notion of dependency graph.

The simple model however is different. A permutation may indeed have several different strategies, but they are either all sorting, or all non-sorting. Moreover, [13] also defines a notion of *structure of a permutation* and notes that the results obtained after applying these strategies, though different, *have the same structure*. In this way, deciding whether or not a given permutation π is Sh-, Sd-, or $\mathsf{Sh} \cup \mathsf{Sd}$-sortable is easy: simply apply operations from the desired set in an arbitrary order; if the final blocked permutation is sorted, then the answer is 'yes', otherwise the answer is 'no': there are no sorting strategies for π.

Example 12. (a) The permutation $\pi_1 = 4\overline{6}71\overline{2}35$ has several sorting strategies. Some of them are shown below.

$$\begin{aligned}
\pi_1^{(1)} &= \mathsf{sd}_5 \circ \mathsf{sh}_6 \circ \mathsf{sh}_1(\pi_1) = 4567123,\\
\pi_1^{(2)} &= \mathsf{sd}_5 \circ \mathsf{sh}_6 \circ \mathsf{sh}_2(\pi_1) = 4567123,\\
\pi_1^{(3)} &= \mathsf{sd}_4 \circ \mathsf{sh}_6 \circ \mathsf{sh}_2(\pi_1) = 6712345,\\
\pi_1^{(4)} &= \mathsf{sh}_6 \circ \mathsf{sh}_1 \circ \mathsf{sd}_4(\pi_1) = 6712345.
\end{aligned}$$

(b) The permutation $\pi_2 = 13685724$ has several unsuccessful strategies. Some of them are shown below.

$$\begin{aligned}
\pi_2^{(1)} &= \mathsf{sd}_2 \circ \mathsf{sd}_7(\pi_2) = 12367854,\\
\pi_2^{(2)} &= \mathsf{sd}_2 \circ \mathsf{sd}_6(\pi_2) = 12385674,\\
\pi_2^{(3)} &= \mathsf{sd}_3 \circ \mathsf{sd}_7(\pi_2) = 18567234,\\
\pi_2^{(4)} &= \mathsf{sd}_3 \circ \mathsf{sd}_6(\pi_2) = 18567234.
\end{aligned}$$

7 Fourth complexity measure: parallelism

The previous three measures of complexity all dealt with *sequential* compositions of operations leading to the assembly of a given gene. We introduce in this section a fourth measure of complexity dealing with more general *parallel assemblies* of genes.

A systematic study of parallelism for gene assembly has been initiated in [15]. We only consider in this paper a graph-based presentation of parallelism, although a string-based study is also possible, see [15].

Intuitively, a set of operations can be applied in parallel to a gene pattern if only if each operation's applicability is independent of the other's. In other words, a number of operations can be applied in parallel to a gene pattern if they can be (sequentially) applied in any order to that gene pattern. Note that this is consistent with how parallelism and concurrency are defined in Computer Science.

E.g., the C_2 gene of S.nova described by the MDS sequence $M_1M_2M_3M_4$ requires three Ld operations. The three Lds can be applied independently of each other and so, they can be applied in parallel. Also, the micronuclear gene R_1 of S.nova described by the MDS sequence $M_1M_2M_3M_4M_5M_6$, requires five Ld operations, and all of them can be applied at once. Consequently, its parallel complexity is one, the same as gene C_2.

Parallelism can be defined in terms of signed graphs as follows.

Definition 5 ([15]). *Let $S \subseteq \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ be a set of k rules and let $G = (V, E, \sigma)$ be a signed graph. We say that the rules in S can be applied in parallel to G if for any ordering $\varphi_1, \varphi_2, \ldots, \varphi_k$ of S, the composition $\varphi_k \circ \cdots \circ \varphi_1$ is applicable to G.*

The following result provides a simple criterium for two rules to be applicable in parallel.

Lemma 2 ([15]). *Let $G = (V, E, \sigma)$ be a signed graph and let $\varphi, \psi \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ be two rules applicable to G with $\mathsf{dom}(\varphi) \cap \mathsf{dom}(\psi) = \emptyset$.*

(i) If $\varphi \in \mathsf{Ld}$, then φ and ψ can be applied in parallel to G.
(ii) If $\varphi = \mathsf{hi}_p$ with $p \in V$, then φ and ψ can be applied in parallel to G if and only if $N_G(p) \cap \mathsf{dom}(\psi) = \emptyset$.
(iii) If $\varphi, \psi \in \mathsf{Dlad}$, then φ and ψ can applied in parallel to G if and only if the subgraph of G induced by $\mathsf{dom}(\varphi) \cup \mathsf{dom}(\psi)$ is not isomorphic to C_4 or D_4. define C_4 and D_4

According to the definition, if a set of rules is applicable in parallel to a signed graph, then any composition of these rules is applicable to that graph. This definition does not require that the result of applying different compositions of rules must be the same. However, it can be proved that this is indeed the case.

Lemma 3 ([15]). *If $\varphi, \psi \in \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ are applicable in parallel to the signed graph G, then $\varphi(\psi(G)) = \psi(\varphi(G))$.*

The general case follows now easily from Lemma 3.

Theorem 4 ([15]). *Let G be a signed graph and let $S \subseteq \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ be a set of rules applicable in parallel to G. Then for any two compositions φ, φ' of the rules in S, $\varphi(G) = \varphi'(G)$.*

Based on Theorem 4 we can now define the notion of parallel complexity.

Definition 6. *Let G be a signed graph. If $S \subseteq \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ is a set of rules applicable in parallel to G, then we say that S is applicable to G and we denote by $S(G)$ the graph obtained as a result of applying to G any sequential composition of the rules in S.*

If $S_1, S_2, \ldots, S_k \subseteq \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ are disjoint sets of rules, $S_i \cap S_j = \emptyset$, for $i \neq j$, we say that $S_k \circ \ldots \circ S_1$ is applicable to G if S_i is applicable to $(S_{i-1} \circ \ldots \circ S_1)(G)$, for all $1 \leq i \leq k$. If $(S_k \circ \ldots \circ S_1)(G) = \emptyset$, then we say that $S_k \circ \ldots \circ S_1$ is a parallel reduction strategy for G. We say that the parallel complexity of $S = S_k \circ \ldots \circ S_1$ is $\mathcal{C}_4(S) = k$.

We define the parallel complexity $\mathcal{C}_4(G)$ of G as follows:

$$\mathcal{C}_4(G) = \min\{\mathcal{C}_4(S) \mid \text{ S is a parallel reduction strategy for G}\}.$$

Example 13. (a) Any discrete graph can be reduced in one parallel step.
(b) The smallest graph with parallel complexity two is shown in Fig. 6(a).
(c) The smallest graph that can be reduced in 3 parallel steps, see Fig. 6(b).

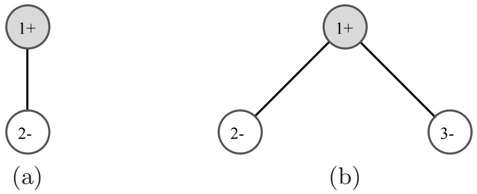


Fig. 6. (a) A graph with parallel complexity two; (b) A graph with parallel complexity three.

Example 14. Let G be the signed overlap graph associated to actin I gene in S. nova, illustrated in Fig. 7. There are only 6 different maximal parallel strategies to reduce G:
$S_1 = \{\mathsf{ld}_7, \mathsf{hi}_3\} \circ \{\mathsf{hi}_2, \mathsf{ld}_4, \mathsf{dlad}_{5,6}, \mathsf{dlad}_{8,9}\}$;
$S_2 = \{\mathsf{ld}_6, \mathsf{hi}_8, \mathsf{hi}_9\} \circ \{\mathsf{hi}_2, \mathsf{hi}_3, \mathsf{ld}_4, \mathsf{dlad}_{5,7}\}$;
$S_3 = \{\mathsf{ld}_6, \mathsf{hi}_3\} \circ \{\mathsf{hi}_2, \mathsf{ld}_4, \mathsf{dlad}_{5,7}, \mathsf{dlad}_{8,9}\}$;
$S_4 = \{\mathsf{ld}_7, \mathsf{hi}_8, \mathsf{hi}_9\} \circ \{\mathsf{hi}_2, \mathsf{hi}_3, \mathsf{ld}_4, \mathsf{dlad}_{5,6}\}$;
$S_5 = \{\mathsf{ld}_5, \mathsf{hi}_3\} \circ \{\mathsf{hi}_2, \mathsf{ld}_4, \mathsf{dlad}_{6,7}, \mathsf{dlad}_{8,9}\}$;
$S_6 = \{\mathsf{ld}_5, \mathsf{hi}_8, \mathsf{hi}_9\} \circ \{\mathsf{hi}_2, \mathsf{hi}_3, \mathsf{ld}_4, \mathsf{dlad}_{6,7}\}$.

Note that there are 3060 sequential strategies to reduce this graph (and assemble the gene), see [6] – the reason for this difference is that many sequential strategies coincide modulo commutation of some rules. Those rules may be applied in parallel.

The following problem seems to be difficult: check whether or not a given set of rules can be applied in parallel to a given signed graph. In the next theorem we give a simple criterium in the case when at most two dlad rules are to be

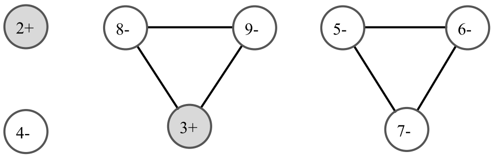


Fig. 7. The signed overlap graph associated to string $3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9$, both representing the structure of the micronuclear gene *actin I* in S.nova.

applied. Giving a general answer, for an arbitrary number of dlad rules, remains an open problem.

Theorem 5. *Let G be a signed graph and $S \subseteq \mathsf{Ld} \cup \mathsf{Hi} \cup \mathsf{Dlad}$ a set of rules containing at most two dlad's. Let P be the union of domains of rules in S with $P^+ = \{p \in P \mid \sigma(p) = +\}$, and $P^- = P \setminus P^+$. Then the rules in S can be applied in parallel to G if and only if the following conditions are satisfied:*

(i) The subgraph induced by P^+ is discrete. Moreover, there is no edge between vertices in P^+ and vertices in P^-.
(ii) The subgraph induced by P^- does not contain induced squares C_4 or diamonds D_4.

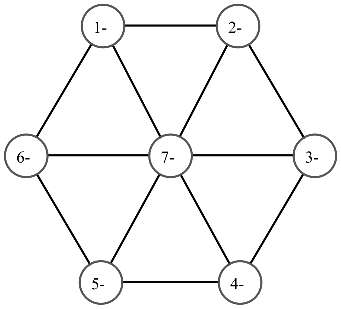


Fig. 8. A negative graph with parallel complexity three.

Two conjectures were given in [15] regarding the parallel complexity of graphs. The authors proposed that any negative graph may be reduced in at most two parallel steps and that any graph may be reduced in at most four parallel steps. Revisiting these conjectures and based on a newly available gene assembly simulator, see [18], we give in the following counterexamples to both these conjectures. It is currently unknown if the parallel complexity of arbitrary graphs is bounded. Several classes of graphs are shown to have bounded parallel complexity in [9].

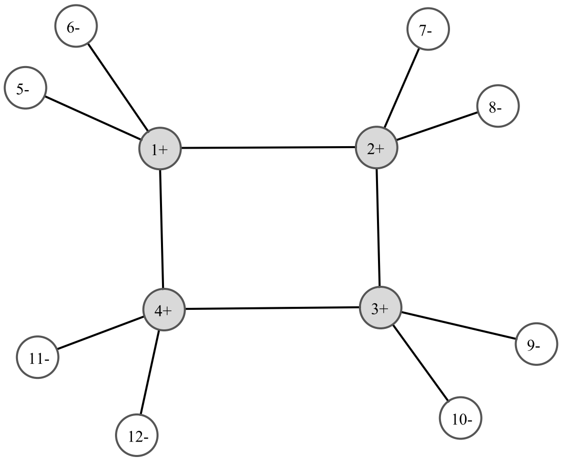


Fig. 9. A graph with parallel complexity five.

Example 15. (a) The negative graph G_3 illustrated in Fig. 8 has parallel complexity three. (As a matter of fact, an automated search shows that this the smallest such graph in terms of number of vertices.) Indeed, one three-step parallel strategy for G_3 is $\{\mathsf{ld}_6\} \circ \{\mathsf{dlad}_{5,7}\} \circ \{\mathsf{dlad}_{1,2}, \mathsf{dlad}_{3,4}\}$. Some straightforward analysis shows that no two-step or one-step strategy for G_3 exists.

(b) The graph G_5 illustrated in Fig. 9 has parallel complexity 5. One 5-step parallel reduction for G_5 is the following: $\{\mathsf{ld}_8, \mathsf{ld}_{12}\} \circ \{\mathsf{ld}_6, \mathsf{ld}_{10}, \mathsf{hi}_7, \mathsf{hi}_{11}\} \circ \{\mathsf{hi}_5, \mathsf{hi}_9\} \circ \{\mathsf{hi}_2, \mathsf{hi}_4\} \circ \{\mathsf{hi}_1, \mathsf{hi}_3\}$.

References

1. Brijder, R., Hoogeboom, H.J., Rozenberg, G., Reducibility of gene patterns in cliates using the breakpoint graph, to appear in *Theoret. Comput. Sci* (2006)
2. Cavalcanti, A., Clarke, T.H., Landweber, L., MDS_IES_DB: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucleic Acids Research* **33** (2005) 396–398.
3. Chang, W.J., Bryson, P.D., Liang, H., Shin, M.K., Landweber, L., The evolutionary origin of a complex scrambled gene. *Proceedings of the National Academy of Sciences of the US* **102**(42) (2005) 15149–15154
4. Chang, W.J., Kuo, S., Landweber, L., A new scrambled gene in the ciliate *Uroleptus. Gene* (2006), to appear
5. Ehrenfeucht, A., Harju, T., Petre, I., and Rozenberg, G. (2002) Characterizing the micronuclear gene patterns in ciliates. *Theory of Comput. Syst.* **35** pp 501–519
6. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G. (2004) Computation in Living Cells: Gene Assembly in Ciliates, Springer
7. Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., Universal and simple operations for gene assembly in ciliates. In: V. Mitrana and C. Martin-Vide (eds.) *Words, Sequences, Languages: Where Computer Science, Biology and Linguistics Meet*, Kluwer Academic, Dortrecht, (2001) pp. 329–342
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) pp. 216–256
9. Harju, T., Li, C., and Petre, I., Results on parallel reductions of signed overlap graphs, manuscript (2006)
10. Harju, T., Petre, I., Rogojin, V., and Rozenberg, G., Simple operations for gene assembly. In: A. Carbone, N. A. Pierce (eds.) *DNA Computing: 11th International Workshop on DNA Computing*, Lecture Notes in Comput. Sci. **3892** (2006), 96 – 111.
11. Harju, T., Petre, I., and Rozenberg, G., Modelling simple operations for gene assembly. In: J.Chen, N.Jonoska, G.Rozenberg (Eds.) *Nanotechnology: Science and Computation* (2006) 361–376
12. Jahn, C. L., and Klobutcher, L. A., Genome remodeilng in ciliated protozoa. *Ann. Rev. Microbiol.* **56** (2000), 489–520.
13. Langille, M., Petre, I. (2006) Simple gene assembly is deterministic. *Fundamenta Informaticae* IOS Press
14. Harju, T., Petre, I., Rogojin, V., and Rozenberg, G. (2006), Simple operations for gene assembly, In: Proceedings of the 11th International Meeting on DNA-based computers DNA11 *Lecture Notes in Computer Science* (2006) Springer
15. Harju, T., Li, C., Petre, I., and Rozenberg, G., Parallelism in gene assemby, In: Proceedings of the 10th International Meeting on DNA-based computers DNA 10, Milan, Italy, *Lecture Notes in Computer Science* **3384** (2005) 140–150
16. 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) pp. 3–15
17. 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)
18. Petre, I., Skogman, S. (2006) Gene assembly simulator. `http://combio.abo.fi/simulator/simulator.php`

19. Prescott, D. M., The DNA of ciliated protozoa. *Microbiol. Rev.* **58**(2) (1994) 233–267
20. Prescott, D. M., DNA manipulations in ciliates. In: W.Brauer, H.Ehrig, J.Jarhumäki, A.Salomaa (eds.) *Formal and Natural Computing: essays dedicated to Grzegorz Rozenberg*, LNCS 2300, Springer (2002) 394–417
21. Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260
22. Swanton, M.T., Heumann, J.M., Prescott, D.M., Gene-sized DNA molecules of the macronuclei in three species of hypotrichs: size distribution and absence of nicks. Chromosoma **77** (1980) 217–227
23. Yao, M.C., Fuller, P., Xi, X., Programmed DNA Deletion As an RNA-Guided System of Genome Defense, *Science* 300 (2003) 1581–1584