IPI Letters

ISSN 2976 - 730X
IPI Letters 2025, Vol 3 (1):1-6
https://doi.org/10.59973/ipil.157

Received: 2025-01-02
Accepted: 2024-01-05
Published: 2024-01-08

Editorial Article

On the "Assembly Theory and its Relationship with Computational Complexity"

Szymon Łukaszyk[1,*]

[1]Łukaszyk Patent Attorneys, Głowackiego 8, 40-052 Katowice, Poland

*Corresponding author: szymon@patent.pl

Abstract -The study provides an alternative definition of the assembly space as an acyclic, 2-in-regular digraph of strings provided with an edge labeling map that preserves the commutativity of an assembly step but defines the order of concatenation of strings in this step. Remarkably, the uniqueness of each vertex is the sufficient criterion to establish if an assembly step is allowed and to introduce the notion of an assembly pool: unit-length strings cannot be assembled from shorter strings and, hence, are inaccessible, forming the initial assembly pool, and strings present in the assembly space can not be *assembled again*, possibly using different pathways, as they would not be unique. What is allowed is the evolution of assembly pathways to make them shorter. We also comment on certain results of [1], showing that the Assembly Steps Problem, not the Assembly Index Problem, has been proved in the referenced study to be NP-complete.

Keywords - Assembly theory; Assembly index; Information theory; Complexity measures; Mathematical physics.

1 Introduction

A recent study [1] shows that assembly theory offers a distinct approach and answers different questions than computational complexity theory with its focus on minimum descriptions via compressibility and discusses fundamental differences in the ontological basis of assembly theory and the assembly index as a physical observable, which distinguish it from theoretical approaches to formalizing life that are unmoored from measurement. Furthermore, the study [1] claims to contain the proof of the conjecture posed in [2] that the Assembly Index Problem is NP-Complete. In general, it argues to show that any instance of the Vertex Cover Problem, which is known to be NP-Hard, can be reformulated as an instance of the Assembly Index Problem. Finally, it leaves open the question of whether the noncommutative concatenation version of string assembly is NP-Complete; its authors see no alternative to reconcile the commutativity of the assembly step with the noncommutativity of the string concatenation (cf. [1], Supplementary Material, footnote 10[1].).

Here, we show that such a reconciliation is, in fact, possible by providing an alternative definition of the assembly space. We also show that it is the Assembly Steps Problem, not the Assembly Index Problem, which has been shown in [1] to be NP-complete. Hence, finding if a string can be assembled along a given path in a given number of steps is NP-complete. But we do not know if finding the minimal number of assembly steps leading to this string, that is, the assembly index of this string, is NP-complete.

[1]Here, all references to [1] relate to the Supplementary Material of [1].

2 An Alternative Definition of an Assembly Space

An assembly space $\Omega = (\Gamma, \phi)$ is defined in [1] (cf. Definition 8) as an acyclic directed graph $\Gamma = (V, E)$, where V is the set of vertices and E is the set of edges together with an edge labeling map $\phi : E \ni e \to v \in V$. Γ contains a finite and non-empty set of vertices $\mathrm{Src}(\Gamma) = B_\Omega$ that form the basis of Ω, each reachable only from itself. All remaining vertices of Γ are reachable from a vertex in B_Ω.

The edge labeling map ϕ cleverly defines the assembly step. Namely

$$e = (x, z) \in E(\Omega) \wedge \phi(e) = y \in V(\Omega) \quad \Leftrightarrow \quad \exists\, e' = (y, z) \in E(\Omega) \;:\; \phi(e') = x, \tag{1}$$

allowing to write $e = (x_y, z)$, $e' = (y_x, z)$, and

$$e = (x_y, z) \in E(\Omega) \quad \Leftrightarrow \quad e' = (y_x, z) \in E(\Omega). \tag{2}$$

However, the commutativity of the relation (2) cannot imply the commutativity of the string concatenation. If all assembly *objects* in $V(\Omega)$ are strings $(x_y, xy) \in E(\Omega)$ does not imply $(y_x, yx) \in E(\Omega)$, as might be expected for string concatenation, but it actually implies also $(y_x, xy) \in E(\Omega)$. Therefore, a string assembly space is endowed in [1] with the additional property (cf. [1], Definition 19 and Eq. (10))

$$(x_y, z) \in E(\Omega) \quad \Rightarrow \quad z = [xy] \vee z = [yx], \tag{3}$$

and likewise for the 2nd edge $(y_x, z) \in E(\Omega)$.

However, the property (3) makes the terminating string z unresolvable if $x \neq y$. Consequently, z cannot be used in subsequent assembly operations. Here, we provide an alternative definition of the assembly space.

Definition 1 (Assembly Space). *An assembly space $\Omega = (C, E, \phi)$ is an acyclic digraph of strings $C = \{C_k\}$, $k \in \mathbb{N}$ where all $b \in \mathbb{N}$ unit length strings (basic symbol(s)) are inaccessible source vertices and the remaining strings are 2-in-regular assembly steps vertices, E is a set of edges, and $\phi : E \ni e \to C_k \in C$ is an edge labeling map, wherein an assembly step $s > 0$ consists of forming a new string C_z from two not necessarily different $s-1+b$ strings C_x, C_y by concatenating them with each other, establishing edges $e = (C_x, C_z)$ and $e' = (C_y, C_z)$, and assigning, strings C_x, C_y to edges e', e using ϕ as*

$$\begin{aligned} C_z = C_x \circ C_y = strcat(C_x, C_y) \quad &\Leftrightarrow \quad \phi(e) = C_y \wedge \phi(e') = -C_x, \\ C_z = C_y \circ C_x = strcat(C_y, C_x) \quad &\Leftrightarrow \quad \phi(e) = -C_y \wedge \phi(e') = C_x, \end{aligned} \tag{4}$$

where $\circ$ denotes the string concatenation (strcat) operator.

The definition of edge labeling map (4) is possible if only $b > 1$, i.e., for more than one basic symbol, as in that case we can say that a given inaccessible symbol is the 1st one, another is the 2nd one, and so on; we can sort them. Otherwise, the notion of a *concatenation direction* is pointless for one symbol only. Contrary to the previous definition of the labeling map ϕ [1], the relation (4) preserves the commutativity of the assembly step but defines the order of concatenation of the strings, as - in general - for different strings $C_x \neq C_y \Leftrightarrow C_x \circ C_y \neq C_y \circ C_x$. The relation (4) is superfluous if the vertices defining the directed edges of Ω are strings, as any edge $e = (C_x, C_z)$ unambiguously resolves to either $e = (C_x, C_x \circ C_y)$ or $e = (C_x, C_y \circ C_x)$. For example, the edge $e = ([010], [0101])$ unambiguously resolves to $e = ([010], [010] \circ [1])$. However, this relation is convenient, if the strings are associated with numerical labels.

The definition 1 is consistent: all vertices are unique (in any standard graph all vertices should be unique), and all are strings. Since an assembly step always consists of joining two parts only [4], this can be thought of as the left and right fragments of the newly formed string [3], and those strings that can be the result of concatenation of two shorter strings are assembly step 2-in-regular vertices, while unit-length strings are inaccessible. Remarkably, the uniqueness of each vertex is the sufficient criterion to establish if an assembly step is allowed (cf. [1], Definition 10) and to introduce the notion of an assembly pool: vertices (strings) present in the assembly space can not be *assembled again*, possibly using different pathways, as they would not be unique; they can only be used in assembly of other strings. What is allowed is the evolution of assembly pathways to make them shorter, as shown in Figure 1. This evolution seems to be stimulated by the trend to decrease the assembly depth [3].

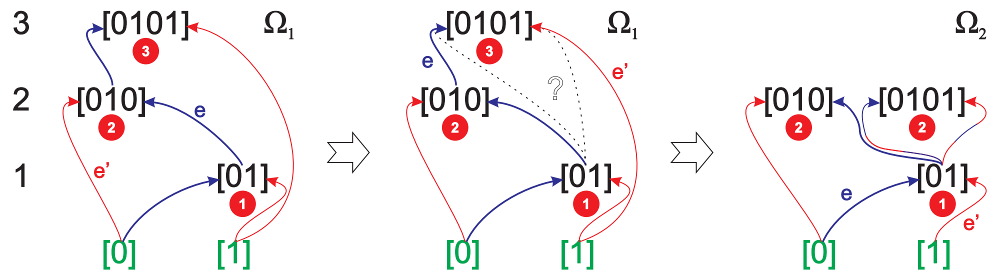

Figure 1: The evolution of assembly pathways leading to a string [0101] to make them shorter and minimize the assembly depth, also illustrating the definition of the edge labeling map ϕ for $C_x \neq C_y$: $\phi(e) = C_y \Rightarrow C_z = C_x \circ C_y$, $\phi(e') = -C_y \Rightarrow C_z = C_y \circ C_x$ ("blue arrow goes first" rule). Numbers in red discs denote the number of steps.

3 The Assembly Steps Problem is NP-Complete

In order to show that any instance of the Vertex Cover Problem (G, k), where $G = (V, E)$ is a graph, $V(G)$ is the set of vertices and $E(G)$ is the set of edges and k is the cardinality of a set of vertices that includes at least one vertex of every edge of G, which is known to be NP-Hard, can be reformulated in polynomial time as an instance of the Assembly Index Problem, the following procedure is offered in [1] (cf. Section 4.2). For a given instance of the Vertex Cover Problem (G, k), where $\tau \leq k < |V(G)|$, and τ is the vertex cover number (the size of a minimum vertex cover), an instance of the Assembly Index Problem $(\Omega, C_x, a^{(N,b)}(C_x))$ is constructed, where Ω is a constructed assembly space, and C_x is the target string for which the assembly index $a^{(N,b)}(C_x)$ is to be determined. It is then claimed that a certificate for the Vertex Cover Problem

$$C = \{v_l \in V(G) \mid C_l = [0v_l0] \in V(\Omega')\} \tag{5}$$

containing a subset $\{v_l\}$ of vertices of G that includes at least one vertex of every edge of G can be used to produce a certificate $(\Omega', C_x, a^{(N,b)}(C_x))$ for the Assembly Index Problem and vice versa, where $\Omega' \subset \Omega$ is a rooted subspace (cf. [1] Definition 15) of the assembly space Ω containing only a proper subset $\{C_l = [0v_l0]\}$ of the strings of the form $C_k = [0v_k0], \forall v_k \in V(G)$. Hence, such an instance of the Assembly Index Problem would be logically equivalent to an instance of the Vertex Cover Problem from which it was constructed.

The construction of C_x (cf. [1], Section 4.2) begins with defining the basis of the assembly space Ω (cf. [1] Eqs. (17), (50)), i.e., the unit-length strings

$$B_\Omega = \{0, 1, 2, \ldots, |V(G)|\}, \tag{6}$$

containing $|V(G)|$ symbols of vertices $V(G)$, and a special symbol that here we call "0" (it is defined as "#" in [1]). Hence, $b = |B_\Omega| = |V(G)| + 1$. Then, a set of $3|V(G)|$ vertex strings

$$C_{k1} = [0v_k], \quad C_{k2} = [v_k0], \quad C_{k3} = [0v_k0], \quad \forall v_k \in V(G) \tag{7}$$

is assembled (cf. [1] Eq. (18)). Subsequently, a set of $|E(G)|$ edge strings

$$C_j = [0v_{s_j}0v_{t_j}0], \quad \forall e_j = (v_{s_j}, v_{t_j}) \in E(G) \tag{8}$$

is assembled (cf. [1] Eq. (19)). The last step of the construction of C_x is a sequence of $2|V(G)|$ strings

$$\begin{aligned} S_0 &= 0, \\ S_k &= S_k \circ [0v_k], \forall v_k \in V(G), \\ S_{|V(G)|+1} &= S_{|V(G)|} \circ [v_10], \\ S_{|V(G)|+k} &= S_{|V(G)|+k-1} \circ [v_k0], \forall v_k \in V(G), \end{aligned} \tag{9a}$$

and $|E(G)|$ strings

$$\begin{aligned} S_{2|V(G)|+1} &= S_{2|V(G)|} \circ [0v_{s_1}0v_{t_1}0], \\ S_{2|V(G)|+j} &= S_{2|V(G)|+j-1} \circ [0v_{s_j}0v_{t_j}0], \forall e_j = (v_{s_j}, v_{t_j}) \in E(G), \end{aligned} \tag{9b}$$

defined in [1] by Eqs. (20)-(25), where the target string $C_x = S_{2|V(G)|+|E(G)|}$ is defined as the last string of this sequence and

$$|V(\Omega)| = |V(G)| + 1 + 3|V(G)| + |E(G)| + 2|V(G)| + |E(G)| = 6|V(G)| + 2|E(G)| + 1. \tag{10}$$

Finally ([1], Section 4.2.3) it is claimed that given $(\Omega', C_x, n_{\text{vcp}}(C_x))$ is a certificate for the Assembly Index Problem if the set (5) is a vertex cover of G with size k, i.e. a certificate for the Vertex Cover Problem is given, wherein $n_{\text{vcp}}(C_x)$ is the assembly index of string C_x and

$$n_{\text{vcp}}(C_x) = 4|V(G)| + 2|E(G)| + k \quad \Leftrightarrow \quad |V(\Omega')| = n_{\text{vcp}}(C_x) + |B_{\Omega'}| = 5|V(G)| + 2|E(G)| + k + 1, \tag{11}$$

which depends on k and is minimal if $k = \tau$.

By construction, the basic symbols (6), the edge strings (8), and the sequence strings (9a) and (9b) contained in Ω must also be contained in Ω' (certificate). However, the vertex strings (7) of the form $[0v_k0]$ are the exception, as each of the edge strings (8) can be assembled from strings (7) in one of the two mutually exclusive steps (cf. [1] Eqs. (53), (54))

$$[0v_{s_j}] \circ [0v_{t_j}0] = [0v_{s_j}0v_{t_j}0] \quad \text{or} \quad [0v_{s_j}0] \circ [v_{t_j}0] = [0v_{s_j}0v_{t_j}0], \tag{12}$$

leaving some of the strings $[0v_{s_j}0]$ or $[0v_{t_j}0]$ redundant. It can be seen by comparing the cardinalities of the spaces Ω (10) and Ω' (11), which - as expected - leads to $k < |V(G)|$. There are $3|V(g)|$ strings (7) in Ω and only $2|V(g)| + k$ such strings in Ω'.

By construction (9a), (9b), the target string has the form

$$C_x = C_{\text{vs}} \circ C_{\text{es}} = [001020\ldots(|V(G)|-1)0|V(G)|1020\ldots(|V(G)|-1)0|V(G)|0] \circ C_{\text{es}}, \tag{13}$$

where the $C_{\text{vs}} = S_{|V(G)|+k}$ is a vertex-specific part of C_x depending solely on $|V(G)|$ and its explicit form is given by the formula (9a), and the C_{es} is an edge-specific part of C_x, generated by the formula (9b), and depending both on $|E(G)|$, edge vertex assignments and the order of labeling of the edges of graph G. However, $|C_{\text{es}}| = 5|E(G)|$, as the length of each edge string (8) is five and there are $|E(G)|$ such strings in C_x and Ω'. Therefore, the length of the target string is

$$N_x = |C_{\text{vs}}| + |C_{\text{es}}| = (4|V(G)| + 1) + 5|E(G)|. \tag{14}$$

Furthermore, by construction C_{vs} contains two copies of the string $C_{\text{lng}} = [1020\ldots(|V(G)| - 1)0|V(G)|]$ of length $N_{\text{lng}} = 2|V(G)| - 1$ having the assembly index equal to $a_{\text{lng}} = 2|V(G)| - 2$ as it does not contain any repetitions of substrings. We can take advantage of the fact that each m copies of an n-plet C_n contained in a string decrease the assembly index of this string at least by $m(n-1) - a(C_n)$ [3], where $a(C_n)$ is the assembly index of this n-plet, to estimate the upper bound for the assembly index of C_x reduced by the presence of these two copies of C_{lng}. Furthermore, excluding the degenerate cases of empty and disjoint graphs G, we can further infer some information about C_x. That is, since any vertex $v_k \in V(G)$ is a part of some edge $e_j \in E(G)$, C_x contains at least two repetitions of doublets $[c_l0]$ (or $[0c_l]$), with $l = 1, 2, \ldots, |V(G)| - 1$ as the string C_{lng} also contains $|V(G)| - 1$ such doublets, and each repetition decreases the assembly index by one. Hence, the upper bound must be further decreased by $|V(G)| - 1$. Finally, each string C_x contains $|E(G)| + 1$ repetitions of a doublet [00] and, hence, the upper bound must be further decreased by $|E(G)|$. Therefore, the initial upper bound on the assembly index that amounts to $N_x - 1$ [4] if $N_x \leq b^2 + b + 1$ [3] decreases to

$$n_{\text{dec}}(C_x) \leq (N_x - 1) - [2(2|V(G)| - 2) - (2|V(G)| - 2)] - (|V(G)| - 1) - |E(G)| = |V(G)| + 4|E(G)| + 3, \tag{15}$$

which, in contrast to $n_{\text{vcp}}(C_x)$ (11), is independent of k.

We have examined a few simple graphs, shown in Figure 2, obtaining the results listed in Table 1. As an example, consider the trivial graph $G = (V, E)$, shown in Figure 2(b) having two edges connected at one vertex. Hence, its vertex cover number is $\tau = 1$. In this case, (10) $|V(\Omega)| = 6 \cdot 3 + 2 \cdot 2 + 1 = 23$ and the target string generated by sequences (9a) and (9b) has the form

$$C_x = [00102031020300102002030]. \tag{16}$$

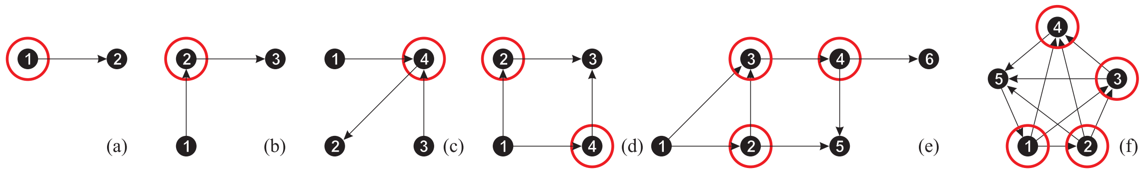

Figure 2: Simple graphs we examined: one edge (a), two edges (b), three edges (c), square (d), "EM rocket"(e), a complete graph K_5 (f). Red circles indicate a minimum vertex cover.

Table 1: Assembly indices $a(C_x)$ of target strings C_x constructed [1] for the Vertex Cover Problem graphs G; minimum and maximum assembly indices $a_{\min}^{(N_x)}$, $a_{\max}^{(N_x,b)}$ for N_x; numbers of steps leading to C_x used in [1] ($n_{\text{vcp}}(C_x)$) and derived here ($n_{\text{dec}}(C_x)$), for examined graphs.

Graph	$\|V(G)\|$	$\|E(G)\|$	τ	N_x	$a_{\min}^{(N_x)}$	$a(C_x)$	$n_{\text{dec}}(C_x)$	$n_{\text{vcp}}(C_x)$	$a_{\max}^{(N_x,b)}$
one edge	2	1	1	14	5	8	9	11	12
two edges	3	2	1	23	7	13	14	17	21
three edges	4	3	1	32	5	18	19	23	30
square	4	4	2	37	7	20	23	26	33
"EM rocket"	6	7	3	60	7	32	37	41	58
K_5	5	10	4	71	9	40	48	44	< 56

As the vertex cover of the graph G is the vertex 2, the subspace Ω' (the certificate) is devoid of triplets [010] and [030], since the edges $(1,2)$ and $(2,3)$ share the vertex 2, and the edge strings (8) could be assembled as [01|020] and [020|30]. Therefore, the number of steps on the assembly pathway of C_x defined by Ω', given by the relation (11), amounts to $n_{\text{vcp}}(C_x) = |\Omega| - |B_\Omega| - 2 = 17$, as shown in Figure 3(a) also illustrating the assembly depth [5] ($d(C_x) = 9$) of this string: 7 steps (1-7) for vertex strings (7), 2 steps (8, 9) for edge strings (8), 6 steps (10-15) for sequence strings (9a), and 2 steps (16, 17) for sequence strings (9b)) which corresponds to the vertex cover number $\tau = 1$, if only the string (16) is assembled using the set of allowed assembly operations defined by the equations Eqs. (38)-(45) of [1].

However, imposing such a set of allowed assembly steps deviates from the principles of assembly theory that assume the possibility of assembling any *object* from any two *objects* in the assembly pool. Even if we assume that only some steps are allowed and some are not due to peculiarities of the assembled data structures, this is certainly not the case for strings, considered in [1] in the proof of Lemma 3. All strings are possible and mathematically well defined [6]. What could be the reason for allowing the assembly of a string $[ABC]$ and disallowing the assembly of a string $[BCA]$ from a set of basic symbols $\{A, B, C\}$?

Therefore, the assembly index of the string (16) is $a(C_x) = 13$. One of the shortest pathways of the string (16) is shown in Figure 3(c) with $d(C_x) = 10$. A quadruplet [1020] present in two independent copies is assembled in step 5, 5-plet $C_{\text{lng}} = [10203]$ present in two copies is assembled in step 6. Furthermore, C_x contains two independent copies of [00] (at the beginning) and [20] (at the end). A slightly longer pathway leading to the string of length (15) is shown in Figure 3(b).

4 Conclusions

The study [1] shows that the Assembly Steps Problem, that is, a problem of determining if a given string can be assembled in a given number of steps according to principles of assembly theory, can be reformulated as the NP-hard Vertex Cover Problem, that is a problem of determining if a given set of vertices of a graph contains at least one vertex of each edge of this graph, and hence the former problem is also NP-hard. Furthermore, the study [1] shows that, since a proposed solution (certificate) to the Assembly Steps Problem can be checked for correctness in polynomial time, the Assembly Steps Problem becomes NP-complete.

However, based on a few simple graphs, we have shown here that the proposed construction of an assembly space from a graph (a certificate) to map the correspondence between the Minimum Vertex Cover Problem and the Assembly Steps Problem does not reflect the shortest pathway leading to this string and hence does not correspond to the Assembly Index Problem.

In this study, we have also answered in the affirmative the question posed in [1]: using the

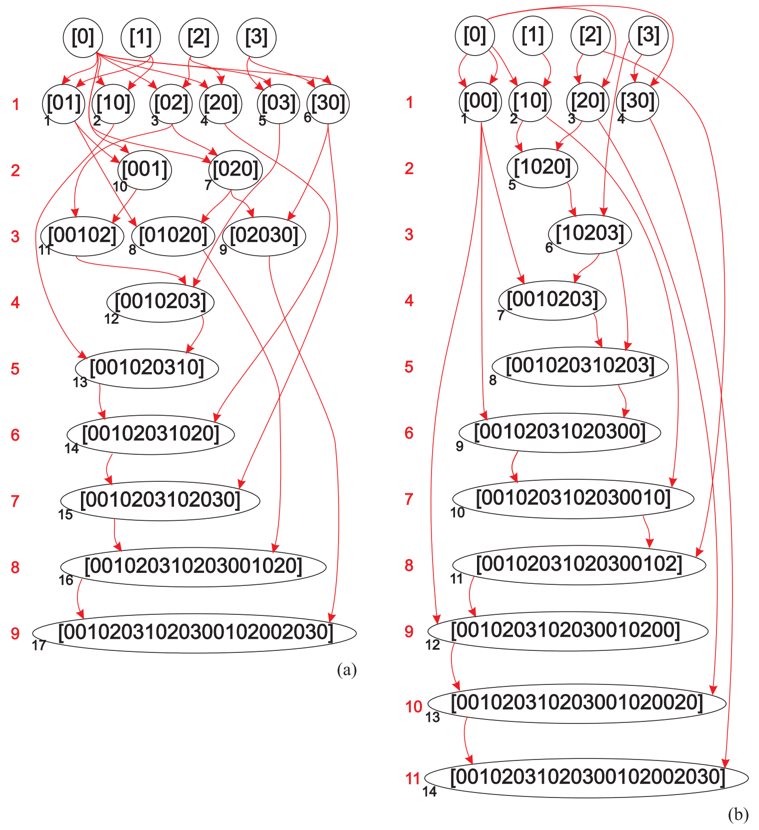

Figure 3: Three assembly spaces of the same string C_x = [0010203102030010202030]: the pathway to produce a vertex cover certificate [1] ($n_{\text{vcp}}(C_x) = 17$ steps) (a), the pathway taking into account the general distributions of substrings in all strings C_x ($n_{\text{dec}}(C_x) = 14$ steps) (b), a shortest, assembly index pathway ($a(C_x) = 13$ steps) (c).

definition of the assembly space of strings 1 and the procedure of constructing the Assembly Steps Problem to correspond to the Vertex Cover Problem [1] shows that the noncommutative concatenation version of the Assembly Steps Problem is also NP-Complete.

Unfortunately, we still do not know if the Assembly Index Problem is NP-Complete.

Acknowledgment

The author thanks Mariola Bala, Wawrzyniec Bieniawski, and Piotr Masierak for their motivation, critical discussions, and numerous clarity, reasoning, and grammar corrections. He also thanks his wife, Magdalena Bartocha, for her everlasting support and his partner and friend, Renata Sobajda, for her prayers.

References

[1] C. P. Kempes, M. Lachmann, A. Iannaccone, G. M. Fricke, M. R. Chowdhury, S. I. Walker, and L. Cronin, Assembly Theory and its Relationship with Computational Complexity, (2024). `https://doi.org/10.48550/ARXIV.2406.12176`

[2] S. Łukaszyk, W. Bieniawski, Assembly Theory of Binary Messages, Mathematics, (2024). `https://doi.org/10.3390/math12101600`

[3] W. Bieniawski, P. Masierak, A. Tomski, and S. Łukaszyk, Assembly Theory. Formalizing Assembly Spaces and Discovering Patterns and Bounds (2024). `https://doi.org/10.20944/preprints202409.1581.v8`

[4] S.M. Marshall, A.R.G. Murray, L. Cronin, A Probabilistic Framework for Identifying Biosignatures Using Pathway Complexity, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 375, 20160342 (2017). `https://doi.org/10.1098/rsta.2016.0342`

[5] S. Pagel, A. Sharma, L. Cronin, Mapping Evolution of Molecules Across Biochemistry with Assembly Theory, (2024). `https://doi.org/10.48550/ARXIV.2409.05993`

[6] A. Sharma, D. Czégel, M. Lachmann, C.P. Kempes, S.I. Walker, L. Cronin, Assembly theory explains and quantifies selection and evolution, Nature, 622, 321-328 (2023). `https://doi.org/10.1038/s41586-023-06600-9`