Supplementary Information

Fibration symmetries uncover the building blocks of biological networks

Flaviano Morone, Ian Leifer, Hernán A. Makse

Contents

III. TRANSCRIPTIONAL REGULATORY NETWORK OF *E. COLI*

To define the transcriptional regulatory network (TRN) we use the transcription factor-gene target bi-partite network of *Escherichia coli* K-12 obtained from the RegulonDB data source (`http://regulondb.ccg.unam.mx`). RegulonDB manually curates all transcriptional regulations from literature searches [11]. We download all transcriptional regulatory interactions catalogued in RegulonDB version 9.0 from `http://regulondb.ccg.unam.mx/menu/download/datasets/files/network_tf_gene.txt`, last accessed September 15, 2018.

The database downloaded from RegulonDB is composed of a bipartite transcription factor - gene target network. In this bi-partite dataset, a directed link between a source transcription factor (TF) and a target gene means that the TF binds to the DNA sequence at the binding site of the target gene to regulate its rate of transcription. In *E. coli*, each gene expresses a single TF (this is not the case in eukaryotic genes that contains introns and splicing of protein-coding RNA can produce many proteins from a single gene). Therefore, a gene-gene regulatory network can be constructed from the bipartite transcription factor-gene target network by associating each TF to the gene that expresses the TF. Then, a directed link in the TRN from gene $i \rightarrow$ gene j implies that gene i encodes for a TF that controls the rate of transcription of gene j. Thus, a directed link encodes the combined processes of transcription, translation and TF binding to a target gene. We denote genes in bacteria in italics, e.g., *gadX* and its protein as GadX. Thus, we say that gene i sends a genetic 'message' to gene j and the 'messenger' is the TF. The history of all messages passing in the network defines the information flow in the network. A TF can either be an activator, repressor or can have a dual function. For the purpose of calculating isomorphisms between input trees, the dual interactions are treated as distinct interactions. Thus, these three interactions are treated as three different types.

For the purpose of building the TRN it is important to distinguish the gene's products between genes encoding for TFs and the rest of the genes encoding for the rest of the proteins (enzymes, kinases, transport proteins, etc). A TF is a regulatory protein that regulates a gene by binding, and therefore will always have an out-going link in the network. There are other regulatory proteins (like kinases, histones, coactivators, etc) that regulate gene expression but they do not have a DNA-binding domain and they regulate gene expression without binding. In our TRN, genes that encode for a protein that is not a TF do not have out-going links in the network. They only have in-going links and therefore are dangling ends in the network. In *E. coli* most of these proteins are enzymes that catalyze biochemical reactions in the metabolic network. Other proteins are involved in transport and signaling processes (kinase) in the cell.

TF are also activated by effector molecules (metabolites) that bind non-covalently to an allosteric site of the TF to alter the conformation of the TF to activate it or deactivated by controlling the binding/unbinding of the TF to DNA. Effectors can also produce covalent activation of the TF like for instance during phosphorylation mediated by kinases in the two component TFs.

We treat these effector activities as external parameters, determined by the growth conditions in the surrounding system (the cell in its changing environment) or by the metabolic network, which is considered external to the TRN. These external perturbations are considered as the external growth conditions when we analyze the co-expression profiles in Section II. In the present study, the metabolic network is considered external to the TRN, so we do not consider feedback loops from the TRN to the metabolic network and back to the TRN mediated by effector metabolites. This extended network is treated in a follow up.

In *E. coli*, genes are also grouped by operons. An operon is a set of contiguous genes that are transcribed as a single unit from the same mRNA molecule and the same promoter site upstream of all genes and a terminator downstream [11]. An operon can contain genes encoding for TF or non-TF proteins, and more than two TFs can be part of the operon. Since the operons are transcribed by the same RNA molecule, then we group these genes into a single node in the network. This is certainly the case when the operon has a single promoter transcribing the full operon. However, there is some ambiguity in the construction of the network using the definition of operon in RegulonDB when there are promoters in the middle of the operon and these promoters transcribe more than one TF in the operon, forming different transcription units. For instance, the operon in the gad system, *gadAXW* which is important in the pH strongly connected component in Fig. 2b. This operon expressed two TFs, GadX and GadW, and one enzyme GadA. Here, each gene has its own promoter and terminator and thus are different nodes in the network. Moreover, each TF is regulated by different TFs as well as each TF regulates different genes. As seen in Fig. 2b, for instance, GadX binds to *hns* but not GadW. Also, GadW is regulated by *ydeO* but *ydeO* does not regulate *gadX*. Thus, putting together these two genes in the same operon *gadAXW* would miss all these links. Thus, when two TF with different promoters are part of the operon, we consider the TF as different genes. On the other hand, the non-TF genes in operons are always put together with other genes in the operon. For instance, the *gadAXW* operon from RegulonDB is considered as two nodes: *gadW* and *gadAX*. To simplify notation, when there is an operon that contains one TF and several non-TF proteins, then for simplicity, we call this operon by the name of the TF. For instance, *gadAX* is simply called *gadX* or the operon *rbsDACBKR* is called *rbsR* and therefore the TF *rsbR* represents the entire operon *rbsDACBKR*. Finally, when all the genes in the operon are non-TF, then we call the operon with all the genes names, as for instance, *lsrACDBFG-tam*.

In the RegulonDB database there are a total of 4690 genes. Out of these genes, Regulon DB provides a bipartite network consisting of 1843 genes with interactions from or to other genes, the remaining genes are not considered in the analysis. There are 192 genes that encode for TFs. We cluster the genes into 313 operons as explained above. Full names of operons and genes appear in SI Table VI. After grouping the genes into operons, the network is reduced to 879 nodes. There are 1835 directed edges with an average in-degree (or out-degree) of 2.1. In this network we find 91 different fibers that encompass 416 different nodes. We find that 28 nodes are involved in 7 strongly connected components of size larger than one node, and the rest are single node connected components.

IV. SYMMETRY FIBRATIONS

Below we provide formal definitions of the main concepts using in the paper: (a) input trees and isomorphisms, (b) from fibrations → surjective minimal graph fibrations called here symmetry fibrations, (c) fibers and minimal bases, and (d) minimal balance coloring algorithm. We start with a review of the literature (not exhaustive).

The literature on fibrations and groupoids crosses the fields of mathematics, computer science and dynamical systems theory. The notion of fibration was first introduced by Grothendieck as fibrations between categories in algebraic geometry [12]. The original paper of Grothendieck has been published as a part of the Séminaire N. Bourbaki in 1958 and can be found at `http://www.numdam.org/article/SB_1958-1960__5__299_0.pdf`. A mathematical account of Grothendieck fibrations in the context of category theory appears in `https://ncatlab.org/nlab/show/Grothendieck+fibration`. For a review of

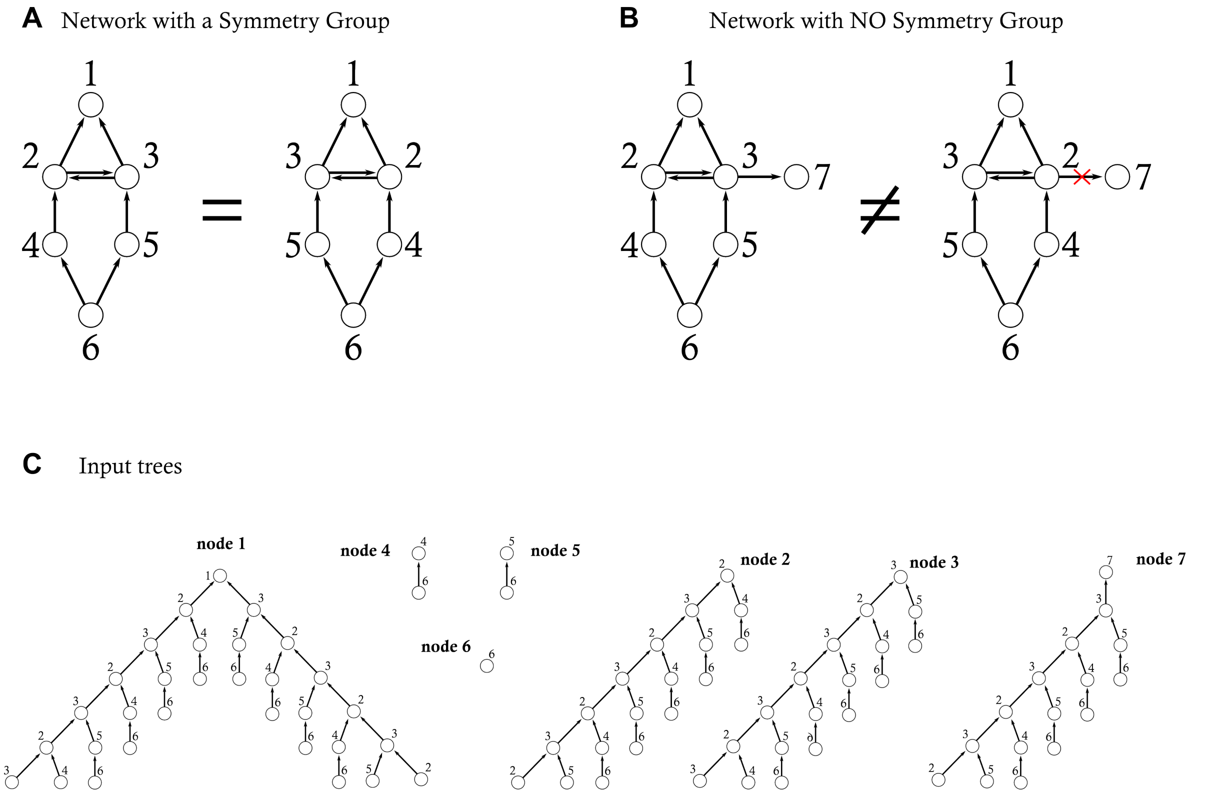

A Network with a Symmetry Group
B Network with NO Symmetry Group
C Input trees
node 1
node 2
node 3
node 4
node 5
node 6
node 7

FIG. 5: **Group symmetries and fibrations with their input tree**. **a,** Example of a network with a symmetry group. The automorphism shown maps the network into another network leaving invariant the connectivity of every nodes in the network [4, 14, 17, 18]. **b,** A network without automorphisms but with a fibration. The addition of a single out-link from 3 $\rightarrow$ 7 breaks the whole group symmetry. However, since fibrations are defined according only to the input tree, then the network still have a symmetry, a fibration arising from the fact that the input trees of nodes 2 and 3 are isomorphic, as well as between the input trees of nodes 4 and 5 as shown in **(c).** There are no more isomorphisms as shown by the rest of the input trees. Therefore, nodes 2 and 3 form a fiber. Nodes 4 and 5 also form another fiber, yet independently of the other fiber. The fibration is a morphism that maps the network into a base which is formed by collapsing the isomorphic nodes into one, i.e., collapsing node 2 and 3 together, and node 4 and 5 together. The resulting base is also called a quotient graph.

the history of fibrations from Grothendieck to modern studies, see the blog of Vigna at `http://vigna.di.unimi.it/fibrations/`. The formulation of Grothendieck is highly abstract and differs from our present work which refers to the notion of surjective minimal graph fibration which is a fibration between graphs. The work of Boldi & Vigna

[13] and DeVille & Lerman [15] on graph fibrations are the closest to our formulation, see `http://vigna.di.unimi.it/ftp/papers/FibrationsOfGraphs.pdf`. Graph fibrations have been applied in computer science to understand PageRank [35], and the state of synchrony of processors in computing distributed systems [36, 37], where fibrations are the key concept in the computation of identical states in distributed system. The relation between surjective minimal graph fibrations and synchronous subspaces is elaborated in DeVille & Lerman [15] and Nijholt, Rink & Sanders [16]. It should be noted that all these works on fibrations pertain to a highly abstract mathematical level which, in turn, provides the concept of fibration with a quite broad applicability. For a more accessible reading on fibrations within the particular context application to biological networks, the reader is recommended to follow our paper and supplementary sections.

In parallel, the work of Golubitsky and Stewart [14, 20] and others in dynamical systems theory consider the equivalent formalism of symmetry groupoids, equitable partition of balanced colored nodes and its relation with synchronization [21–23]. A review of the groupoid formalism and its application to synchronization in dynamical systems appears in [14]. DeVille and Lerman [15] also discuss the relation between graph fibrations and the groupoid formalism.

Synchronization arises also as a consequence of permutation symmetries in the network, called automorphisms [4], which form symmetry groups and are different from symmetry fibrations and symmetry groupoids. There is a large literature in the dynamical system community dealing with cluster synchronization from automorphisms, since synchronization is an ubiquitous phenomenon across all sciences [21–23]. Reviews can be found in the work of Golubitsky and Stewart [14, 20] to recent work in [17–19] and references therein. Symmetry groups are the cornerstone of physical phenomena appearing in all physical systems [5].

Below, to elaborate on the definition of symmetry fibrations, we first compare fibrations to automorphisms which form symmetry groups [4, 14, 17–19] using the example networks of Figs. 5a and 5b. An automorphism is a transformation that preserves the full connectivity of the network. That is, an automorphism preserves not only the inputs but also the outputs of each node in the network, and therefore, it presents more stringent conditions on the connectivity than symmetry fibrations which preserve only the input trees. For example, the network of Fig. 5a is invariant under the automorphism defined by the permutation:

$$\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 1 & 3 & 2 & 5 & 4 & 6 \end{pmatrix} , \qquad (6)$$

because the nodes are connected exactly to the same nodes before and after the application of the permutation σ, which is a global mirror symmetry.

Next, consider the slightly modified network depicted in Fig. 5b left, which differs from the network in Fig. 5a by one extra out-going link from node 3 to 7. In this network, the permutation of nodes $2 \leftrightarrow 3$ and $4 \leftrightarrow 5$, Eq. (6), is not an automorphism anymore, because it does not preserve the in and out connectivities of all nodes, e.g., node 3 is connected with 7 but loses this connection after the permutation (Fig. 5b right). It is interesting to see how fragile group symmetries are: if we connect just one extra node to the network as shown in Fig. 5b, the symmetry (i.e. the network automorphism group) is broken. This occurs because automorphisms require very strict arrangements of nodes and links to preserve, rigidly, the global structure of the network. Fibration symmetries, with their emphasis in the preservation of the input trees only, is less restrictive. This might explain why fibration symmetries emerged in living systems as opposed to the more restrictive automorphisms which describe all aspects of matter, from elementary particles to atoms, molecules and phases of matter.

This example raises the following question: are there extra symmetries in the network shown in Fig. 5b beyond its automorphisms? The answer to this question is, indeed, yes: there are extra symmetries in the network of Fig. 5b, the fibration symmetries [12, 13], which do not form a group [4] but groupoids [14]. A groupoid is a set of transformations satisfying the axioms of invertibility, identity and associativity but not the composition law (closure) [14], while in a group, transformations satisfy the four axioms. For this reason, groupoids are fundamentally different algebraic structures compared with traditional group symmetries.

A. Input tree

Roughly speaking, symmetry fibrations take into account only the input trees of the nodes, but not the output-trees (this is not true though when the input and output trees are connected). Thus, node 3 in Fig. 5b is connected to node 7 via an out-going link, and this link destroys the symmetry group, but node 3 is still symmetric with 2 via a symmetry fibration, since the input trees of nodes 2 and 3 are isomorphic, even though node 3 is connected with 7. This is because the connection $3 \rightarrow 7$ is an out-going link of node 3 and, therefore, is not part of its input tree. Simply put, symmetry fibrations preserve input trees only, while automorphisms preserve both input and output-trees, since they preserve the full connectivity of the network, and thus, they represent more stringent symmetries than fibrations. We formalize this idea next after introducing some definitions.

The basic ingredient to define a new symmetry beyond automorphisms is the **input tree**, which contains the full information received by a given node through the totality of all the possible paths ending in that node and starting from every other node in the network. Thus, for every node i in the network G there is a corresponding input tree, called T_i, which is defined as a tree with a selected node r_i, called the root, and such that every other node is a path $\mathcal{P}_{j\to i}$ of G starting from j and ending in i [16]. A link from node $\mathcal{P}_{j\to i}$ to node $\mathcal{P}_{k\to i}$ exists if $\mathcal{P}_{j\to i} = e_{j\to k}\mathcal{P}_{k\to i} =$, where $e_{j\to k}$ is an edge of G.

The concept of input tree has appeared in the literature as the universal total space in traditional categorical or topological terminology [12], the universal total graph from [13], the view in the theory of distributed systems, or the unfolding of a nondeterministic automaton in concurrency theory [13].

For example, let us construct the input tree T_2 of node 2 in the network on the left of Fig. 5b. The root is the node r_2 at the uppermost level of the tree. Every other node of the input tree of node 2 is a path $\mathcal{P}_{j\to 2}$ ending in 2. There are two paths of length 1: $\mathcal{P}^{(1)}_{3\to 2}$ and $\mathcal{P}^{(1)}_{4\to 2}$; three paths of length 2: $\mathcal{P}^{(2)}_{2\to 2}$, $\mathcal{P}^{(2)}_{5\to 2}$, and $\mathcal{P}^{(2)}_{6\to 2}$; and so on. Since $\mathcal{P}^{(2)}_{2\to 2} = e_{2\to 3}\mathcal{P}^{(1)}_{3\to 2}$, we put a link in the input tree from $\mathcal{P}^{(2)}_{2\to 2}$ to $\mathcal{P}^{(1)}_{3\to 2}$ because $\mathcal{P}^{(2)}_{2\to 2} = e_{2\to 3}\mathcal{P}^{(1)}_{3\to 2}$. We then add all other links in the input tree using the same criterion. The resulting input tree T_2 is shown in Fig. 5c, together with the input trees of all other nodes in the network in Fig. 5b.

To simplify, we label each node of T_i using the starting point of the corresponding path $\mathcal{P}_{j\to i}$. For example, in T_2 nodes $\mathcal{P}^{(1)}_{3\to 2}$ and $\mathcal{P}^{(1)}_{4\to 2}$ are labeled 3 and 4 respectively, and the length of the path is equal to the depth of the node in the input tree.

Thus, in practice, we arrive at the following way to construct the input tree: we start with the node at the root, lets say node 2. We label every node $\mathcal{P}_{j\to 2}$ in the input tree by node j where the path starts. The first layer of the input tree consists of all the nodes that are at a distance one from the root. In this case, nodes 3 and 4. Thus we add two links to 2 from 3 and 4 in the input tree.

The second layer of the input tree is obtained applying the same procedure to each node in the first layer, 3 and 4. For instance, node 3 receives a link from 2 and 5. Therefore the second layer of the input tree contains nodes 2 and 5 connected to node 3. We repeat the procedure with the other node in layer 2: node 4. Node 4 receives a link only from node 6, and node 6 from no one. So, we add a link from 6 to 4 and this path does not propagate further. The third layer of the input tree is obtained iteratively applying the same procedure, and so on.

We note that the input trees of nodes 1, 2, 3 and 7 are infinite since the network contains a cycle (or loop) between nodes $2 \rightleftarrows 3$. For instance, T_1 is infinite because there are paths crossing the loop infinite times. On the other hand, the input trees of nodes 4, 5 and 6 are finite since they do not cross the loop.

B. Isomorphic input trees

The input tree T_i at node i can be interpreted as the collection of all possible 'histories' starting at some node and ending in node i. As shown in Section I C, if two input trees T_i and T_j are isomorphic, then the corresponding nodes i and j in network G have the same dynamical state [15, 16]. This equivalence is understood in terms of a local in-isomorphism that maps nodes to nodes and links to links, so it formalizes the fact that the dynamical interactions represented by a directed link from gene to gene could be in principle different across genes, as long as the links are the same (or similar, in case that the produced synchronization is approximate) inside the fiber.

An isomorphism between T_i and T_j is defined as a bijective map $\tau : T_i \to T_j$, which maps one-to-one the nodes and edges of T_i to nodes and edges of T_j.

A minimal condition for the existence of an isomorphism between the input trees is that the two input trees have the same number of nodes (we could also add a condition of the same degree sequence). Thus, it is clear that there could be no isomorphism between the input trees of nodes 2 and 4, since the former contains an infinite number of nodes and the later just two. Thus, a minimal condition for an isomorphism to exist is that it should be a mapping between two input trees with the same number of nodes, since the mapping needs to be bijective, i.e., with an inverse. By inspection it is then clear that there is an isomorphism between the input trees of nodes 4 and 5. This isomorphism is the map $\tau_{4\to5} : T_4 \to T_5$, and it is written as a transformation following the notation:

$$\tau_{4\to5} = \begin{pmatrix} 4 : & 6 \\ \downarrow & \downarrow \\ 5 : & 6 \end{pmatrix}, \qquad \text{(isomorphism between input trees of nodes 4 and 5).} \tag{7}$$

which maps the root of T_4 to the root of T_5 as $\tau_{4\to5}(4) = 5$, and node $6 \in T_4$ to node $6 \in T_5$ as $\tau_{4\to5}(6) = 6$. The notation starts with the root of the tree and then we write nodes in each level from top to bottom starting from left to right in each level. In this particular example the links are of the same type, so there is no need to specify the mapping between links in the isomorphism, but in general the local equivalence require that nodes are map to nodes and also links are mapped to the same type of link by the isomorphism.

The map in Eq. (7) is one of the simplest isomorphism since the input tree contains only one level. In this particular case, to see that nodes T_4 and T_5 are isomorphic, it is thus enough to see that both nodes 4 and 5 connect to one and the same node, which is node 6 in this case. That is, both input trees of nodes 4 and 5 are isomorphic because they are made up of just two nodes and one edge, and this isomorphism implies that 4 and 5 receive the same information. This is the simplest form of an isomorphism between input trees. In this case, we say that node 4 and 5 have the same *input-set*, which is an input tree of only one level, that is the set of incoming links. The input-set is used in the groupoid formalism in Ref. [14].

Next, we consider the input trees of nodes 2 and 3. By visual inspection, both input trees have the same 'shape'. However, these trees are infinite in the number of levels. How do we decide if two input trees are isomorphic when they have an infinite number of levels? Remarkably, to determine if two input trees are isomorphic, it suffices to check that they are isomorphic up to the $N - 1$ level, thanks to a theorem by Norris [26], where N is the total number of nodes in the network G. This is an important result that allows us to avoid to check an infinite number of equivalences. Since G has $|N_G| = 7$, we use six levels in the input trees to determine that there is an isomorphism between T_2 and T_3 which corresponds to the following map:

$$\tau_{2\to 3} = \begin{pmatrix} 2: & 3 & 4 & 2 & 5 & 6 & 3 & 4 & 6 & \dots \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ 3: & 2 & 5 & 3 & 4 & 6 & 2 & 5 & 6 & \dots \end{pmatrix}, \quad \text{(isomorphism between input trees of 2 and 3).} \tag{8}$$

There are no other isomorphism between the other input trees. Notice that T_7 is not isomorphic to T_3 and T_2 by just one link to the root.

The existence of an isomorphism τ from the input tree of node i to the input tree of node j implies the synchronization of x_i and x_j [15]. In the groupoid formalism of Golubitsky and Stewart, it is said that two nodes are synchronized if their input-set are synchronized, too [14]. Analogous work in dynamical systems shows that automorphisms in networks lead to synchronized nodes in orbits, see [17–20] and references therein. The orbit of a given node is obtained by applying all automorphisms of a network to the node and the nodes in the orbit are synchronous. The synchronized orbits obtained from automorphisms are analogous to the synchronized fibers obtained from symmetry fibrations. In general, every orbit is also a fiber, but the opposite is not true, since a fiber is not necessarily an orbit.

In our analysis of the *E. coli* network, we find some automorphisms. Some of the star fibers with $n = 0$ are also orbits of the networks since they are invariant under permutation symmetries of the symmetric group of order n, S_n. But this is only when the genes in the star have no out-going links. As shown in the example of Fig. 5, an out-going link in any of the star genes, will destroy the automorphism, but not the fiber. For this reason, automorphisms are somehow more prevalent in undirected networks. For instance, we have found that automorphisms describe the symmetries of the gap junction connectome of *C. elegans*, which is composed all of undirected links [34]. In the case of directed biological networks treated here, while automorphisms could be of use to discover some synchronized nodes, the majority of synchronization is due to symmetry fibrations, which are not described by automorphisms.

C. From fibrations to symmetry fibrations via isomorphic input trees and minimal bases

A fibration is any morphism from a network $G = (N_G, E_G)$ to a base $G = (N_G, E_G)$: $\psi : G \to B$ [12]. If a network $G = (N_G, E_G)$ has at least one pair of isomorphic input trees, then there exists a network $B = (N_B, E_B)$, called the **base** of G, such that G can be 'fibered' over B by the graph fibration. The base B is defined as follows:

- a node $I \in N_B$ is a representative of the set of nodes $\{i \in N_G\}$ whose input trees are isomorphic;
- an edge $e_{I\to J}$ where $I, J \in E_B$ is defined as $e_{I\to J} = \sum_{i\in I} e_{i\to j}$, where $e_{i\to j} \in E_G$.

Having defined the base network B, we say that G is fibered over B if there exists a surjective morphism $\psi : G \to B$, called surjective graph fibration [13], that maps nodes and edges of G to nodes and edges of B as: $\psi(i) = I$ for all $i \in N_G$, and $\psi(e_{i\to j}) = e_{I\to J}$. A surjective morphism is a map between two sets (the domain and codomain) where each element of the codomain (in this case B) is mapped to, at least, by one element of the domain (in this case G). The set of nodes $i \in N_G$ that are mapped to the same node $I \in N_B$, and denoted by $\psi^{-1}(I)$, is called the fiber of G over node I. We notice that all input trees of nodes which belong to the same fiber are pairwise isomorphic.

In general a surjective graph fibration ψ can map nodes with isomorphic input trees to different bases, thus, the number of fibers is not minimal.

A surjective graph fibration that maps all genes with isomorphic input trees to a single common node in B is called a surjective minimal graph fibration in the sense of [13]. Such a minimal fibration will generate then the minimal bases of the network and will produce the largest collapse of nodes in fibers. In this work we only deal with surjective minimal graph fibrations and we call them symmetry fibrations for short.

In practice, a symmetry fibration maps G to the minimal base B (analogous to the quotient), that consists of the following steps: *(i)* consider all the nodes in a fiber (which have isomorphic input trees) and choose one as the representative I, *(ii)* collapse the nodes in the fiber into one single node in B and call it by the name of the representative node I, *(iii)* for every link of a node j in G directed to the node I in G, add a link in B from j to I. If the node j belongs to the fiber, then the corresponding link in B is an autoregulation loop in B, *(iv)* repeat for every fiber in G. When fibers belong to disjoint components of the network, then they are considered as distinct fibers.

V. ALGORITHM TO FIND FIBERS WITH MINIMAL BALANCE COLORING

The algorithm to partition the network into fibers is based on the 'minimal balanced coloring' algorithm developed by Cardon & Crochemore in Ref. [24]. Here we follow a version developed by Kamei & Cock [25] to construct a *minimal* balanced coloring of a network, namely a coloring that employs the least possible number of colors, which is associated with minimal graph fibrations. The algorithm's runtime scales as $O(|E_G| \log_2 |N_G|)$, which implies that it is essentially linear with the network size, specially for sparse networks, and can be applied to very large networks.

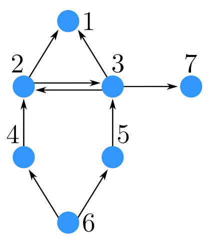
The theory of balance coloring is explained in Ref. [14]. A balance coloring creates a partition of nodes of G into disjoint sets (corresponding to synchronous fibers) such that each node in one set receives the same number of colors from nodes within other sets [14, 20]. A coloring of G with this property is the *balanced coloring* and represents an *equitable partition* of the network, see [14, 20]. The sets identified by a *minimal balanced coloring* partitions the network with minimal colors and corresponds to the fibers of G identified by minimal graph fibrations ψ [13–15].

Thus, we color nodes such that synchronous nodes in a fiber receive the same colors from their synchronous nodes. As example, the genes *baeR* and *spy* (Fig. 1a) have the same color and are in the same fiber since they receive the same colors from their neighbors: both *baeR* and *spy* receive one red color via the activator link from one red node (*baeR* from itself and *spy* from *baeR*) and one green activator link each from the green node *cpxR*.

The algorithm constructs a coloring of the nodes that is balanced. A coloring is balanced if two identically colored nodes are connected to identically colored nodes via their inbound links. Each balanced colored cluster is a fiber in the network. The fibers also corresponds to the orbits in a network when the symmetries are automorphisms rather than isomorphisms in the input trees. The flow of the algorithm is exemplified with the example network of Fig. 6.

- **Step 1** - We start by assigning the same color to all nodes. In Fig. 6a all nodes are initially colored in blue. In addition, we assign to each link the same color of the

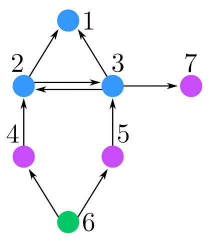
a



STEP 1

	1 🔵	2 🔵	3 🔵	4 🔵	5 🔵	6 🔵	7 🔵
🔵→	2	2	2	1	1		1

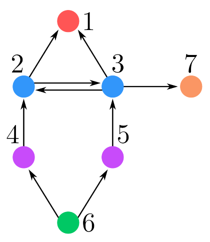
b



STEP 2

	1 🔵	2 🔵	3 🔵	4 🟣	5 🟣	6 🟢	7 🟣
🔵→	2	1	1				1
🟣→		1	1				
🟢→				1	1		

c



STEP 3

	1 🔴	2 🔵	3 🔵	4 🟣	5 🟣	6 🟢	7 🟠
🔵→	2	1	1				1
🟣→		1	1				
🟢→				1	1		
🔴→							
🟠→							

FIG. 6: **Algorithm to find the fibers of a network through a minimal balanced coloring.** The goal of the algorithm is to find a minimal balanced coloring of the network, so that two nodes have the same color only if they are connected to the same number of identically colored nodes via inbound links. The colors represent the fibers in the network.

node from where it emanates. To update the coloring (or, equivalently, to generate a new partition) of nodes, we construct the table shown in the right panel of Fig. 6a, as explained next. In the top row of this table we put the network nodes colored with their current color. In the leftmost column we put each type of colored link. In this initial stage of the algorithm we only have a blue link for all the nodes. Then, we fill the entries of the table with the number of colored links of this blue type that are received by the corresponding node. For example, node 1 receives two 2 blue links as well as nodes 2 and 3. Nodes 4, 5 and 7 receive one blue link each, and node 6 nothing. The structure of this table determines the new coloring as explained in the next step.

- **Step 2** - Using the table in Fig. 6a we update the coloring of nodes as follows. We assign the same color to all nodes that receive the same number of colored links of each type. Specifically, nodes 1, 2 and 3 receive two blue links, so we assign them the same (blue) color. Analogously, nodes 4, 5 and 7 receive one blue link, so we assign them the same color, but different from blue. We assign them a purple color. Similarly, we assign another color to node 6 (green). We then obtain the colored network in the left of Fig. 6b. Applying the counting of receiving coloring links to this network, we obtain the new coloring table shown in Fig. 6b, where each link has the color of the node from where it emanates. Thus, we update the table to generate the new coloring, as shown in the right panel of Fig. 6b.

- **Step 3** - Using the same criterion as in Step 2, we update the coloring of nodes, comprising now five different colors, and then we generate the new table, as shown in Fig. 6c. At this point the algorithm stops, because we do not need to introduce more colors, since each color is balanced. Each color corresponds to a fiber, and each node in each colored fiber receives the same colors from other fibers or from nodes in the same fiber. Therefore, the coloring shown in the network of Fig. 6c is the minimal balanced coloring of the network, and the colors indicate the fibers in the network.

As far as only minimal fibrations are considered, the algorithm will return always the same fibers containing the same nodes, for any initial condition and realization. Below we provide the pseudo-code to clarify the algorithm. More detailed instructions and methodology for obtaining fiber building blocks will be given in a follow-up paper. We start by assigning all nodes to the same fiber and then continue to refine the partition basing on the input set of the node until no further refinement can be obtained.

Algorithm 1 Finding fibers following Kamei & Cock Ref. [25]

Input: Graph $G = \{N_G, E_G\}$, where N_G are vertices and E_G are edges of the analyzed network

$\mid N_G \mid$ - number of vertices, $N_G = \{v_1 \dots v_{|N_G|}\}$

Output: $C = \{c_i\}$, where c_i - color of node i and $i = 1 \cdots \mid V \mid$

Notation: $I_i = \{I_i^1 \dots I_i^N\}$, where N = current number of colors

```
1:  $N_0 = 1$
2:  for $i = 1 \cdots \mid N_G \mid$ do
3:      $c_i = 1$
4:  end for
5:  $j = 0$
6:  repeat
7:      for $i = 1 \cdots \mid N_G \mid$, $k = 1 ... N_j$ do
8:          $I_i^k$ = number of nodes of color $k$ in the input set of $v_i$
9:      end for
10:     $H$ = set of all unique $\{I_i\}$
11:     // assign each unique vector a color and color the graph accordingly
12:     for $i = 1 \cdots \mid N_G \mid$ do
13:         $c_i$ = index of $I_i$ in $H$, e.g. if two nodes have the same $I_i$ and $I_j \rightarrow c_i = c_j$
14:     end for
15:     $j = j + 1$
16:     $N_j = \mid H \mid$
17: until $N_j \neq N_{j-1}$
18: return $\{c_i\}$
```

VI. STRONGLY CONNECTED COMPONENT

In a directed network, the strongly connected component is composed of nodes that are reachable from every other node in the component. That is, there is a directed path from every node to any other node in the strongly connected component. A weakly connected component is obtained when we ignore the directionality of the links. Strongly connected components are relevant to genetic fibers since they contain loops that control the state of the genes. We find four types of strongly connected components. Single-gene components composed of autoregulator loops like *cpxR* and *fadR* in Figs. 1a and 1e. The other type of components are those in Fig. 2a and Fig. 2b and also a five-gene connected component shown in SI Fig. 7. We note that most of the fibers regulated by these components do not belong to the connected component. This is because they receive information but do not send information back to the connected component. These fibers are characterized by integer fiber numbers. When the fiber receives and sends back information, that is, when the fiber belongs to the strongly connected component, then it becomes a Fibonacci fiber. The largest strongly connected component in the *E. coli* network controls the pH system shown in Fig. 2b.

VII. STATISTICS OF FIBERS IN THE TRN OF *E. COLI*

A. Fibers statistics in *E. coli*

SI Table I shows the counts in the *E. coli* network of each building block. For instance the most abundant building blocks are the following:

$|n = 0, \ell = 1\rangle$: 45
$|n = 1, \ell = 0\rangle$: 13
$|n = 0, \ell = 2\rangle$: 13
$|n = 1, \ell = 1\rangle$: 8

The list is completed with the fractal building blocks of Fibonacci sequences which are less numerous but more complex in their structure:
$|\varphi_2 = 1.6180.., \ell = 2\rangle$: 1

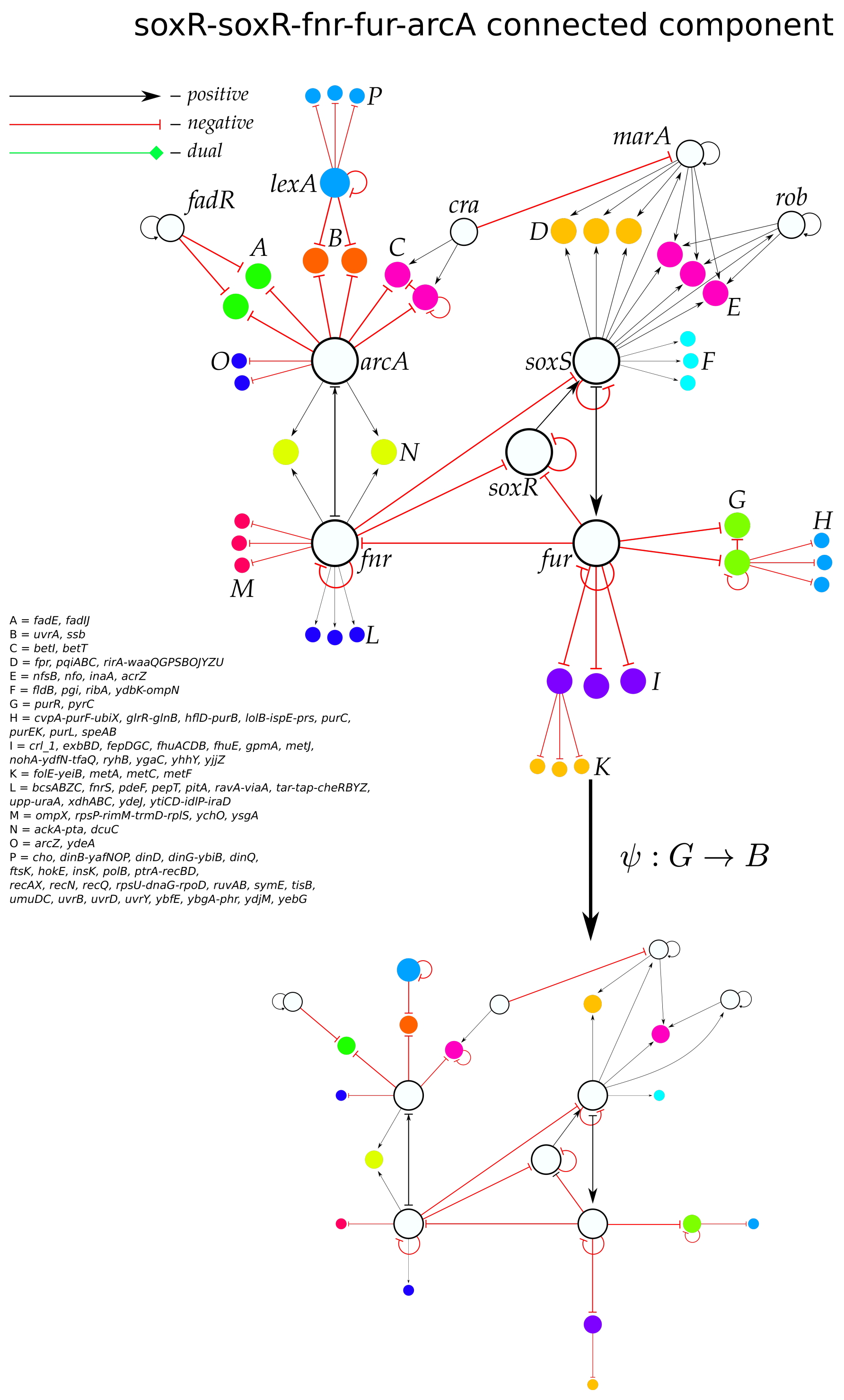

soxR-soxR-fnr-fur-arcA connected component
– positive
– negative
– dual
P
lexA
marA
fadR
cra
rob
A
B
C
D
E
O
arcA
soxS
F
N
soxR
G
H
M
fnr
fur
L
I
K
A = fadE, fadIJ
B = uvrA, ssb
C = betI, betT
D = fpr, pqiABC, rirA-waaQGPSBOJYZU
E = nfsB, nfo, inaA, acrZ
F = fldB, pgi, ribA, ydbK-ompN
G = purR, pyrC
H = cvpA-purF-ubiX, glrR-glnB, hflD-purB, lolB-ispE-prs, purC, purEK, purL, speAB
I = crl_1, exbBD, fepDGC, fhuACDB, fhuE, gpmA, metJ, nohA-ydfN-tfaQ, ryhB, ygaC, yhhY, yjjZ
K = folE-yeiB, metA, metC, metF
L = bcsABZC, fnrS, pdeF, pepT, pitA, ravA-viaA, tar-tap-cheRBYZ, upp-uraA, xdhABC, ydeJ, ytiCD-idlP-iraD
M = ompX, rpsP-rimM-trmD-rplS, ychO, ysgA
N = ackA-pta, dcuC
O = arcZ, ydeA
P = cho, dinB-yafNOP, dinD, dinG-ybiB, dinQ, ftsK, hokE, insK, polB, ptrA-recBD, recAX, recN, recQ, rpsU-dnaG-rpoD, ruvAB, symE, tisB, umuDC, uvrB, uvrD, uvrY, ybfE, ybgA-phr, ydjM, yebG
$\psi : G \to B$

FIG. 7: A five-gene connected component of *soxR*, *soxS*, *fnr*, *fur*, and *arcA* with its regulated fibers.

$|\varphi_3 = 1.4655.., \ell = 1\rangle$: 1

$|\varphi_4 = 1.3802..., \ell = 1\rangle$: 1

Structure type	Amount in E-coli
$\lvert n = 0, l = 1\rangle$	45
$\lvert n = 0, l = 2\rangle$	13
$\lvert n = 0, l = 3\rangle$	3
$\lvert n = 1, l = 0\rangle$	13
$\lvert n = 1, l = 1\rangle$	8
$\lvert n = 1, l = 2\rangle$	3
$\lvert n = 2, l = 0\rangle$	1
$\lvert n = 2, l = 1\rangle$	1
$\lvert \varphi_d = 1.3802.., l = 1\rangle$	1
$\lvert \varphi_d = 1.4655.., l = 1\rangle$	1
$\lvert \varphi_d = 1.6180.., l = 2\rangle$	1
Composite Fiber	1
Total number of building blocks	91

TABLE I: Building block statistics. We show the count of every building block defined by the fiber numbers.

B. Full list of fibers in *E. coli*

SI Table VI shows the complete list of the 91 fibers building blocks found in the genetic network of *E. coli*. We list the genes in the fiber plus their external regulators. If a gene or operon is not in this list, for instance *lacZYA*, it means that the gene or operon is not in a fiber. Supplementary File 1 shows the plot of the circuit of every fiber and the fiber building block.

The first column in SI Table VI is the ID of the fiber. This ID refers to the plot of the fiber building block in Supplementary File 1. The second column lists the genes in the fiber, the third column lists the external regulators. The last column specifies the fiber number associated with each fiber as $|n, \ell\rangle$ or $|\varphi_d, \ell\rangle$.

VIII. DATASETS OF BIOLOGICAL AND NON-BIOLOGICAL NETWORKS

To investigate the applicability of fibrations in a broader context, we performed an extensive analysis of different complex networks from diverse domains in systems science.

Full details of each network analyzed can be accessed at `https://docs.google.com/spreadsheets/d/1-RG5vR_EGNPqQcnJU8q3ky1OpWi3OjTh5Uo-XaOPjOc`. The codes to reproduce this analysis are at `github.com/makselab` and the full datasets appear at `kcorelab.org`. See also tables below with information about the networks.

We first show the symmetry fibrations in biological networks and species. See Section I H. We characterize biological networks spanning from:

- **Biological networks: transcriptional regulatory networks, metabolic networks, cellular processes networks and pathways, disease networks, neural networks.**

We study the following species:

- **Species: A. thaliana, E. coli, B. subtilis, S. enterica (salmonella), M. tuberculosis, D. melanogaster, S. cerevisiae (yeast), M. musculus (mouse), and H. sapiens (human).**

We then study non-biological networks in Section I H:

- **Social Networks: online social networks, Facebook, Twitter, Wikipedia, Youtube, email networks, communication networks, citation networks, collaboration networks, bloggers**
- **Internet: routers, autonomous systems, web graphs, hyperlinks, peer-to-peer**
- **Infrastructure Networks: power grid, airport, roads, flights**
- **Economic Networks**
- **Software Networks: Linux, jdk**
- **Ecosystems**

Network Domain	Total No. of nodes	Total No. of edges	No. of networks
Biological	287390	4211856	289
Economic	1752	108639	5
Ecosystems	1879	5378	14
Infrastructure	24511	82534	16
Internet	244634	835565	27
Social	104909	1261009	15
Software	43391	503645	3

TABLE II: Features of the networks across domains. We report the total numbers for each domain summed over all the networks in the domain.

Species	Total No. of nodes	Total No. of edges	No. networks
Yeast	55932	1392926	11
Arabidopsis Thaliana	790	1431	1
Bacillus subtilis	5602	11417	3
Drosophila	39549	321734	5
Escherichia coli	879	1835	1
Human	72587	1198712	248
Micobacterium Tuberculosis	1624	3212	1
Mouse	64709	987424	7
Salmonella	8293	15589	6

TABLE III: Number of networks per species.

	Arabidopsis Thaliana	Bacillus subtilis	Caenorhabditis elegans	Cat	Drosophila	Escherichia coli	Human	Micobacterium Tuberculosis	Mouse	Rat	Salmonella	Yeast
TF	1	2	2	0	4	1	4	1	4	0	2	11
Neuron	0	0	0	1	1	0	0	0	3	3	0	0
Metabolic	0	0	0	0	0	0	48	0	0	0	2	0
Disease	0	0	0	0	0	0	66	0	0	0	0	0
Kinase	0	0	0	0	0	0	2	0	0	0	0	0
Pathway	0	0	0	0	0	0	127	0	0	0	0	0
Protein	0	1	0	0	0	0	1	0	0	0	2	0

TABLE IV: Table with the count of networks per type of biological network and species. These networks are used to calculate the distributions of fiber across species and biological types in Figs. 4a, b, and c. For each type of biological network in Fig. 4a, b, we calculate the count over the total number of networks as indicates at the end of each row for each biological type. The same occurs with the number of networks at the end of each column for each species. Figure 4c shows the counts over all the network shown in the last row/column.

Network Subdomain	Total No. of nodes	Total No. of edges	No. of networks
Autonomous systems graphs	141842	481415	14
Bitcoin	9664	59777	2
Collaboration networks	50260	504897	4
Disease	4309	15254	66
Facebook	4039	88234	1
Youtube subscriptions	13723	76765	1
Internet peer-to-peer networks	31978	110154	4
Jazz	198	5484	1
Linux	30837	213954	1
Metabolic	4273	33829	50
Networks with ground-truth communities	1005	25571	1
Neural networks	3694	129812	8
Cellular processes and Pathways	9825	54712	127
Plant-Pollinator	1631	2719	11
Plant-Seed-Disperser	65	165	2
Power grid	4941	6594	1
Sentiment	99	278	2
Transcriptional regulatory	260258	3908769	32

TABLE V: Subtypes of networks belonging to the different domains.

Id	Fiber	Regulators	Fiber Number
1	aaeR, ampDE, azuC, comR, cyaA, narQ, sohB, speC, spf, trxA, yaeP-rof, yaeQ-arfB-nlpE, yjeF-tsaE-amiB-mutL-miaA-hfq-hflXKC	crp	$\lvert n=0, l=1\rangle$
2	aaeXAB, agp, cpdB, cstA, glgS, glpR, grpE, hofMNOP, ivbL-ilvBN-uhpABC, lacI, mcaS, mhpR, nadC, ompA, ppdD-hofBC, preTA, raiA, rmf, rpsF-priB-rpsR-rplI, sfsA-dksA-gluQ, sxy, ubiG, ychH, yeiP, yeiW, yfiP-patZ, yibN-grxC-secB-gpsA, ykgR	crp	$\lvert n=0, l=1\rangle$
3	accA, accD, fabI, fadR, yceD-rpmF-plsX-fabHDG-acpP-fabF		$\lvert n=1, l=0\rangle$
4	accB, iclR	fadR	$\lvert n=1, l=1\rangle$
5	ackA-pta, dcuC	arcA, fnr	$\lvert n=0, l=2\rangle$
6	acrZ, inaA, nfo, nfsB	marA, rob, soxS	$\lvert n=0, l=3\rangle$
7	add, dsbG, gor, grxA, hemH, oxyS, trxC	crp, oxyR, rbsR	$\lvert n=0, l=1\rangle \oplus \lvert n=1, l=1\rangle$
8	adeD, adiY, chiA, gspAB, hchA, hdfR, mdtJI, rcsB, yjjP	hns	$\lvert \varphi_d=1.4655.., l=1\rangle$
9	agaR, agaS-kbaY-agaBCDI		$\lvert n=1, l=0\rangle$
10	alaA-yfbR, avtA, leuE, livJ, livKHMGF, lysU, sdaA	lrp	$\lvert n=0, l=1\rangle$
11	alaE, kbl-tdh, yojI	lrp	$\lvert n=0, l=1\rangle$
12	alaWX, argU, argW, argX-hisR-leuT-proM, aspV, flxA, glyU, leuQPV, leuX, lptD-surA-pdxA-rsmA-apaGH, lysT-valT-lysW, metT-leuW-glnUW-metU-glnVX, pheU, pheV, proK, proL, queA, serT, serX, thrU-tyrU-glyT-thrT-tufB, thrW, trmA, tyrTV-tpr, valUXY-lysV	fis	$\lvert n=0, l=1\rangle$
13	aldB, hupB	crp, fis	$\lvert n=0, l=2\rangle$
14	allA, allS, gcl-hyi-glxR-ybbW-allB-ybbY-glxK	allR	$\lvert n=0, l=1\rangle$
15	alsR, rpiB		$\lvert n=1, l=0\rangle$
16	amiA-hemF, cmk-rpsA-ihfB, uspB	IHF	$\lvert n=0, l=1\rangle$

17	amn, mipA, phnCDE_1E_2FGHIJKLMNOP, phoA-psiF, phoB, phoE, phoH, ydfH, yegH, yhjC, ytfK		$\|n=1,l=0\rangle$
18	ampC, dacC	bolA	$\|n=0,l=1\rangle$
19	araE-ygeA, araFGH	araC, crp	$\|n=0,l=2\rangle$
20	arcZ, ydeA	arcA	$\|n=0,l=1\rangle$
21	argA, argCBH, argE, argF, argI, argR, artJ, artPIQM, lysO		$\|n=1,l=0\rangle$
22	argO, lysP	argP, lrp	$\|n=0,l=2\rangle$
23	aroF-tyrA, tyrB	tyrR	$\|n=0,l=1\rangle$
24	aroH, trpLEDCBA, trpR		$\|n=1,l=0\rangle$
25	asnB, clpPX-lon, glsA-ybaT, uspE	gadX	$\|n=0,l=1\rangle$
26	aspA-dcuA, dcuR	crp, fnr, narL	$\|n=0,l=3\rangle$
27	bacA, cpxPQ, cpxR, ftnB, ldtC, ldtD, ppiD, sbmA-yaiW, slt, srkA-dsbA, xerD-dsbC-recJ-prfB-lysS, yccA, yebE, yidQ, yqaE-kbp, yqjA-mzrA		$\|n=1,l=0\rangle$
28	baeR, spy	cpxR	$\|n=1,l=1\rangle$
29	bcsABZC, fnrS, pdeF, pepT, pitA, ravA-viaA, tar-tap-cheRBYZ, upp-uraA, xdhABC, ydeJ, ytiCD-idlP-iraD	fnr	$\|n=0,l=1\rangle$
30	bdcA, dkgB, grxD, mepH, mhpT, pgpC-tadA, rfe-wzzE-wecBC-rffGHC-wecE-wzxE-rffT-wzyE-rffM, rybB, tehAB, tsgA, ydbD, yeaE	nsrR	$\|n=0,l=1\rangle$
31	betI, betT	arcA, cra	$\|n=1,l=2\rangle$
32	bioA, bioBFCD	birA	$\|n=0,l=1\rangle$
33	bluF, ydeI	rcdA	$\|n=0,l=1\rangle$
34	borD, envY-ompT, mgrB, mgrR, mgtLA, mgtS, pagP, rstA, ybjG	phoP	$\|n=0,l=1\rangle$
35	cbpAM, gltX, gyrB, msrA	fis	$\|n=0,l=1\rangle$
36	cdaR, garD, gudPXD		$\|n=1,l=0\rangle$

37	cho, dinB-yafNOP, dinD, ding-ybib, dinQ, ftsK, hokE, insK, lexA, polB, ptrA-recBD, recAX, recN, recQ, rpsU-dnaG-rpoD, ruvAB, symE, tisB, umuDC, uvrB, uvrD, uvrY, ybfE, ybgA-phr, ydjM, yebG		$\lvert n=1,l=0\rangle$
38	cirA, entCEBAH, fepA-entD, fiu	crp, fur	$\lvert n=0,l=2\rangle$
39	copA, cueO	cueR	$\lvert n=0,l=1\rangle$
40	cra, pitB, sbcDC	phoB	$\lvert n=0,l=1\rangle$
41	crl_1, exbBD, fepDGC, fhuACDB, fhuE, gpmA, metJ, nohA-ydfN-tfaQ, ryhB, ygaC, yhhY, yjjZ	fur	$\lvert n=0,l=1\rangle$
42	cusCFBA, cusR, yedX	hprR, phoB	$\lvert n=1,l=2\rangle$
43	cvpA-purF-ubiX, glrR-glnB, hflD-purB, lolB-ispE-prs, purC, purEK, purL, speAB	purR	$\lvert n=0,l=1\rangle$
44	cysDNC, cysK, tcyP, yciW, ygeH, yoaC	cysB	$\lvert n=0,l=1\rangle$
45	cytR, nagC, nagE, ycdZ	crp	$\lvert n=1,l=1\rangle$
46	dapB, lysC	argP	$\lvert n=0,l=1\rangle$
47	ddpXABCDF, patA, potFGHI, yeaGH, yhdWXYZ	ntrC	$\lvert n=0,l=1\rangle$
48	decR, mlaFEDCB, yncE	marA	$\lvert n=0,l=1\rangle$
49	dgcC, iraP, nlpA, wrbA-yccJ, yccT	csgD	$\lvert n=0,l=1\rangle$
50	dicB-ydfDE-insD-7-intQ, dicC-ydfXW	dicA	$\lvert n=0,l=1\rangle$
51	dsdC, norR	nsrR	$\lvert n=1,l=1\rangle$
52	dtpA, omrA, omrB	ompR	$\lvert n=0,l=1\rangle$
53	ecpA, ecpR	matA	$\lvert n=0,l=1\rangle$
54	efeU_1U_2, motAB-cheAW, psd-mscM, tsr, ung	cpxR	$\lvert n=0,l=1\rangle$
55	epd-pgk-fbaA, gapA-yeaD, mpl	cra, crp	$\lvert n=0,l=2\rangle$
56	erpA, iscR, rnlAB		$\lvert n=1,l=0\rangle$
57	evgA, nhaR	hns	$\lvert \varphi_d=1.3802..,l=1\rangle$
58	fabA, fabB	fabR, fadR	$\lvert n=0,l=2\rangle$
59	fadE, fadIJ	arcA, fadR	$\lvert n=0,l=2\rangle$
60	fbaB, fruBKA, glk, gpmM-envC-yibQ, pfkA, ppc, pykF, pyrG-eno, tpiA	cra	$\lvert n=0,l=1\rangle$

61	fldB, pgi, ribA, ydbK-ompN	soxS	$\|n=0,l=1\rangle$
62	folE-yeiB, metA, metC, metF	metJ	$\|n=0,l=1\rangle$
63	fpr, pqiABC, rirA-waaQGPSBOJYZU	marA, soxS	$\|n=0,l=2\rangle$
64	fucAO, fucR, zraR	crp	$\|n=1,l=1\rangle$
65	gfcA, ybhL, yfiR-dgcN-yfiB, ymiA-yciX	yjjQ	$\|n=0,l=1\rangle$
66	hupA, trg	crp, fis	$\|n=0,l=2\rangle$
67	ibaG-murA, rplU-rpmA-yhbE-obgE	mlrA	$\|n=0,l=1\rangle$
68	ibpAB, yadV-htrE	IHF	$\|n=0,l=1\rangle$
69	idnK, idnR	crp, gntR	$\|n=1,l=2\rangle$
70	isrC-flu, pth-ychF	oxyR	$\|n=0,l=1\rangle$
71	lgoR, uxuR	crp, exuR	$\|\varphi_d=1.6180..,l=2\rangle$
72	lolA-rarA, osmB	rcsB	$\|n=0,l=1\rangle$
73	lsrACDBFG-tam, lsrR, oxyR, rbsR	crp	$\|n=1,l=1\rangle$
74	malI, mlc	crp	$\|n=1,l=1\rangle$
75	manA, yhfA	crp	$\|n=0,l=1\rangle$
76	mngAB, mngR		$\|n=1,l=0\rangle$
77	nadA-pnuC, nadB	nadR	$\|n=0,l=1\rangle$
78	nimR, nimT		$\|n=1,l=0\rangle$
79	ompX, rpsP-rimM-trmD-rplS, ychO, ysgA	fnr	$\|n=0,l=1\rangle$
80	pepD, yhbTS	csgD	$\|n=0,l=1\rangle$
81	phoP, slyB		$\|n=2,l=0\rangle$
82	pspABCDE, pspG	IHF, pspF	$\|n=0,l=2\rangle$
83	purR, pyrC	fur	$\|n=1,l=1\rangle$
84	rhaR, rhaS	crp	$\|n=2,l=1\rangle$
85	rrsA-ileT-alaT-rrlA-rrfA, rrsE-gltV-rrlE-rrfE	fis, lrp	$\|n=0,l=2\rangle$
86	rrsB-gltT-rrlB-rrfB, rrsC-gltU-rrlC-rrfC, rrsD-ileU-alaU-rrlD-rrfD-thrV-rrfF, rrsG-gltW-rrlG-rrfG, rrsH-ileV-alaV-rrlH-rrfH	fis, hns, lrp	$\|n=0,l=3\rangle$
87	ssb, uvrA	arcA, lexA	$\|n=0,l=2\rangle$
88	ttdABT, ttdR		$\|n=1,l=0\rangle$

89	ycjG, ycjY-ymjDC-mpaA	pgrR	$\|n = 0, l = 1\rangle$
90	yegRZ, yfdX-frc-oxc-yfdVE	evgA	$\|n = 0, l = 1\rangle$
91	ykgMO, znuA, znuCB	zur	$\|n = 0, l = 1\rangle$

TABLE VI: List of fiber building blocks with ID, genes in the fiber, external regulators of the fiber and fiber numbers. We provide Supplementary File 1 which plots every building block using the same IDs.