Journal of Computational Science 2 (2011) 165–177

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Beyond graphs: A new synthesis

Claudio Mattiussi*, Peter Dürr, Daniel Marbach, Dario Floreano

Laboratory of Intelligent Systems, École Polytechnique Fédérale de Lausanne (EPFL), Switzerland

ARTICLE INFO

Article history:
Received 28 July 2010
Received in revised form 11 January 2011
Accepted 25 January 2011
Available online 10 March 2011

Keywords:
Graph
Terminal graph (t-graph)
Network representation
Network synthesis
Network inference
Complex networks
Gene networks
Neural networks

ABSTRACT

Artificial neural networks, electronic circuits, and gene networks are some examples of systems that can be modeled as networks, that is, as collections of interconnected nodes. In this paper we introduce the concept of the terminal graph (t-graph for short), which improves on the concept of graph as a unifying principle for the representation, computational synthesis, and inference of technological and biological networks. We begin by showing how to use the t-graph concept to better understand the working of existing methods for the computational synthesis of networks. Then, we discuss the issue of the "missing methods", that is, of new computational methods of network synthesis whose existence can be inferred using the perspective provided by the concept of t-graph. Finally, we comment on the application of the t-graph perspective to problems of network inference, to the field of complex networks, social networks, and to the understanding of biological networks and developmental processes.

1. Introduction

Many biological and technological systems of considerable relevance can be modeled as networks, that is, as collections of interconnected devices. For example, electronic circuits are collections of interconnected electronic devices; neural networks are collections of interconnected neurons; gene regulatory networks are collections of interconnected genes, and similar interpretations can be maintained for many other systems like communication networks, control systems, and metabolic networks. Typically, these systems contain nonlinearities, feedback loops, and exhibit complex dynamics at different timescales. As a consequence, they are in general difficult to synthesize (design) by hand. Thus, many deterministic and stochastic methods for their computational synthesis have been proposed. For example, [13] and [41] describe several stochastic methods for the computational synthesis of neural networks, [14] and [42] illustrate various approaches to the computational synthesis of electronic circuits, and [24] considers the problem in the context of biological networks.

In this paper we describe a new approach to the representation of networks. This approach is based on the concept of *t-graph*, which will be introduced in the next section. The proposed approach provides a common framework for the classification and assessment of existing methods, and for the design of new methods for the computational synthesis and inference of networks. Thus, in addition to its contribution to the understanding of existing methods, the present work constitutes also a position paper proposing a research agenda for the development of new algorithms for the synthesis and inference of networks.

2. t-Graphs

To introduce the concept of t-graph let us start by considering the artificial neural network represented in Fig. 1a. Here the devices[1] are artificial neurons, and they are connected by weighted directed links. This kind of neural network is often represented using the abstract structure from graph theory called *weighted directed graph* [39,18], which is composed of a set of nodes, a set of directed edges connecting ordered pairs of nodes, and a function that assigns a weight to every edge. For many applications the graph drawn in Fig. 1a is ideally suited to model the neural network. In fact, in graph theory the term "network" is sometimes used as a synonym of "weighted directed graph" [18].

---

* Corresponding author.
E-mail addresses: claudio.mattiussi@gmail.com (C. Mattiussi), duerrp@gmail.com (P. Dürr), dmarbach@mit.edu (D. Marbach), dario.floreano@epfl.ch (D. Floreano).

[1] Strictly speaking, we should distinguish the actual devices from their models. For example, in circuit theory one refers to the actual devices as *circuit devices* and to their models as *circuit elements* [6,16]. With some abuse of terminology, we will use the term "device" to refer to both the actual devices and to their models, relying on the context for disambiguation.

1877-7503/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.jocs.2011.01.007

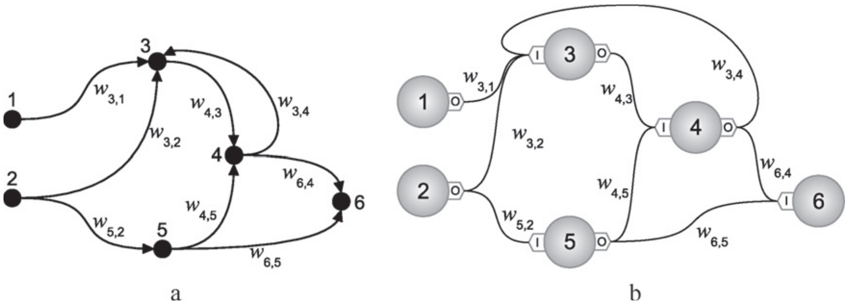

Fig. 1. (a) The conventional graphical representation of an artificial neural network. The network is seen as a collection of devices, drawn here as black circles, connected by directed weighted links, drawn as labeled arrows. (b) The representation of the same neural network as a collection of devices with terminals connected by weighted links. According to their role, the devices in the network are now explicitly attributed input (I) and output (O) terminals, and the links connect the terminals rather than the devices.

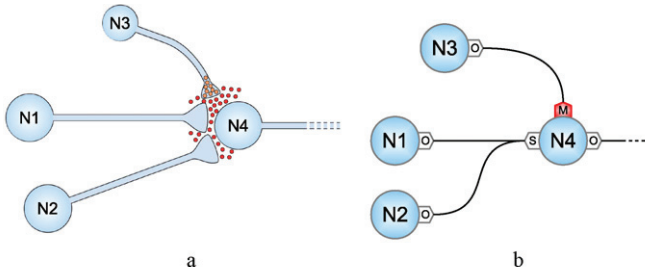

Fig. 2. (a) A schematic representation of neuromodulatory connections in biological neural networks. Neuron N3 releases a neuromodulator (represented as small vesicles) that affect the interaction between the remaining neurons. (b) The presence of neuromodulatory effects affecting neuron N4 can be modeled by adding a distinct modulatory input terminal (M) to the standard input (S) of the conventional artificial neuron model.

For some other applications, however, the graph abstraction can be improved by the simple act of specifying and keeping distinct the identity of the points of interconnection between the devices. Adopting the circuit theory terminology [6,16] we will call the points of interconnection between the devices, *terminals*. Using this approach, the neural network of Fig. 1a can be represented as in Fig. 1b, that is, as a collection of devices whose terminals are connected via weighted links.[2] In this simple example the two representations look very similar and the advantages of the new approach are not readily apparent. Let us then consider a few additional examples where the advantages of the new representation can be better appreciated.

The first example is drawn from biological neural systems. In these systems a crucial role is played by signals called *modulatory signals* [19] which influence the characteristics of the neurons or the behavior of the connections between them, rather than their level of activation (Fig. 2a). The influence of modulatory signals range from the change of the sensitivity of the neurons to incoming signals, to the reconfiguration of the network connectivity, to the modulation of the plasticity and learning rates. Observations of biological neural systems [19] and experiments with simulated agents [33,10] provide evidence that the inclusion of neuromodulatory effects can substantially increase the potential and evolvability of neural networks. The presence of a neuromodulatory signal affecting the characteristics of a neuron and of its synapses can easily be included in an artificial neuron model by adding a modulatory input to the model. This implies, however, that a neuron can no longer be simply represented as a node in a graph as in Fig. 1a, because the model must retain the distinct character of the neuron inputs. In the words of neurobiologist Katz [19, p. 15]:

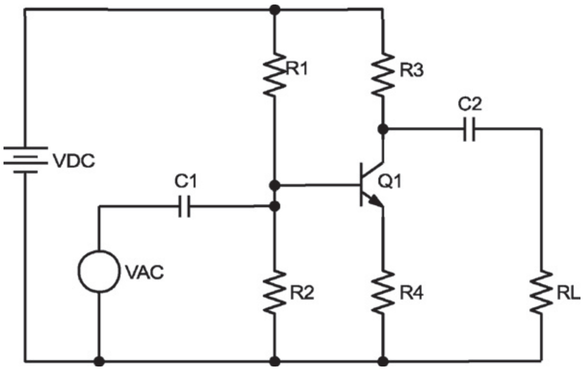

Fig. 3. The schematic representation of a simple electronic circuit.

> By ignoring neuromodulatory actions, these so-called 'ball and stick' diagrams [i.e., diagrams equivalent to those shown in our Fig. 1a] do not represent adequately the richness of communication between neurons. Such diagrams can be very misleading because [. . .] neurons communicate with both neurotransmission and neuromodulation simultaneously, providing a rich environment where signals vary not only in time, space, and intensity but also in *character*. [emphasis in the original]

To overcome the limitation lamented by Katz it is necessary to model neurons as devices with as many separate connection points as there are inputs and outputs conveying signals with different character, i.e., signals playing distinct roles in the operation of the device.[3] For example, the effect of the neuromodulatory neuron N3 on the neuron N4 in Fig. 2a can be represented using for N4 a model with three terminals, as shown in Fig. 2b.[4]

As second example, let us consider electronic circuits. Fig. 3 shows a schematic diagram of a simple electronic circuit, where the symbol associated with the label *Q*1 stands for a bipolar transistor, that is, a semiconductor device with three terminals called base (B), collector (C) and emitter (E) (Fig. 4a). Using for the transistor

---

[2] Note that in the representation of Fig. 1b the links are undirected, because the role played by the directed nature of the links in the graph representation is taken by the identity of the terminals in the new representation.

[3] On the other hand, signals that have the same role (e.g., because they are all summed together before further processing) can be carried by links connected to the same terminal.

[4] Note that in the model represented in Fig. 2b it is the special nature of the input terminal M which distinguishes the role of the neuromodulatory signal, and not the special nature of neuron N3. This means that, in the t-graph perspective, we can interpret the existence of special neuromodulatory neurons in biological brains as motivated by the need to keep distinct the character of the signals conveyed to other neurons, and by the biological constraint that signals with different roles are better kept distinct and carried by different chemicals.

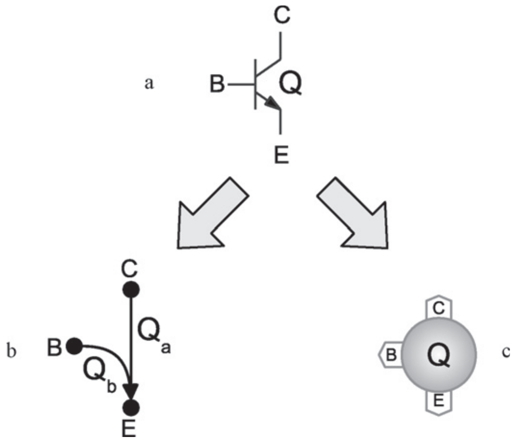

Fig. 4. (a) The standard symbol representing a bipolar transistor for circuit drawing. (b) The representation of a transistor as a graph with two edges used for circuit analysis. (c) The representation of a transistor as a device with terminals suited to the new approach proposed in this paper.

the conventional graph representations used in circuit theory [6] (Fig. 4b), and representing each two-terminal device as a directed edge, the circuit can be drawn as the graph shown in Fig. 5a. This graph representation is very useful if one wants to write the circuit equations and compute the circuit currents and voltages. Fig. 5a, however, reveals that devices with three terminals correspond to two links in the graph representation (more generally, n-terminal devices correspond to $n-1$ links). Thus, although essential for the simulation of the circuit, the conventional graph representation of electronic circuits is not ideal when the unity of the devices must be preserved, for example, during the synthesis of circuits, when devices are typically tentatively inserted and removed from the circuit. A model that complies with this latter requirement is shown in Fig. 4c. Now the transistor is represented as a single device with three terminals. Using this same approach to model all the non-resistive devices appearing in the circuit of Fig. 3, one can represent the circuit as in Fig. 5b, that is, as a collection of devices whose terminals are interconnected via links which are weighted by the value of resistance between pairs of terminals.

A further example that illustrates the advantages of endowing with terminals the traditional nodes of a network can be drawn from the design of computer networks and, more generally, of communication networks. Present-day computers and routers are typically equipped with several distinct communication interfaces [32]. For example, a computer may be equipped with several types of wireless interface (implementing, for example, the Bluetooth, Wi-Fi 802.11, and Consumer Infrared standards) and several types of wired interface (e.g., Ethernet, Firewire, USB). Each type can further come in different versions having different maximum transfer rates (e.g. 802.11 b/g/n, Ethernet 10/100/1000 Mbit/s, USB 1.1/2.0/3.0). The process of synthesizing and optimizing a computer network must take into account the number, type, and version of the communication interfaces available on each computer and router composing the network. This is awkward using the conventional representation of the network as a graph since there is no natural way to identify the types of interfaces available on each node, but can be easily done by representing the system as a network of devices with terminals that correspond to the type and version of communication interfaces available on the device.

As final example, let us consider gene regulatory networks (from now on, simply *gene networks*). In the traditional graph representation of gene networks the genes are represented as nodes and the interactions between regulator genes and regulated target genes are represented as directed links [15]. But – in the words of Huang and Kauffman [15, p. 1182] –

> since each node can receive several inputs ("upstream regulators"), it is more appropriate to combine modality of interaction together with the way the target gene integrates the various inputs to change its expression behavior (= output). Thus, each node can be assigned a function that maps all its inputs in a specific way to the output.

A discussion with further details on biological networks and on gene networks in particular can be found in Section 6.4 below. However, a consequence of Huang and Kauffman's observation that is already apparent at this stage is that each gene should be represented as a device having one terminal for each way the target gene can functionally integrate its inputs [28,23], rather than as a node of the conventional graph representation, because the latter does not permit to identify the way the regulator and regulated genes interact.

Summing up, we have shown that many systems traditionally represented as weighted graphs can be effectively and usefully represented as collections of devices that are explicitly provided with one or more points of interconnection called terminals. Given the crucial role played in it by terminals, we propose the name *terminal graph* (*t-graph* for short) for this kind of representation.[5]

3. Representation and computational synthesis of networks

In order to permit the computational use of the t-graph concept we need a computational representation for it. A t-graph is constituted of (see the Appendix A for a formal definition):

1. A finite collection of (possibly parametrized) *devices*
2. Each device possessing a finite collection of *terminals*
3. A collection of *links* between the terminals, each associated with a value of *interaction strength* or *weight*

A simple computational representation for these elements can be obtained by *labeling* the device terminals and defining an *interaction map*[6] which associates a numeric value to every unordered pair of terminal labels, possibly with a special value (e.g., zero) used to denote the absence of a link between a pair of terminals.

As mentioned in the introduction, one of the most significant problems concerning networks is their computational synthesis. Given the constituents of the representation for t-graphs listed above, a computational method of synthesis for networks can be structured as a search involving one or more of the following three elements (Fig. 6):

1. The number, type, and parameter values of the *devices* composing the network
2. The *labels* associated with the terminals of the devices
3. The *interaction map* associating a weighted link with each pair of terminal labels

As we will show below, existing methods for the computational synthesis of networks perform a search on one or two of these three elements, and assign the remaining ones. We can thus classify these methods according to which elements of the representation are preassigned and which elements are instead the subject of a search

---

[5] In previous works [26–28] we used the term *analog network* for this same representation.

[6] We use the term *map* rather than *function* to emphasize the fact that the terminal labels are elements of an arbitrary set, which is not necessarily a numeric field.

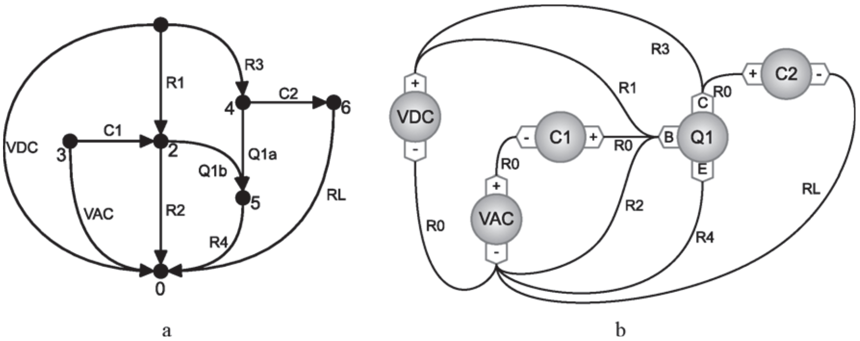

Fig. 5. (a) The representation of the electronic circuit of Fig. 3 as a directed graph. (b) The representation of the same electronic circuit as a collection of devices with terminals connected by weighted links.

process. To simplify the classification we will consider jointly the number, the type, and the parameter values of the devices, and distinguish only methods that operate on one or more of these device properties from methods that do not implicate any of them in the search.

In the rest of this paper we will mainly cite examples of evolutionary synthesis of artificial neural networks, considered as representative of the general problem of defining methods for the computational synthesis of networks. The reason for this choice is that there exist a vast literature and a rich variety of methods for the evolutionary synthesis of neural networks [41,13]. This choice does not entail a loss of generality, because the discussion and results apply to any kind of network, and they are valid in general for stochastic and deterministic search and optimization methods.

4. Existing methods for the computational synthesis of networks

4.1. Methods that search the space of interaction maps

The earlier methods [13,41] that were developed for the computational synthesis of networks assign and keep fixed during the synthesis the number, type, and parameter values of the devices that compose the network. They also implicitly assign and keep

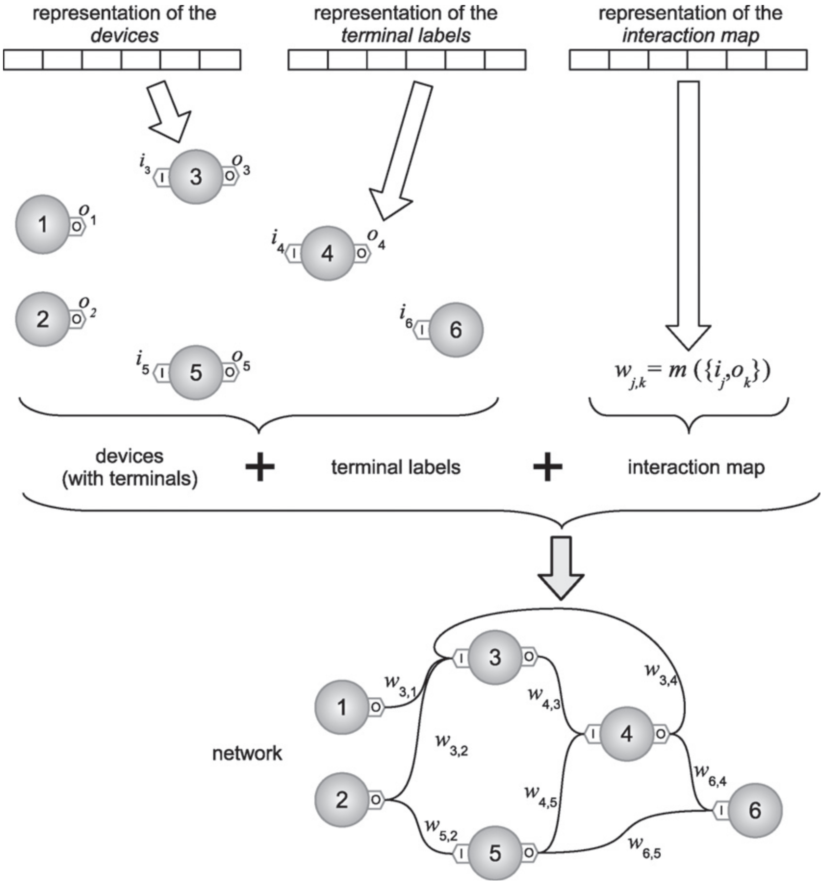

Fig. 6. The elements that realize the t-graph representation of a network in view of its computational synthesis: a collection of *devices* (with terminals), a collection of *terminal labels* (here, i_j for input terminals and o_k for output terminals), and an *interaction map*. The interaction map $m(\cdot)$ takes as arguments an unordered pair of terminal labels and produces the value of interaction strength between the corresponding terminals. By applying the interaction map to all pairs of terminal labels, one obtains the complete network.

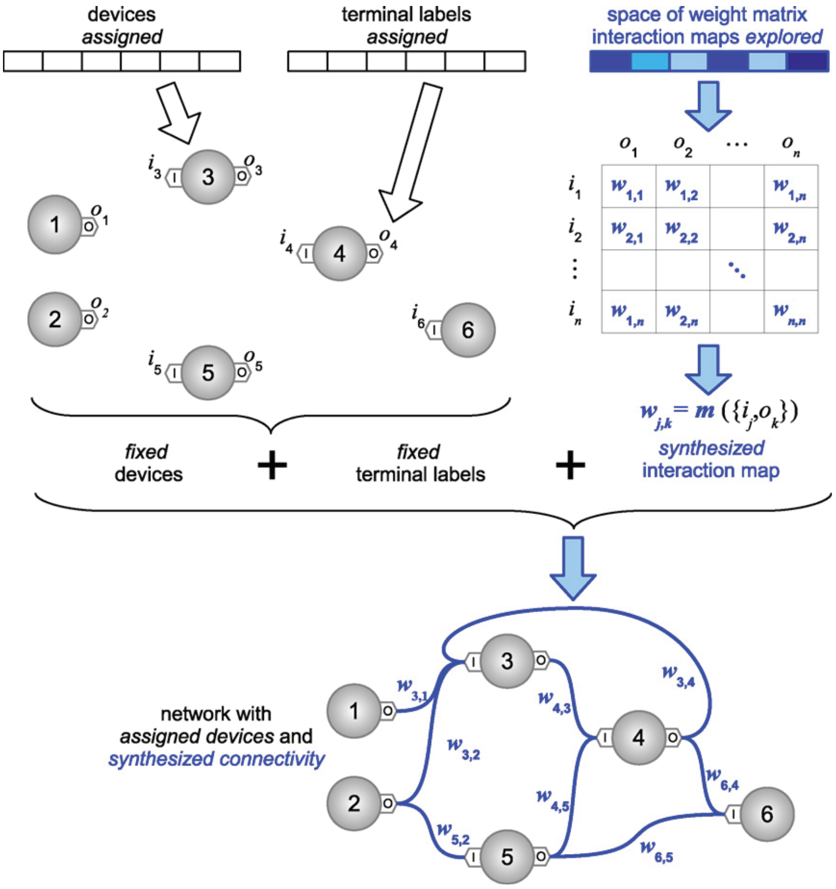

Fig. 7. A schematic representation of a class of methods for the synthesis of networks which assigns the collection of devices with labeled terminals of the t-graph representation, and synthesizes the interaction map as a matrix of values. The synthesis of the matrix entries is typically performed with a deterministic or stochastic search method (e.g., gradient descent, or an evolutionary algorithm). In this schematic representation and in those shown in the following pages, the elements of the network that are the subject of the synthesis process are represented in blue and drawn with thick lines. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of the article.)

fixed the terminal labels of the t-graph representation by deriving them from the labels of the devices. Once the devices and terminal labels are assigned, what remains as the subject of the synthesis is thus the sole interaction map. For example, assuming as given the indexed collection of devices shown in the top row of Fig. 6, the labels i_j and o_j can be considered as implicitly assigned to the input and output terminal, respectively, of the jth device. The fact that actual connections can be established only between inputs and outputs corresponds to assigning a default null value of the interaction map to all the pairs of terminal labels of the type (i_j, i_k) and (o_j, o_k). The synthesis method is thus required to generate the values of interaction strength $w_{j,k} = m(\{i_j, o_k\})$ taken by the interaction map on all the unordered pairs of terminal labels of the type (i_j, o_k). This corresponds to the generation of the entries of a matrix (Fig. 7).[7]

Examples of methods falling under this heading are most of the fixed-genome-length methods for evolutionary neural network synthesis described in [13,41]. Some of these methods encode and evolve the entries of this matrix in a binary genome; other methods, for example, Evolutionary Strategies [2], represent and evolve this matrix as a real-valued vector. Note that even classical methods of neural network training such as back-propagation of error [3,14] can be interpreted in this framework as searching for the minimum of the error function in the space of interaction maps represented by the matrix of values shown in Fig. 7. Searching in the space of interaction maps that are represented as matrices of map values has the advantage of simplicity but suffers from a lack of scalability since the number of entries in the matrix grows quadratically with the number of devices in the network. This is a well-known problem of representations of multidimensional maps based on simple structures such as matrices and look-up tables [3]. We will consider below methods capable of assuaging this problem.

4.2. Methods that operate on an efficient representation of the interaction map

In the t-graph perspective, an obvious way to overcome the scalability problem of the matrix representation of the interaction map that was mentioned above is to employ a more efficient representation of the interaction map (Fig. 8). An example of a potentially efficient representation of a map is a feed-forward neural network (Fig. 8c). A feed-forward neural network represents a map as a parametrized composition of simple maps selected from a set of *neuron activation functions* [3]. The first benefit of this kind of representation is that with a judicious choice of the activation functions one can generate a large variety of maps by using a reasonably small number of parameters. A neural network used to represent the interaction map of a t-graph will take pairs of terminal labels as inputs, and will produce the value of interaction strength

---

[7] This matrix is similar to the *weighted adjacency matrix* defined for weighted graphs [4], except for its defining the values associated with connections between terminals rather than with connections between nodes.

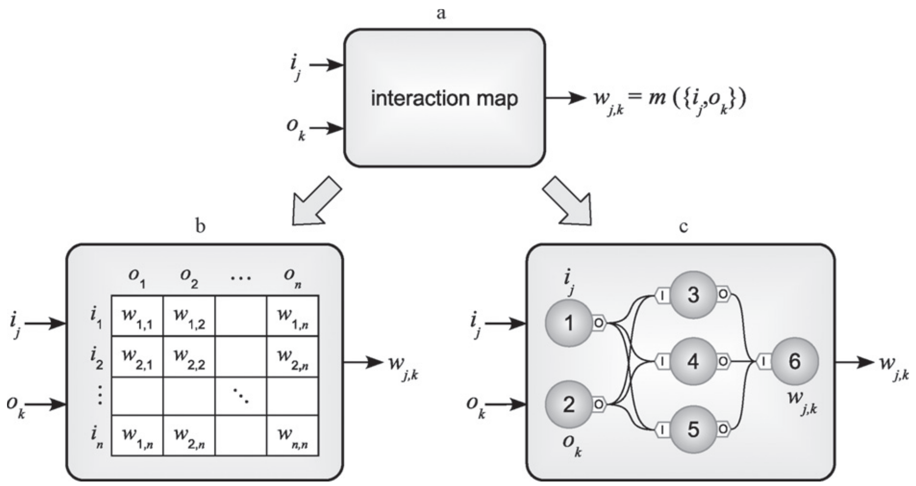

Fig. 8. Methods for the synthesis of networks that search in the space of the interaction maps (a) can operate on a simple representation of the map such as a weight matrix (b) or adopt a more efficient representation of the map. For example (c), the interaction map can be represented as a neural network that takes as inputs pairs of terminal labels and produces as output the value of the weight associated with the connection between the terminals.

associated with the connection between the corresponding terminals as output.[8]

Using this representation, a method for the synthesis of networks which performs a search in the space of interaction maps will actually search in the space of the neural networks representing the interaction map. More generally, this kind of method will search in the space of some efficient representation of the interaction map (Fig. 9). An efficient representation of the interaction map such as a feed-forward neural network has the benefit that it scales better than the matrix representation when the number of devices and terminals of the synthesized network increases. Moreover, a composition of functions defined on a continuous domain is not limited to a unique, predefined resolution, but, once generated by the search process, it can be sampled at several different resolutions and can thus be used to generate the connectivity pattern of networks with largely different number of devices [35]. An actual example the class of methods illustrated in Fig. 9 is *Hyper-NEAT* [35,9].

4.3. Methods that search the space of devices and interaction maps

To increase the flexibility of the methods described in Section 4.1, many methods add to the search in the space of the interaction maps, a search in the space of devices [13,41]. Fig. 10 shows a schematic representation of a class of methods that explicitly represent the number and kind of devices present in the network and the weights between the terminals of the devices. This gives the possibility of generating networks with different numbers of devices. Examples of methods falling under this heading are many of the variable-genome-length methods for evolutionary neural network synthesis described in [13,41].

4.4. Methods that search the space of devices and terminal labels

An approach to the synthesis of networks which is complementary to the assignment of the terminal labels combined with the synthesis of the interaction map described so far, consists in assigning the interaction map while synthesizing the devices and the terminal labels. Fig. 11 shows a schematic representation of this class of methods, where the terminal labels, the number, the type, and the parameters of the devices forming the network are synthesized by the search algorithm. This approach to the synthesis of networks has the advantage that the interaction map must no longer be represented in view of the search, an operation which, as we have seen, can suffer from scalability problems. On the other hand, this approach creates the problem of the choice of an interaction map, which may also be far from trivial [26]. Examples of methods for the synthesis of networks belonging to this class are the evolutionary methods based on the *analog genetic encoding* (AGE) [26–28].

5. The missing methods

The examples considered so far show that it is possible to classify existing methods for the computational synthesis of networks according to their way of handling the three elements that form the t-graph representation of the network (Fig. 6). We have seen that each element of the representation can be either assigned and kept fixed, or synthesized by the algorithm. Moreover, we have seen that when the interaction map is synthesized, it can be represented either in an inefficient or in an efficient way. The upper half of Table 1 summarizes this classification. Note that further combinations corresponding to methods that are obvious simplifications of existing entries have not been included in the table (e.g., synthesizing only the terminal labels while keeping fixed both the devices and the interaction map would be a simplification of the methods corresponding to the fourth row).

What is interesting at this point is to consider combinations that, to the best of our knowledge, do not correspond to any existing method for the computational synthesis of networks. We will call the methods corresponding to these combinations the *missing methods*. Let us proceed to consider a few of these and to list them in the lower half of Table 1. A first combination that does not appear among the existing methods is one where all the three elements of the representation are synthesized. Note that the representations considered so far for the devices and terminal labels are all potentially inefficient, in the sense that they involve the explicit representation of each device or of each terminal label. It is certainly possible, however, to conceive of an efficient representation for the devices and the terminal labels as well. Just to give an example, mimicking the efficient representation of the interaction map

---

[8] Note that, as pointed out previously, each terminal label can be an entity more complex than a single numeric value. For example, it could be a list of numeric values, or an arbitrary sequence of symbols. Thus, each of the two inputs of the box representing the interaction map in Fig. 8a must be interpreted as potentially corresponding to several (not necessarily numerical) actual inputs.

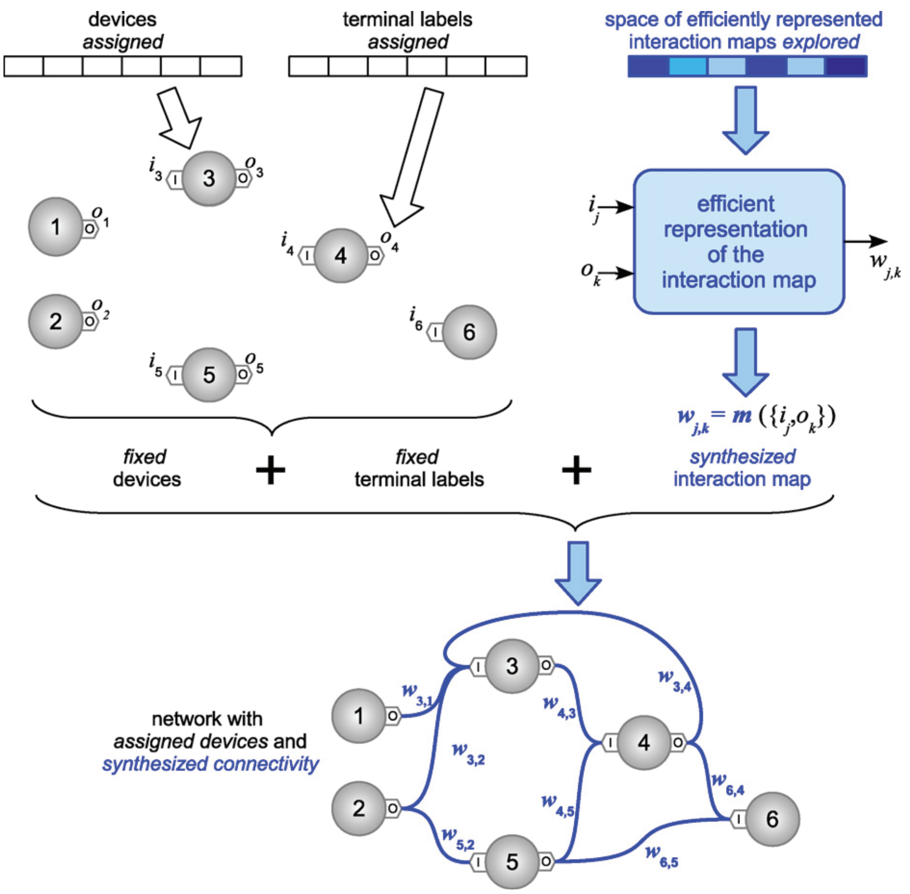

Fig. 9. A schematic representation of a class of methods for the synthesis of networks which assigns the collection of devices with labeled terminals and synthesizes an efficient representation of the interaction map.

described above, we could use a function whose properties (say, the position in the domain and the nature of some kind of critical point of the function [11,43]) determine a set of coordinates to be used as terminal labels and the properties of the device, respectively. This means that we can conceive of classes of missing methods that synthesize an efficient representation of one or more of the three elements forming the computer representation of t-graphs, as shown schematically in the last four rows of Table 1.

6. Discussion

Equipped with the t-graph concept and with the classification scheme of Table 1 for the methods of network synthesis we can now consider a few questions that arise in relation to the application of the missing methods and discuss the place of further social, technological, and biological examples of networks in the classification scheme of Table 1.

Table 1
The sketch of a classification scheme for the methods of synthesis of networks (a finer classification can be obtained considering separately the status of the number, type, and parameters of the devices). In the synthesis of a network, each element of the t-graph representation can be either *fixed* or *synthesized*. Additionally, the synthesized elements can be represented in an inefficient or an *efficient* way. The top half of the table lists existing methods that have been discussed in the text. The bottom half of the table represents combinations that do not correspond to existing computational methods for network synthesis (missing methods). Besides methods that synthesize all the elements of the t-graph representation, we can infer the existence of missing methods based on combinations of one or more efficient representations of the devices, terminal labels, and interaction map.

	Devices	Terminal labels	Interaction map	Examples
Existing methods	Fixed	Fixed	Synthesized	Fixed-length direct encodings
	Fixed	Fixed	Synthesized, with efficient representation	Hyper-NEAT
	Synthesized	Fixed	Synthesized	Variable-length direct encodings
	Synthesized	Synthesized	Fixed	AGE
Missing methods	Synthesized	Synthesized	Synthesized	Biological circuits (e.g., gene networks), but no computational method so far
	Synthesized, with efficient representation	Fixed or synthesized	Fixed or synthesized	
	Fixed or synthesized	Synthesized, with efficient representation	Fixed or synthesized	
	At least one synthesized		Synthesized, with efficient representation	
	At least two synthesized and with efficient representation			

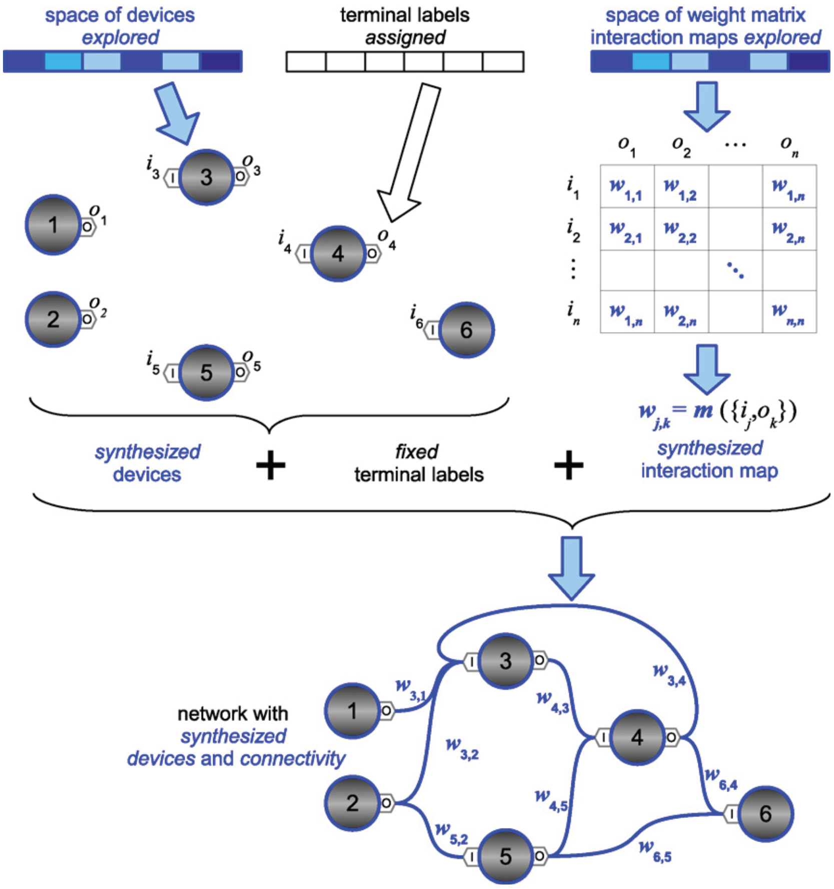

Fig. 10. A schematic representation of a class of methods for the synthesis of networks which synthesizes both the collection of devices with terminals composing the t-graph representation, and the interaction map represented as a matrix of values. The terminal labels are implicitly derived from the device identifiers.

6.1. Applications of the missing methods

A first question that is worth considering concerns the possible advantages and applications of the missing methods listed in Table 1. For example, assuming that it is necessary to consider networks having different number and type of devices in the search, one can consider the possibility of defining a new method which synthesizes both the interaction map and the terminal labels rather than use existing methods that operate on only one of these two elements. The possibility of using methods that operate on the space of both elements adds degrees of freedom to the definition of the network, so that a search process that might get stuck when operating on only one of these elements could use the additional degrees of freedom to escape this condition. Other advantages and improvements over existing methods might stem from the use of novel combinations of efficient representations. For example, the experiments reported in [35] have shown how the use of an efficient representation for the interaction map can lead to an excellent scalability of the outcome of the synthesis. The use of combinations of more than one efficiently encoded element of the t-graph representation could further extend this favorable property to cases where, for example, it is also crucial to define networks that are heterogeneous in terms of the nature of the devices.

We must keep in mind, however, that each type of method has its own niche of applicability, and that we don't always have to turn to the most complex method. For example, if the network need only be composed of a small and fixed number of preassigned devices, the adoption of a simple, inefficient representation of the interaction map might be the best choice.

6.2. Network inference

So far we have considered mainly the impact of the t-graph perspective on the computational *synthesis* of networks. There is, however, another important class of computational problems concerning networks: that of *network inference* (also called *reverse engineering* of networks). In a network inference problem one is given some prior information about the network and some data collected from it, and is asked to estimate its structure and parameters. As explained in [23,28,36], a computational inference process can be based on algorithms that synthesize an ensemble of tentative solutions. Thus, the approach presented above in relation with the problem of classifying and designing methods for network synthesis extends naturally to the important problem of classifying and designing methods for the computational inference of networks.

6.3. Complex networks

Fostered by the wealth of newly available data on large biological, technological, and social networks, a new discipline of *complex networks* has emerged in the last few years [1,5,8,31,37]. There is no universally agreed definition of what constitutes the field of complex network studies, but we can take as a reasonably encompassing viewpoint the one that sees it (or at least most of it) as the application of the methods of statistical physics to the study of networks [8]. The main goal is typically to estimate the statistical properties of a class of networks based on a few global properties observed in a sample of networks representative of the class. Mimicking the approach followed by statistical mechanics, in studying complex networks one thus starts by observing a few structural

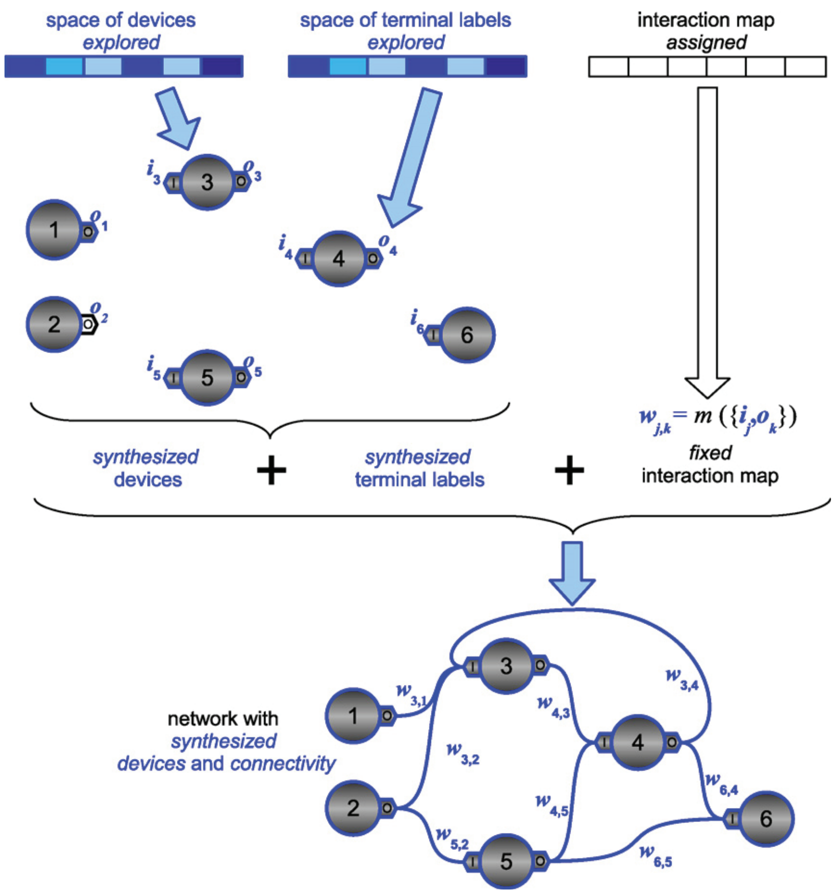

Fig. 11. A schematic representation of a class of methods for the synthesis of networks which assigns the interaction map and synthesizes the collection of devices and the terminal labels.

characteristic of some actual network (for example, one may count the total number of nodes, and evaluate the degree distribution [1,8] of, say, a given gene network). Next, one tries to determine the ensemble of networks that comply with the observed structural characteristics. Ideally, one would like to determine and focus on the ensemble of networks that maximize the entropy given the constraints represented by the observed structural characteristics, since this ensemble represents the most probable type of network complying with these characteristics. Endowed with the maximum entropy ensemble one can finally determine the statistical properties possessed by the networks of the kind originally observed.

Unfortunately, for many real-world networks the number of structural characteristics that must be taken into account to reasonably characterize them is large. This is especially true if the networks are not static but change over time. The consequence is that the number of constraints that must be considered for entropy maximization is so large that the problem becomes computationally infeasible [1]. To circumvent this issue one can try to generate a sample of the maximum entropy ensemble of networks by means of stochastic dynamical procedures that specify how to construct the networks [1,5,8]. This implies that the constraints derived from the observation of the actual networks can no longer be structural characteristics but, rather, constraints on the construction procedures themselves. This, in turn, presupposes that one can easily represent those constraints in the network construction algorithm.

It is in relation to this requirement that we see a first way in which the study of complex networks can benefit from the concept of t-graph introduced in this paper. As we have shown in the examples of Section 2, for many real-world networks the establishment of the connections between the devices is contingent upon the nature of the device terminals. Thus, any growth process concerning the network is constrained by the nature of the terminals. This implies that the representation of the real-world network as a collection of devices with terminals, i.e., as a t-graph, is essential for the representation of the actual constraints existing in the process of network growth.[9]

This aspect is related to a second way in which the t-graph perspective can benefit the study of complex networks, namely, in helping to answer some criticisms that have been leveled at the complex network approach regarding its actual relevance for many kinds of real-world systems, and the necessity to move beyond observation and modeling to tackle problems of validation and control [20,22,29]. For example, Li and coworkers note that [22, p. 495] "[in] domains like engineering (or biology) . . . design, evolution, functionality, and constraints are all key ingredients that simply cannot be ignored". The use of the t-graph perspective can help realize a first step towards the inclusion of these ingredients in the complex network analysis by expressing existing functional constraints via the functional identity of the device terminals. Discussing the problem of adopting new approaches to validation, Mitzenmacher observes that [29, p. 529] "A compelling possibility is to make use of time series analysis, with the goal that one observes the system over time to judge the underlying assumptions of the proposed model." As shown in [24] for the case of gene networks,

---

[9] Note that once the collection of networks has been generated using the t-graph representation, one can always reduce the t-graph representation to a conventional graph representation in order to perform the traditional kind of statistical analyses on the collection.

the analysis of the compatibility of a given network model with a collection of observed time series can be effectively founded on a process of network inference. Thus, the use of time series analysis advocated by Mitzenmacher may benefit from the new methods of network inference whose existence is revealed by the t-graph perspective.

Finally, the t-graph concept can lead to the identification of new classes of networks sharing important structural properties. For example, many real-world systems can be seen as networks composed by devices possessing input and output terminals, and a *gating* terminal that regulates the input–output flow (active devices in electronic circuits, enzymes in metabolic networks, and gated and modulated neurons in neural networks can be interpreted as such kind of device [25], and the effect of sanitary interventions might be interpreted in a similar way in the context of the modeling of epidemic spreading in population networks [7]). Thus, one could conceive a program of study of the properties of a class of "gated" complex networks. Other classes of networks may be identified on the basis of similar structural properties related to the nature of the device terminals.

6.4. Biological networks

Another interesting question is whether the genetic representations of biological networks can be classified according to the scheme of Table 1 and, the case being, to which class they belong. Let us first consider the case of gene networks.

As schematized in Fig. 12, and detailed in [28,23], a gene network can be modeled as a t-graph where the genes correspond to the devices and the interaction between two genes is determined by the interaction between the gene products of the first gene with the regulatory regions of the second gene. Thus, it is possible to consider the coding and the regulatory regions of two genes as terminal labels associated with the genes' output and input terminals, respectively. Evolution can modify these regions of the genes and, thus, we can interpret the evolution of biological gene networks as a method of network synthesis where the genes are the devices and the terminal labels are encoded and evolved. Since genes can be mutated, duplicated, and inactivated, the devices are also encoded and evolved. The interaction map is determined by the cellular environment in which the genome is immersed and depend on the laws of physics and chemistry which influence the generation of the gene products and their interactions with the regulatory regions of the genome. Since the cellular environment can also change during evolution, and is determined in part epigenetically [17], in part by the molecular machinery and by other gene products encoded in the genome, we can conclude that the interaction map is also encoded and evolved. Thus, the evolutionary synthesis of biological gene networks can be interpreted as belonging to the class of methods of network synthesis that vary all the three elements of the t-graph representation. Concerning the distinction between inefficient and efficient representations, we might tentatively say that in the case of biological gene networks the representation of the devices and terminal labels is inefficient, since all the elements are explicitly encoded in the genome. The case of the interaction map is less clear, because the genome also influences the properties of the interaction map via the encoding of the molecular machinery and of the chemicals that will be released within the cell. One could argue that this representation is potentially efficient.

Another example of biological networks that can be modeled as t-graphs are neural networks. The case of the classification of biological neural networks according to the scheme of Table 1 is definitely less clear than that of gene networks just examined, since the synthesis of biological neural networks depends on a still poorly understood developmental process that certainly depends on the genome but is also heavily influenced by environmental factors. The devices are certainly evolved, and the connectivity is also genetically influenced. Moreover, the complexity and scalability of the neural network with respect to the genetic information points to an efficient representation. It is however difficult to say if and how in biological nervous systems the genetic instructions determine a hypothetical labeling of the device terminals and a hypothetical interaction map which establish a set of tentative connections, to be later refined according to environmental and functional factors.

6.5. Multiplex networks

Social network analysis is the joint subfield of network and social science that studies the relations between social entities called *actors* (which can be either individuals or social groups) [40]. In a *social network*, nodes represent actors, and links represent relations between actors. Typically, given a set of actors one is interested in representing and studying several distinct relations between pairs of actors. These distinct relations can be represented using several distinct networks on the same set of nodes, or using a network representation called *multiplex* (or *multivariate*) *network* that allows the presence of several distinct links between pairs of nodes [30,38,40].

A multiplex network can be realized as a t-graph by simply endowing each of its nodes, considered as a t-graph device, with as many distinct terminals as possible relations in the multiplex network, and stipulating that only pairs of terminals of the same type can be connected. In general, however, this interpretation appears contrived because it shifts the multiple nature of the relations from the links – where it belongs – to the terminals. A possible exception to this contra-indication concerns the computational inference of a multiplex network, where one is given the set of nodes and must reconstruct the links from indirect observations. In this case it appears useful to employ the t-graph representation, which, contrary to the multiplex network representation, explicitly identifies the number and kind of links that can exist between nodes before the links themselves are inferred.

6.6. Developmental representations

At first sight, the important class of developmental representations for the synthesis of networks [21,12,14] does not fit in the classification proposed above and schematized in Table 1. In most cases, however, the result of an artificial developmental process can be conceived of as defining a hierarchy of coordinate systems in the evolved structure, and the final outcome of the developmental process can be replicated in terms of a composition of functions, without the intervention of the developmental process [34].

One could object that biological developmental processes also have the additional role of permitting the conditioning of the developmental outcome by the environmental conditions under which development takes place. Although this additional role might be mimicked in an artificial setting using an efficiently encoded interaction map that includes inputs from the environment, most existing artificial developmental processes do not include any kind of effect based on the interaction with the environment. Thus, most of them could be properly represented by methods capable of reproducing just their final outcome such as those schematically illustrated in Table 1.

6.7. The self-decoding paradox

Methods of network synthesis like those described in Section 4.2, which represent and evolve the interaction map of the t-graph representation using a feed-forward neural network, that is, using a further network, materialize at first sight a paradoxical infinite regress because in the t-graph representation the decoding of the

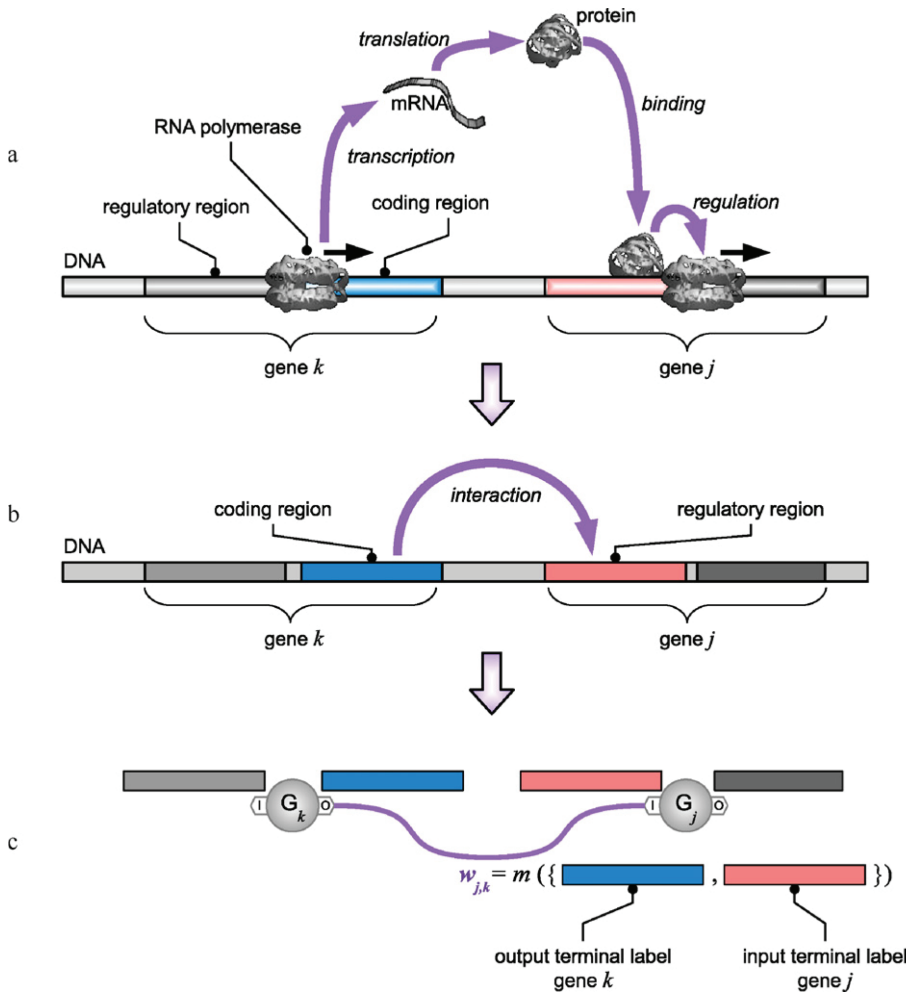

Fig. 12. (a) In biological gene networks, the interaction between genes is realized by molecules that are synthesized (by molecular machines such as the RNA polymerase) from the coding region of one gene and interact with the regulatory region of another gene. (b) Abstracting the details of the biological process, the interaction can be interpreted as involving the coding region of the first gene and the regulatory region of the second gene. (c) This abstraction can be interpreted as a t-graph where the genes correspond to the devices and the coding and regulatory regions correspond to terminal labels associated with the output and input terminals of the genes, respectively. The interaction map is determined by the action of the molecular machines that transform the coding regions into gene products, by the cellular environment, and by the laws of physics and chemistry which determine the working of the molecular machines and the interaction of the gene products with the regulatory regions.

second network in turn requires the definition of an interaction map. In fact, the paradox is easily solved. The simplest solution is to adopt two different classes of representations for the two networks, using the first network to define the interaction map of the second, and assigning or synthesizing, for example, as a matrix of weights the interaction map of the first network. Biological gene networks illustrate another interesting solution to the paradox. In biological gene networks, the molecular machinery and chemical environment that initially realize the interaction map (for example, in a fertilized cell) are essentially epigenetically inherited from the mother [17] and determine the initial "wiring" of the gene network. Subsequently, the working of this initial gene network results in the decoding of the genetic information to produce new molecular machinery and a new cellular environment, which gradually replaces those that were epigenetically inherited. In the t-graph perspective, the whole process can be interpreted as an initial assignment of the interaction map by epigenetic inheritance, followed by the gradual replacement of this interaction map with one that is encoded in the genome and can be genetically evolved. Taking inspiration from this interpretation of the biological evidence, one could conceive a computational approach to the synthesis of networks where the interaction map is efficiently represented as a t-graph, and this further network is initially decoded using an epigenetically inherited interaction map, possibly later replaced by the one encoded in the genome and initially decoded using the epigenetically inherited decoding information.

7. Conclusion

In this paper we have shown that for some important application like the computational synthesis and inference of networks, that is, of systems composed of collections of connected devices, it is very useful to represent the networks in terms of a new abstraction that we have tentatively named *terminal graph* (*t-graph* for short). This new representation differs from the conventional graph representation by attributing terminals to the devices and by connecting the device terminals rather than the devices as in the conventional graph representation. Using the t-graph concept as a guideline, we have proposed a computational representation that is based on the labeling of the device terminals and the definition of an interaction map that transforms pairs of terminal labels into weights associated with the links. This representation can be used as a basis for the computational synthesis and inference of networks. We have shown that the existing computational methods for network synthesis can be classified according to the elements of the t-graph representation that the user must assign, those that are synthesized by the search process, and on the efficiency of the representation used for the latter. Finally, we have shown that the proposed representation and the ensuing classification permits the identification of a whole class of *missing methods* for the synthesis and inference of networks which operate on combinations of the elements that constitute the t-graph representation. The next step consists in the implementation of the missing methods and in the identification of the conditions under which each of them performs best and holds the greatest potential of improvement over existing techniques for the synthesis and inference of networks.

Acknowledgments

This work was supported by the Swiss National Science Foundation, grants no. 200021-112060 and no. 200021-127143. Many thanks to Sara Mitri and Thomas Schaffter for reading and commenting on the manuscript.

Appendix A. Definitions

Graph: A *graph* [18] is a pair $G = (N, L)$ in which N is a finite set of *nodes* and L is a set of two-element subsets of N called *links*.

Weighted graph: A *weighted graph* [18] is a triple $W = (N, L, w)$ in which $G = (N, L)$ is a graph, and w is a function that assigns a *weight* to each link of G.

Terminal graph: A *terminal graph* (*t-graph*) is a quadruple $R = (D, N, L, w)$ in which D is a finite set of *devices*, $N = \{(d, t \in T_d) | d \in D\}$ is a set of *device terminals* with $\{T_d\}_{d \in D}$ a family of finite sets of *terminals* indexed by D, and $W = (N, L, w)$ is a weighted graph.

References

[1] A. Barrat, M. Barthelemy, A. Vespignani, Dynamical Processes on Complex Networks, Cambridge University Press, Cambridge, 2008.

[2] T. Bäck, F. Hoffmeister, Basic aspects of evolution strategies, Statistics and Computing 4 (June (2)) (1994) 51–63.

[3] C.M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, New York, 1995.

[4] D. Bonchev, G.A. Buck, Quantitative measures of network complexity, in: D. Bonchev, D.H. Rouvray (Eds.), Complexity in Chemistry, Biology, and Ecology, Mathematical and Computational Chemistry, Springer, New York, 2005, pp. 191–235 (Chapter 5).

[5] G. Caldarelli, Scale-Free Networks: Complex Webs in Nature and Technology, Oxford University Press, Oxford, 2007.

[6] L.O. Chua, C.A. Desoer, E.S. Kuh, Linear and Nonlinear Circuits, McGraw-Hill, New York, 1987.

[7] V. Colizza, M. Barthélemy, A. Barrat, A. Vespignani, Epidemic modeling in complex realities, Comptes Rendus Biologies 330 (April (4)) (2007) 364–374.

[8] S.N. Dorogovtsev, J.F.F. Mendes, Evolution of Networks: From Biological Nets to the Internet and WWW, Oxford University Press, Oxford, 2003.

[9] J. Drchal, J. Koutník, M. Šnorek, HyperNEAT controlled robots learn how to drive on roads in simulated environment, in: CEC'09: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, 2009, pp. 1087–1092.

[10] P. Dürr, C. Mattiussi, D. Floreano, Genetic representation and evolvability of modular neural controllers, IEEE Computational Intelligence Magazine 5 (August (3)) (2010) 10–19.

[11] W. Ebeling, Functions of Several Complex Variables and Their Singularities, American Mathematical Society, Providence, RI, 2007.

[12] D. Federici, K. Downing, Evolution and development of a multicellular organism: scalability, resilience, and neutral complexification, Artificial Life 12 (3) (2006) 381–409.

[13] D. Floreano, P. Dürr, C. Mattiussi, Neuroevolution: from architectures to learning, Evolutionary Intelligence 1 (1) (2008) 47–62.

[14] D. Floreano, C. Mattiussi, Bio-inspired Artificial Intelligence: Theories, Methods, and Technologies, MIT Press, Cambridge, MA, 2008.

[15] S. Huang, S.A. Kauffman, Complex gene regulatory networks—from structure to biological observables: cell fate determination, in: R.A. Meyers (Ed.), Encyclopedia of Complexity and Systems Science, Springer, New York, 2009, pp. 1180–1213.

[16] IEC TC/SC 1. IEC 60050-131 2.0 b, International Electrotechnical Vocabulary—Part 131: Circuit Theory, International Electrotechnical Commission, Genève, 2002.

[17] E. Jablonka, M.J. Lamb, Epigenetic inheritance in evolution, Journal of Evolutionary Biology 11 (2) (1998) 159–183.

[18] D. Jungnickel, Graphs, Networks and Algorithms, Springer, Berlin, 2008.

[19] P. Katz (Ed.), Beyond Neurotransmission: Neuromodulation and its Importance for Information Processing, Oxford University Press, Oxford, 1999.

[20] E.F. Keller, Revisiting "scale-free" networks, BioEssays 27 (10) (2005) 1060–1068.

[21] S. Kumar, P.J. Bentley (Eds.), On Growth, Form and Computers, Academic Press, Orlando, FL, 2003.

[22] L. Li, D. Alderson, J.C. Doyle, W. Willinger, Towards a theory of scale-free graphs: definition, properties, and implications, Internet Mathematics 2 (4) (2005) 431–523.

[23] D. Marbach, C. Mattiussi, D. Floreano, Replaying the evolutionary tape: biomimetic reverse engineering of gene networks, Annals of the New York Academy of Sciences 1158 (2009) 234–245.

[24] D. Marbach, R.J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, G. Stolovitzky, Revealing strengths and weaknesses of methods for gene network inference, Proceedings of the National Academy of Sciences of the United States of America (PNAS) 107 (14) (2010) 6286–6291.

[25] C. Mattiussi, D. Floreano, Connecting transistors and proteins, in: J. Pollack, M. Bedau, P. Husbands, T. Ikegami, R.A. Watson (Eds.), ALife IX: Proceedings of the Ninth International Conference on Artificial Life, MIT Press, Boston, MA, 2004, pp. 9–14.

[26] C. Mattiussi, Evolutionary Synthesis of Analog Networks, PhD thesis, École Polytechnique Fédérale de Lausanne, Lausanne, 2005.

[27] C. Mattiussi, D. Floreano, Analog genetic encoding for the evolution of circuits and networks, IEEE Transaction on Evolutionary Computation 11 (October (5)) (2007) 596–607.

[28] C. Mattiussi, D. Marbach, P. Dürr, D. Floreano, The age of analog networks, AI Magazine 29 (3) (2008) 63–76.

[29] M. Mitzenmacher, The future of power law research, Internet Mathematics 2 (4) (2005) 525–534.

[30] P.J. Mucha, T. Richardson, K. Macon, M.A. Porter, J.-P. Onnela, Community structure in time-dependent, multiscale, and multiplex networks, Science 328 (May (5980)) (2010) 876–878.

[31] M.E.J. Newman, The structure and function of complex networks, SIAM Review 45 (2) (2003) 167–256.

[32] L.L. Peterson, B.S. Davie, Computer Networks: A Systems Approach, 4th edition, Morgan Kaufmann, San Francisco, 2007.

[33] A. Soltoggio, P. Dürr, C. Mattiussi, D. Floreano, Evolving neuromodulatory topologies for reinforcement learning-like problems, in: Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007), 25–28 September 2007, 2007, pp. 2471–2478.

[34] K.O. Stanley, Compositional pattern producing networks: a novel abstraction of development, Genetic Programming and Evolvable Machines 8 (June (2)) (2007) 131–162.

[35] K.O. Stanley, D.B. D'Ambrosio, J. Gauci, A hypercube-based encoding for evolving large-scale neural networks, Artificial Life 15 (April (2)) (2009) 185–212.

[36] G. Stolovitzky, A. Califano (Eds.), Reverse Engineering Biological Networks: Opportunities and Challenges in Computational Methods for Pathway Inference, volume 1115 of Annals of the New York Academy of Sciences, Wiley-Blackwell, New York, 2008.

[37] S.H. Strogatz, Exploring complex networks, Nature 410 (March (6825)) (2001) 268–276.

[38] M. Szell, R. Lambiotte, S. Thurner, Multirelational organization of large-scale social networks in an online world, Proceedings of the National Academy of Sciences 107 (31) (2010) 13636–13641.

[39] J. van Leeuwen, Graph algorithms, in: J. van Leeuwen (Ed.), Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity, Elsevier, Amsterdam, 1990, pp. 525–631 (Chapter 10).

[40] S. Wasserman, K. Faust, Social Network Analysis: Methods and Applications, Cambridge University Press, New York, 1994.

[41] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE 87 (September (9)) (1999) 1423–1447.

[42] R.S. Zebulum, M.A.C. Pacheco, M.M.B.R. Vellasco, Evolutionary Electronics: Automatic Design of Electronic Circuits and Systems by Genetic Algorithms, CRC Press, Boca Raton, FL, 2002.

[43] W. Zou, M. Schechter, Critical Point Theory and its Applications, Springer, New York, 2006.

Claudio Mattiussi is an independent researcher. His interests include evolutionary computation, evolutionary electronics, artificial immune systems, cellular systems, developmental systems, machine learning, probabilistic and bio-inspired intelligence and engineering, and the consistent numerical formulation of physical field problems.

Peter Dürr received the M.Sc. degree in Mechanical Engineering from the Swiss Federal Institute of Technology in Zürich, Switzerland in 2005. He is currently with the Laboratory of Intelligent Systems (LIS), School of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland. His research interest is in evolutionary computation and biologically inspired robotics. He regularly contributes to the popular "Robots Podcast".

Daniel Marbach is a Postdoctoral Research Fellow at the Computer Science and Artificial Intelligence Laboratory (CSAIL) of the Massachusetts Institute of Technology (MIT). His research aims at understanding the structure, function, and evolution of regulatory networks of biological cells by means of computational methods. During his doctoral studies at the Laboratory of Intelligent Systems, Ecole Polytechnique Fédérale de Lausanne (EPFL) he developed methods for inference and modeling of gene regulatory networks. He won the international network inference competition in 2007 (DREAM challenge) and his thesis was nominated for the best doctoral thesis at EPFL in 2010.

Prof. Dario Floreano is Director of the Laboratory of Intelligent Systems at the School of Engineering in Ecole Polytechnique Federale de Lausanne (EPFL). He received an M.A. in Cognitive Science from University of Trieste (Italy) in 1988, an M.S. in Neural Computation from the University of Stirling (UK) in 1991, and a Ph.D. in Evolutionary Robotics in 1995 from University of Trieste (Italy), all of them with distinction. In 1996 he was appointed senior researcher at the Department of Computer Science of EPFL where he established the Robot Learning group. In 1998 he was invited researcher at Sony Computer Science Labs in Tokyo. In 2000 he was awarded a Swiss National Science Foundation professorship at EPFL and in 2005 he was appointed associate professor by EPFL and established the Laboratory of Intelligent Systems. His research interests cover Bio-inspired Artificial Intelligence and Robotics. Prof. Floreano is co-founder of the International Society for Artificial Life, Inc., member of the Advisory Group to the European Commission for Future Emerging Technologies, and past-member of the Board of Governors of the International Society for Neural Networks. He published almost 200 peer-reviewed technical papers and edited and co-authored several books, among which *Evolutionary Robotics* with Stefano Nolfi (hardcover 2000; paperback 2004) and *Bioinspired Artificial Intelligence* with Claudio Mattiussi (2008), both by MIT Press. He delivered more than 100 invited talks worldwide, co-organized more than 10 international conferences, and is on the editorial board of 10 international journals.