

# An Algorithm for Automatically Discovering Dynamical Rules of Adaptive Network Evolution from Empirical Data

Hiroki Sayama

Collective Dynamics of Complex Systems Research Group  
Departments of Bioengineering & Systems Science and Industrial Engineering  
Binghamton University, State University of New York  
P.O. Box 6000, Binghamton, NY 13902-6000, USA  
sayama@binghamton.edu

**Abstract.** An algorithm is proposed for automatic discovery of a set of dynamical rules that best captures both state transition and topological transformation in the empirical data showing time evolution of adaptive networks. Graph rewriting systems are used as the basic model framework to represent state transition and topological transformation simultaneously. Network evolution is formulated in two phases: extraction and replacement of subnetworks. For each phase, multiple methods of rule discovery are proposed and will be explored. This paper reports the basic architecture of the algorithm, as well as its implementation and evaluation plan.

**Keywords:** Adaptive networks, automatic rule discovery, graph rewriting systems, generative network automata, algorithm.

## 1 Introduction

Modeling and predicting state-topology coevolution in adaptive networks is now becoming well recognized as one of the most significant challenges in complex network research [1-3]. To provide a novel framework for modeling adaptive network dynamics, I proposed to use graph rewriting systems [4,5] as a means of uniform representation of state-topology coevolution. This framework, called Generative Network Automata (GNA), is among the first to systematically integrate graph rewritings in the representation and computation of complex network dynamics that involve both state transition and topological transformation. However, it has remained an open question how one could derive a rule set of a GNA-based model from empirical data of network evolution.

Here I propose an algorithm that automatically discovers a set of dynamical rules that best captures state transition and topological transformation expressed in the empirical data. Network evolution is formulated using the GNA framework and the subnetwork extraction and replacement phases are analyzed separately. Multiple methods are proposed and will be tested for each phase. This paper reports the basic architecture of the algorithm, as well as its implementation and evaluation plan.

## 2 Generative Network Automata

The theoretical framework used in this paper is Generative Network Automata (GNA) [4,5]. Its working definition is described below.

*Configuration:* In the GNA framework, a network consists of dynamical nodes and directed links between them. Undirected links can also be represented by a pair of directed links symmetrically placed between nodes. Each node takes one of the (finitely or infinitely many) possible states defined by a node state set  $S$ . The links describe referential relationships between the nodes, specifying how the nodes affect each other in state transition and topological transformation. Each link may also take one of the possible states in a link state set  $S'$  (not considered within the scope of this paper). A configuration of a GNA at time  $t$  is a combination of states and topologies of the network. Formally, it is defined as  $G_t = \langle V_t, C_t, L_t \rangle$ , where:

- $V_t$ : A finite set of nodes of the network at time  $t$ .
- $C_t : V_t \rightarrow S$ : Node states at time  $t$ .
- $L_t : V_t \rightarrow \{V_t \times S'\}^*$ : Links and their states at time  $t$ . This maps each node to a list of destinations of outgoing links and the states of those links.

*Dynamics:* States and topologies of a GNA are updated through repetitive GNA rewriting events, each of which consists of the following three steps:

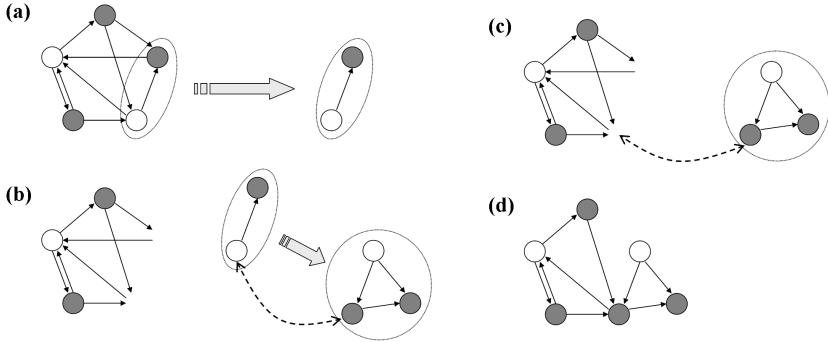
1. Extraction of part of the GNA (subGNA) that will be subject to change.
2. Production of a new subGNA that will replace the subGNA selected above.
3. Embedding of the new subGNA into the rest of the whole GNA.

The GNA evolution model can be formally defined by the following triplet  $\langle E, R, I \rangle$ :

- $E$ : An extraction mechanism that determines which part of the GNA is selected for the updating. It is defined as a function that takes the whole GNA configuration and returns a specific subGNA in it to be replaced.
- $R$ : A replacement mechanism that produces a new subGNA from the subGNA selected by  $E$  and also specifies the correspondence of nodes between the old and new subGNAs. It is defined as a function that takes a subGNA configuration and returns a pair of a new subGNA configuration and a mapping between nodes in the old subGNA and nodes in the new subGNA.
- $I$ : An initial configuration of the GNA.

The above  $\langle E, R, I \rangle$  triplet is sufficient to uniquely define a specific GNA evolution model in this framework. Figure 1 illustrates how these mechanisms work together in a rewriting event.

The function of the extraction and replacement mechanisms ( $E$  and  $R$ ) may be defined as either deterministic or stochastic, as opposed to typical deterministic graph grammatical systems [6]. A stochastic representation of GNA dynamics will be particularly useful when applied to the modeling of real-world complex network data,



**Fig. 1.** GNA rewriting process (from [4,5]). (a) The extraction mechanism  $E$  selects part of the GNA. (b) The replacement mechanism  $R$  produces a new subGNA as a replacement of the old subGNA and also specifies the correspondence of nodes between old and new subGNAs (dashed line). This process may involve both state transition of nodes and transformation of topologies. The “bridge” links that used to exist between the old subGNA and the rest of the GNA remain unconnected and open. (c) The new subGNA produced by  $R$  is embedded into the rest of the GNA according to the node correspondence also specified by  $R$ . In this particular example, the top gray node in the old subGNA has no corresponding node in the new subGNA, so the bridge links that were connected to that node will be removed. (d) The updated configuration after this rewriting event.

in which a considerable amount of random fluctuations and observation errors are inevitable.

Also, the GNA framework is unique in that the mechanism of subGNA extraction is explicitly described in the formalism as an algorithm  $E$ , not implicitly assumed outside the replacement rules like what other graph rewriting systems typically adopt (e.g., [7]). This algorithmic specification makes global rewriting events possible (as well as local rewriting ones) and allows more flexibility in representing diverse network evolution and less computational complexity in implementing their simulations.

### 3 Proposed Algorithm

In this section, I describe the proposed algorithm for automatic discovery of rewriting rules from network evolution data. Within the scope of this paper, I will simplify the problem by requiring the data to satisfy the following:

1. A given data set is a series of configurations of labeled directed networks in which labels (states) and topologies coevolve over discrete time steps (Fig. 2 (a)).
2. The data set contains information about the correspondence of nodes between every pair of two successive time points (Fig. 2 (a)).
3. States are discrete, finite, and assigned only to nodes, not to links.

4. Changes that take place between successive time points are reasonably small so that they can be identified as one small network rewriting event per each time step.
5. The extraction mechanism  $E$  and the replacement mechanism  $R$  are memoryless, i.e., they produce outputs solely based on inputs given to them.

Here I note that the GNA framework has a significant advantage for the algorithm design. It formulates the network evolution using two separate phases, i.e., the extraction of subGNA (performed by  $E$ ) and its replacement (performed by  $R$ ). Therefore, the estimation and construction of models of  $E$  and  $R$  can be conducted independently and concurrently using separate training data sets, which will make the algorithm simple and tractable.

A general procedure of the proposed algorithm is explained below (Fig. 2).

- (1) *Preprocess the original network evolution data using data-dependent heuristics, if necessary, so that they meet all the aforementioned requirements.*
- (2) *Detect the difference between each pair of configurations at two successive time points ( $G_t, G_{t+1}$ ) and represent it as a rewriting event  $s_t \equiv r_t$  (Fig. 2 (b)), where  $s_t$  is a subGNA to be replaced,  $r_t$  is another subGNA that replaces  $s_t$ , and “ $\equiv$ ” denotes correspondence from nodes in  $s_t$  to nodes in  $r_t$ .*

The difference between two configurations ( $G_t = \langle V_t, C_t, L_t \rangle$ ,  $G_{t+1} = \langle V_{t+1}, C_{t+1}, L_{t+1} \rangle$ ) will be detected in the following way:

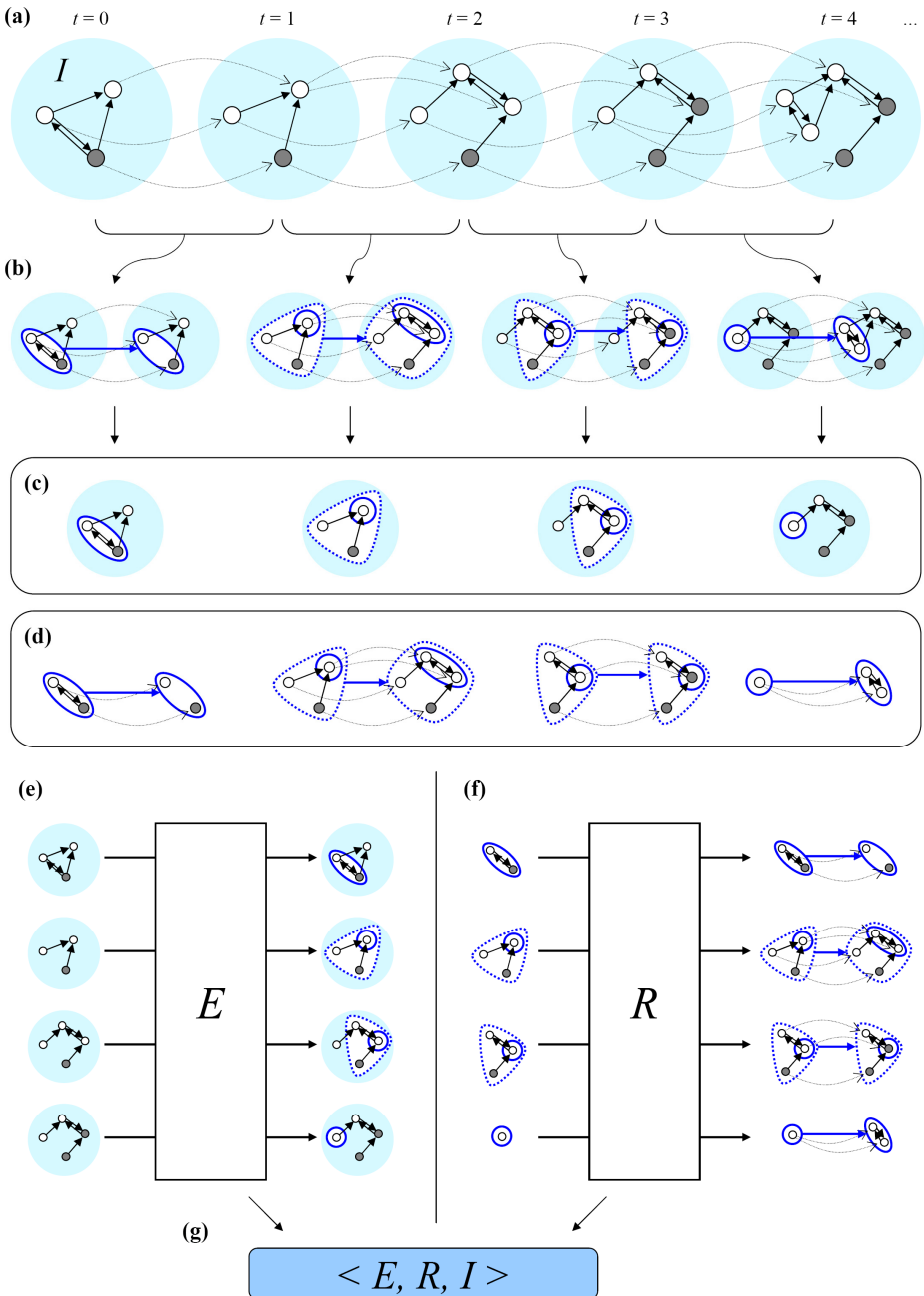
- i. Let  $A$  be a set of nodes in  $G_t$  which disappeared in  $G_{t+1}$  ( $A = \{x \mid x \in V_t \wedge x \notin V_{t+1}\}$ ).
- ii. Let  $B$  be a set of nodes in  $G_{t+1}$  which did not exist in  $G_t$  ( $B = \{x \mid x \in V_{t+1} \wedge x \notin V_t\}$ ).
- iii. Add to  $A$  and  $B$  all the nodes whose states or neighbors changed between  $G_t$  and  $G_{t+1}$   
 $(D = \{x \mid x \in V_t \wedge x \in V_{t+1} \wedge (C_t(x) \neq C_{t+1}(x) \vee L_t(x) \neq L_{t+1}(x))\}$ ,  
 $A = A \cup D, B = B \cup D$ ).

At this point,  $A$  and  $B$  contain the nodes that experienced some changes (enclosed by solid lines in Fig. 2 (b)).

- iv. Add to  $A$  and  $B$  all the nodes which have a link to any of the nodes in  $A$  ( $D' = \{x \mid x \in V_t \wedge x \in V_{t+1} \wedge L_t(x) \cap A \neq \{\emptyset\}\}$ ,  $A = A \cup D', B = B \cup D'$ ).

The above step includes in  $A$  and  $B$  additional nodes that may have influenced the rewriting event (enclosed by dashed lines in Fig. 2 (b)).

- v. Let  $s_t$  and  $r_t$  be subgraphs of  $G_t$  and  $G_{t+1}$  induced by nodes in  $A$  and  $B$ , respectively.



**Fig. 2.** Overview of the proposed algorithm. (a) Original network evolution data starting with the initial configuration  $I$ . (b) Detection of rewriting events at every time step. (c) Training data for the extraction mechanism  $E$ . (d) Training data for the replacement mechanism  $R$ . (e, f) Construction of models of  $E$  and  $R$  based on the training data. (g) Final GNA model.

Then the detected rewriting event is represented as  $s_i \Rightarrow r_i$ , where “ $\Rightarrow$ ” is the set of all the node correspondences between  $s_i$  and  $r_i$  present in the original data.

- (3) *Construct a model of the extraction mechanism  $E$  by using  $\{ (G_i, s_i) \}$  as training data, where  $G_i$  is the input given to  $E$  and  $s_i$  the output that  $E$  should produce (Fig. 2 (c), (e)).*

This step is the most challenging part in this algorithm development effort. The task to be achieved in this step is to identify an unknown mechanism that chooses a subset of a given set of nodes. Exact identification of an unknown computational mechanism is theoretically not possible in general. Therefore, I will test several heuristic approaches as candidates of practical solutions, including:

i. *Statistical analysis of node properties*

I will first test a simple statistical approach, where correlations will be measured, for the whole training data, between several node properties and the probability of a node to be selected for rewriting. Node properties to be used will include state, degree (absolute or relative) and local clustering coefficient (absolute or relative), among others. If a clear correlation is found, it will be used as the mechanism of  $E$ .

ii. *Statistical selection from multiple mechanisms*

In the second approach, I will assume several predefined candidate mechanisms (e.g., random selection, preferential selection based on node degrees, etc.) and calculate the probability for each extraction result given in the training data to occur with each candidate mechanism. If a mechanism includes parameters, they will be optimized to attain the maximal probability. This calculation will be conducted and aggregated for the whole training data to evaluate how likely the given training data could result from each of the candidate mechanisms. Then the mechanism with highest likelihood will be returned as the estimated mechanism of  $E$ .

iii. *Parameter estimation for a preprogrammed mechanism*

In some situations the actual extraction mechanism  $E$  may be known to the researcher. If this is the case, the mechanism can be preprogrammed in detail, with several parameters left unspecified to give room for fitting to the training data. Then I will use the same probability calculation method as in the second approach to optimize the parameters.

iv. *Evolutionary search with Genetic Programming*

I will also test an evolutionary approach, where possible mechanisms of  $E$  will be represented by short pseudo codes and be evolved using Genetic Programming techniques.

- (4) *Construct a model of the replacement mechanism  $R$  by using  $\{ (s_i, s_i \Rightarrow r_i) \}$  as training data, where  $s_i$  is the input given to  $R$  and  $s_i \Rightarrow r_i$  the output  $R$  should produce (Fig. 2 (d), (f)).*

In this step, the task can be achieved in a much simpler manner than in step (3), though technically it still remains identification of an unknown mechanism. This is because a single rewriting event typically involves just a few nodes so the number of possible inputs given to the replacement mechanism  $R$  is virtually finite in contrast to the number of possible inputs to  $E$  that is virtually infinite. Therefore I will use straightforward pattern matching methods to construct a model of  $R$  from the data.

Specifically, the algorithm will construct  $R$  as a simple procedure that searches for a rewriting event in the training data whose left hand side matches the given input. If there is only one such event found, the event itself will be the output of  $R$ . If multiple events are found, the output will be determined either deterministically (e.g., event with greatest frequency) or stochastically (e.g., random selection with weights set proportional to event frequencies). Or, if no event is found, either identity (“input  $\Rightarrow$  input”; no change) will be returned or seek similar events will be sought using partial graph matching schemes.

- (5) *Construct a complete GNA model by combining the results of the above steps (3) and (4) together with the initial configuration  $I$  (Fig. 2 (g)).*

## 4 Summary and Future Work

In this paper, I proposed an algorithm for automatic rule discovery of adaptive network evolution from empirical data, and described its outline as well as multiple candidate methods for some of its components. The implementation of the algorithm is currently ongoing in Python, partly based on the existing NetworkX module [8]. The completed implementation will be made freely available to researchers and other professionals under an open-source license.

The implemented algorithm will be evaluated firstly via application to abstract data generated by artificial GNA models used in my preliminary study [4,5]. The algorithm will be applied to several sample simulation runs selected from those data to check if they could correctly recover the actual network rewriting rules used to produce the data. In this testing, the correct answers are already known, so it will be possible to evaluate the success/failure ratio of each algorithm implementation and revise it to improve the accuracy of its outputs. Several additional complexities will also be introduced into the abstract network models, including more than two states on nodes, more subGNA rewriting options, and simultaneous application of multiple rewriting events, to test and revise the algorithm under model variations. After this initial evaluation is completed, I will also plan to test the algorithm by applying it to real-world network evolution data (such as [9,10]).

As the research progresses, the entire architecture or the details of each step may be revised as needed. At later stages of the project, expansions of the algorithm to broader classes of complex adaptive network dynamics will also be considered. Specifically, I plan to investigate how to incorporate continuous states, how to incorporate links with states, and how to handle the possibility for  $E$  or  $R$  to have its own internal memory.

**Acknowledgments.** This material is based upon work supported by the National Science Foundation under Grant No. 1027752.

## References

1. Albert, R., Barabási, A.-L.: Statistical mechanics of complex networks. *Rev. Mod. Phys.* 74, 47–97 (2002)
2. Gross, T., Blasius, B.: Adaptive coevolutionary networks: a review. *J. R. Soc. Interface* 5, 259–271 (2008)
3. Gross, T., Sayama, H. (eds.): *Adaptive Networks: Theory, Models and Applications*. Springer (2009)
4. Sayama, H.: Generative network automata: A generalized framework for modeling complex dynamical systems with autonomously varying topologies. In: *Proceedings of The First IEEE Symposium on Artificial Life*, pp. 214–221. IEEE (2007)
5. Sayama, H., Laramée, C.: Generative network automata: A generalized framework for modeling adaptive network dynamics using graph rewritings. In: Gross, T., Sayama, H. (eds.) *Adaptive Networks: Theory, Models and Applications*, pp. 311–332. Springer (2009)
6. Rozenberg, G. (ed.): *Handbook of Graph Grammars and Computing by Graph Transformation, Foundations*, vol. 1. World Scientific (1997)
7. Kurth, W., Kniemeyer, O., Buck-Sorlin, G.H.: Relational Growth Grammars – A Graph Rewriting Approach to Dynamical Systems with a Dynamical Structure. In: Banâtre, J.-P., Fradet, P., Giavitto, J.-L., Michel, O. (eds.) *UPP 2004. LNCS*, vol. 3566, pp. 56–72. Springer, Heidelberg (2005)
8. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using NetworkX. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) *Proceedings of the 7th Python in Science Conference (SciPy 2008)*, pp. 11–15 (2008), NetworkX, <http://networkx.lanl.gov>
9. Fowler, J.H., Jeon, S.: The authority of Supreme Court precedent. *Social Networks* 30, 16–30 (2008)
10. Fowler, J.H., Jeon, S.: Supreme Court Citation Network Data website, <http://jhfwolwer.ucsd.edu/judicial.htm>