FINDING STEADY STATES OF LARGE SCALE REGULATORY NETWORKS THROUGH PARTITIONING

Ferhat Ay, Gunhan Gulsoy, Tamer Kahveci

Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611

{fay, ggulsoy, tamer}@cise.ufl.edu

ABSTRACT

Identifying steady states that characterize the long term outcome of regulatory networks is crucial in understanding important biological processes such as cellular differentiation. Finding all possible steady states of regulatory networks is a computationally intensive task as it suffers from state space explosion problem. Here, we propose a method for finding steady states of large-scale Boolean regulatory networks. Our method exploits scale-freeness and weak connectivity of regulatory networks in order to speed up the steady search through partitioning. In the trivial case where network has more than one component such that the components are disconnected from each other, steady states of each component are independent of those of the remaining components. Thus, it is possible to generate the steady states of the entire network from the steady states of the individual components. When the size of at least one connected component of the network is still prohibitively large, further partitioning is necessary. In this case, we identify weakly dependent components (i.e., two components that have a small number of regulations from one to the other) and calculate the steady states of each such component independently. We then combine these steady states by taking into account the regulations connecting these components. We show that this approach is much more efficient than calculating the steady states of the whole network at once when the number of edges connecting them is small. Since regulatory networks often have small in-degrees, this partitioning strategy can be used effectively in order to find their steady states. Our experimental results on real datasets demonstrate that our method leverages steady state identification to very large regulatory networks.

Index terms: Network partitioning, Steady states, Regulatory networks

1. INTRODUCTION

Regulatory networks (RNs) show how different genes regulate each other either by physical binding of transcription factors in promoter or enhancer regions, or by presence of evolutionary conserved sequence-specific motifs. RNs are of utmost importance in understanding functional properties of the genes of an organism such as changes in their expression levels and their process and/or function annotations. In order to extract such information from RNs it is necessary to characterize their long term behavior by analyzing the possible states of the network. The *state* of a RN shows the expression levels of all the genes in that RN. *Steady states*, states that are once reached are re-visited infinitely many times (i.e. stable or periodic expression), characterize the outcomes of a RN and has proven to be successful in explaining a number of biological processes such as cell cycle of yeast [1, 2] differentiation of T-helper cells [3] and flower morphogenesis [4].

Notation and Problem Definition: We first introduce some notation and define different types of states that will be used throughout the rest of the paper. We start with the definition of steady state:

Definition 1 *Let S be a set of states. Each $s_i \in S$ is steady if and only if the following two conditions are satisfied:*

1. *The set of the successor states of all the states in S is equal to S*
2. *For each $s_i \in S$ once it is visited the probability of revisiting s_i is equal to 1 in a finite number of state transitions.* ■

The rest of the states of the network are called *transient states* and they are usually not of interest from biological viewpoint. The above definition suggests that there are two types of possible steady states, namely self loops (e.g., Figure 1(a)) and simple loops (e.g., Figure 1(b)) as named in [1, 3]. If a set of states create a complex loop, then all the states of this set are transient (e.g., Figure 1(c)).

In our earlier work, we segregated the states into three non-overlapping types according to the number of possible out-going transitions from them as described in Figure 1. Using these state types, we observed the following relations: **(1)** Type 0 states have no outgoing edges, hence they are steady (e.g., state [110] in Figure 1(a)). **(2)** Type 2 states have at least two out going edges. Thus, they are transient (e.g., state [110] in Figure 1(c)). **(3)** All the remaining states are of Type 1. These states may be steady or transient. We refer the reader to our earlier work for further details of the state types and the segregation strategy [1].

This work was supported partially by NSF under grants CCF-0829867 and IIS-0845439.

The number of possible states of a RN is exponential in the number of genes even when we consider only Boolean states for each gene ("ON" or "OFF" meaning high or low expression). Therefore, exhaustive search of the state space is not feasible for even moderately sized networks. A number of methods have been developed to address this problem [3,5]. However, they have strong assumptions (i.e. only synchronous state transitions) that are arguable from a biological viewpoint, and/or do not scale well for large networks. In an earlier work, we proposed a randomized traversal algorithm that computes online estimates with quality guarantees for identifying the steady state profile. However, when all the steady states are queried with around 100% confidence, the running time can still be prohibitive for large networks [1].

Contributions: In this paper, we develop a computational method that can identify all the steady states of a given RN. We extend the method described in Ay *et al.* [1] in order to leverage the steady state identification problem to larger networks. Here, we introduce a partitioning strategy that exploits weak connectivity of regulatory networks. Our method provides a framework that identifies steady states independently for each partition of the network and combines them to get the list of all steady states for the complete network by taking into account the regulatory meaning of the edges connecting different partitions without losing any accuracy. Our results on real data shows that this approach provides significant improvement in running time over the original method described in Ay *et al.* [1] as well as other existing methods [3].

The organization of the rest of this paper is as follows: Section 2 provides a detailed discussion of our method. Section 3 presents the experimental results. Section 4 briefly concludes the paper.

2. METHOD

The state space of a RN grows exponentially with the number of nodes in the network. Therefore, speeding up the steady state identification process is necessary for large scale networks. For this purpose, we propose a network partitioning strategy that works in two modes, namely for independent components and for weakly dependent components (i.e. there are a small number of interactions between different components). In here, we discuss how we combine the results gathered from these components to construct overall steady states of the actual network without loss of any information.

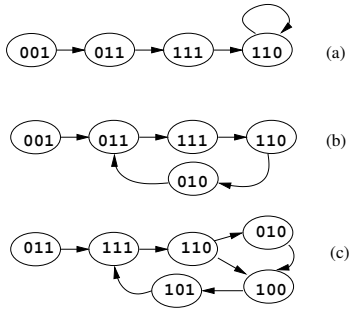

Fig. 1: Each circle represents a state of a hypothetical network with 3 genes. The binary values correspond to activation levels of these genes. (a) The three states on the left are transient and of Type 1. The state with self loop is steady and Type 0. (b) The four states in simple loop are cyclic steady states and they are of Type 1. (c) The leftmost state is transient and Type 1. Even though only [110] is of Type 2 (others are Type 1), the remaining states create a complex loop, and thus they are transient.

2.1. Combining steady states from independent components

When there are no activations or inhibitions between two or more components of the input network, the steady states of each component can be determined independent of the other components. Combining the results from different components is rather easy in this case. Assume that we have n independent components for a given RN. We know that Type 0 states have no out-going edges and Type 1 steady states have only one out-going edge which is a part of a cycle. Therefore, the combination of two Type 0 steady states from two independent components results in a Type 0 steady state for the union of these partitions. This is also true for multiple components. On the other hand, combining a Type 1 steady state of a component with a Type 0 steady state of another component we get a Type 1 steady state for the union of these two components. However, when we combine two Type 1 steady states, we end up having two out-going edges from the combined state, hence this state is not steady anymore. All the other possible combinations of different state types (e.g. Type 0 with Type 2, Type 1 with Type 2 and Type 2 with Type 2) result in transient states.

Here, we formulate the number of steady states for the overall RN in terms of the numbers of steady states of independent components. Let, $\Theta_1, \Theta_2, \cdots, \Theta_n$ and $\Omega_1, \Omega_2, \cdots, \Omega_n$ denote the number of Type 0 and Type 1 steady states in each partition. Using the above observations on combining the steady states, the total number of Type 0 and Type 1 steady states (Θ and Ω respectively) of the actual network is: $\Theta = \prod_{i=1}^{n} \Theta_i$ and $\Omega = \sum_{i=1}^{n} [\Omega_i \prod_{i \neq j} \Theta_j]$.

2.2. Combining steady states from weakly dependent components

When the above partitioning fails to divide the network in small enough pieces, we use a more complex strategy to for partitioning. We first identify the two components of the network that have the least number of interactions connecting each other among all other possible partitionings. When necessary, this division can be repeated recursively for either one or both of the components. For ease of discussion, we first focus on a network that is divided into two components which are connected by a set of interactions from a single gene in one component (see Figure 2). The discussion can be generalized for components connected by more than one genes, however, the complexity of combining the steady states from two components increases

exponentially with the number of genes connecting them. Therefore, we use partitioning of this type for only weakly dependent components. The good news is that, the weakly dependent subnetworks are commonly observed in real RN data. The average number of interactions per gene is 1.05 for the real networks we use in our experiments. We discuss the applicability of partitioning on these networks in Section 3.

Let us turn our attention back to the case with two components connected by a single gene. Figure 2 displays this scenario. Here, P and R denote two components that weakly depend on each other only through a set of out-going edges from gene A_p. For simplicity, we will focus on the interaction that shows that A_p activates gene B_r and ignore the other genes in R interacting with A_p. We can deal with all such genes similarly. We first define a new component R' such that $R' = R \cup \{A_p\}$. Then, we identify all the steady states of each of these three components independently. The tricky part is combining the steady states of these components without missing any of the steady states of the overall network. For this we pick one steady state, say u_P, from the solution space of component P and depending on the type of u_P we have three different cases to consider:

- *Case1 :* (u_P is of Type 0) In this case, each steady state of component R' can be combined with u_P to create a combined steady state. One thing that we need to be careful is that, since gene A_p is present in both component P and component R', when combining if A_p is active in state u_P we combine u_P with only the steady states of R' that have gene A_p in active state. The case when A_p is inactive is combined similarly.
- *Case2 :* (u_P is of Type 1 and state of A_p is fixed in this cycle) If gene A_p is inactive in u_P, we combine it with all the steady states of component R which are of Type 0. This is due to the fact that, when A_p is always inactive in a cyclic steady state of P, the activation between A_p and B_r has no effect on the states of B and two components act independently. In the case when A_p is always active, the combining procedure is similar to the active case described in Case 1.
- *Case3 :* (u_P is of Type 1 and state of A_p is not fixed in this cycle) If gene A_p changes activity level (oscillates) in a steady state cycle with state u_P, then we can only combine u_P with the Type 0 steady states of partition R in which gene A_p has *no effect* on the state of gene B_r. The only case when A_p can have a state changing effect on B_r is when all the other activators and all inhibitors of gene B_r are in active state. Hence, we combine u_P with only the steady states of R in which at least one activator or an inhibitor of B_r is in active state.

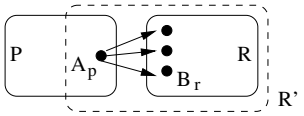

Fig. 2: A hypothetical RN that has two weakly connected components, namely P and R which are connected by gene A_p. R' is a new component that is defined as $R \cup A_p$ which will be used to combine the steady states of P and R.

Two important points worth discussing are as follows: (1) Once we partition a given RN into two components, one or both of the components may still be too large to find its steady states. In that case we can partition such components recursively into smaller components until each component is small enough. (2) It is possible that there is no single gene (such as A_p in Figure 2) whose removal decomposes the given RN into two components. In that case we may partition the RN by removing two or more genes. Let n be the number of such genes. The cost of combining the steady states from P and R' increases exponentially with n. Particularly, the number of cases described above is $1+2^n$ in the worst case (proof omitted).

We observed that recursive application of the different strategies discussed above allows us to find all the steady states of large RNs in the order of seconds. Next, we discuss the running time improvement over existing algorithm achieved by the method we describe here.

3. RESULTS

In this section, we compare the running time of the method we develop here to that of Genysis [3] and our earlier algorithm that does not partition the RN [1]. We report the results for a number of real RNs taken from published literature [3, 6] and Pathway Interaction Database (PID) [7].

Table 1 summarizes the results for RNs of different sizes. For all of these RNs Genysis and our earlier method takes significant amount of time. For most of the RNs, they do not complete in over 5 hours. The RNs in the first three rows are moderately sized. For these RNs one step of partitioning suffices to identify all the steady states in the order of seconds. We obtained these partitionings by removing only one or two edges. Hence, combining the steady states can be done in constant time. We observe that there is a dramatic decrease in running time (from hours to seconds) and it is due to only one step application of the partitioning scheme we develop here.

In order to see the merits of recursive application of partitioning, we also used large scale networks with more than 100 genes. Fourth network exemplifies a case where only one step of partitioning is not sufficient to divide the RN into small enough components to decrease the running time into seconds. However, when we apply a second step of partitioning on the largest component (i.e. 38 nodes), we can solve the steady state problem in less than a minute. The fifth network, taken from Garg *et al.* [3], has a connected component of 103 genes. Removing two edges from this component brought us to two components with 63 and 40 genes respectively. After removing two more edges from the component with 63 gene we further decomposed

ID	Network Name	Nodes	Edges	*Genysis*	No partitioning Ay *et al.*		First phase of partitioning		Second phase of partitioning	
					Sizes [1]	Time [2]	Sizes	Time	Sizes	Time
1	p38-MAPK [7]	26	28	$>$ 5hrs	26	837	17-9 †	26	-	-
2	T-CellReceptor [3]	40	58	1,242	40	$>$ 5hrs	21-19 ‡	13	-	-
3	hsa-Insulin.txt [7]	62	53	$>$ 5hrs	41-6	$>$ 5hrs	20-21 †	45	-	-
4	E.coli RN [6]	150	119	$>$ 5hrs	75-5	$>$ 5hrs	38-12 §	$>$ 5hrs	17-12 §	51
5	network-DC [3]	111	117	$>$ 5hrs	103	$>$ 5hrs	63-40 ‡	$>$ 5hrs	40^a-37^b-26^c ‡	$>$ 5hrs
5a	(a)	40	40				20-17 ‡	48	-	-
5b	(b)	37	38				17-17 ‡	42	-	-
5c	(c)	26	30				13-13 ‡	12	-	-

Table 1: Running time comparison of the method we propose here versus two existing algorithms Genysis [3] and Ay *et al.* [1]. We apply the algorithm described in Ay *et al.* to find the steady states of each component after partitioning. A third phase of partitioning is done for the three largest connected components (a,b,c) of the RN with ID 5. Last three rows are dedicated to the results of this partitioning for each of these components.

[1] Number of nodes of the three largest connected components for the input RN (components of size less than 5 are omitted).

[2] Running time in terms of seconds. $>$ 5hrs denotes that the method could not find all steady states within this time.

− Further partitioning is not done when the running time drops below one minute. † Partitioning is achieved by only removing one edge.‡ Partitioning is achieved by removing two edges. § Partitioning is achieved by removing a vertex with all its outgoing edges.

the RN into three components with 40, 37 and 26 genes. We, then, listed each of these components as separate rows in Table 1 (5a, 5b and 5c). We applied one last step of partitioning on each component which decreased the running time to under one minute for each component. The combining phase for these components is a three step bottom-up approach starting from the steady states of components (a), (b) and (c) and ending at the all the steady states of the original network that we started with.

The above results showed that the method we describe here is significantly faster than Genysis *and our earlier work with no partitioning. Furthermore, we proved that we can combine the steady states of the partitioned RN without loss of any steady states or introducing any false steady states such that we get the true set of steady states for the original network in a much shorter time than the other methods discussed. Therefore, we believe that our method can be used for accurate identification of all the steady states for even the large scale RNs.*

4. CONCLUSION

In this paper, we developed a computational method that can identify all the steady states of Boolean regulatory networks accurately and efficiently. By utilizing weak connectivity of RNs, we devised a partitioning scheme that provides significant improvement over existing algorithms in terms of scaling the steady state identification for large RNs. We proposed a number of partitioning modes ranging from independent components to removing one or two edges or nodes from the input network. For the challenging problem of combining the steady states of different components, we argued that this combination can be done without losing accuracy in terms of the overall steady states of the RN. We observed that combining this partitioning scheme with an earlier algorithm that we developed for efficient filtering of transient states, is significantly more efficient than calculating the steady states of the whole network at once. Our experimental results on real datasets showed that our method leverages the steady state identification problem to networks with more than 100 nodes and edges.

5. REFERENCES

[1] F. Ay, F. Xu, and T. Kahveci, "Scalable steady state analysis of boolean biological regulatory networks," *PLoS ONE*, vol. 4, no. 12, pp. e7992, 2009.

[2] MI. Davidich and S. Bornholdt, "Boolean network model predicts cell cycle sequence of fission yeast.," *PLoS ONE*, vol. 3, no. 27, pp. e1672, 2 2008.

[3] A. Garg, I. Xenarios, L. Mendoza, and G. De Micheli, "An efficient method for dynamic analysis of gene regulatory networks and *in silico* gene perturbation experiments.," in *Conf Proc Res Comput Mol Biol*, 2007, pp. 62–76.

[4] ER. lvarez Buylla, . Chaos, M. Aldana, M. Bentez, Y. Cortes-Poza, C. Espinosa-Soto, DA. Hartasnchez, RB. Lotto, D. Malkin, GJ. Escalera-Santos, and P. Padilla-Longoria, "Floral morphogenesis: stochastic explorations of a gene network epigenetic landscape.," *PLoS ONE*, vol. 3, no. 11, pp. e3626, 11 2008.

[5] MA. Schaub, TA. Henzinger, and J. Fisher, "Qualitative networks: A symbolic approach to analyze biological signaling networks," *BMC Syst Biol*, vol. 1, pp. 4, 2007.

[6] R. Jothi, S. Balaji, A. Wuster, J. Grochow, JA. andGsponer, TM. Przytycka, L. Aravind, and MM. Babu, "Genomic analysis reveals a tight link between transcription factor dynamics and regulatory network architecture.," *Mol Syst Biol.*, vol. 5, no. 12, pp. 294, 2009.

[7] CF. Schaefer, K. Anthony, S. Krupa, J. Buchoff, M. Day, T. Hannay, and KH. Buetow, "PID: The Pathway Interaction Database.," *Nucleic Acids Res*, vol. 37, pp. 674–679, 2009.