Theoretical Computer Science 679 (2017) 126–144

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Inferring local transition functions of discrete dynamical systems from observations of system behavior

Abhijin Adiga [a], Chris J. Kuhlman [a], Madhav V. Marathe [a], S.S. Ravi [b,*], Daniel J. Rosenkrantz [b], Richard E. Stearns [b]

[a] Network Dynamics and Simulation Science Laboratory, VBI, Virginia Tech, USA
[b] Computer Science Department, University at Albany – SUNY, USA

ARTICLE INFO

Article history:
Received 30 November 2015
Received in revised form 7 June 2016
Accepted 6 July 2016
Available online 16 July 2016

Keywords:
Threshold inference
Computational complexity
Synchronous dynamical systems
Threshold functions
Phase space
Trajectory
Stable and unstable configurations
Fixed parameter tractability

ABSTRACT

We consider the problem of inferring the local transition functions of discrete dynamical systems from observed behavior. Our focus is on synchronous systems whose local transition functions are threshold functions. We assume that the topology of the system is known and that the goal is to infer a threshold value for each node so that the system produces the observed behavior. We show that some of these inference problems are efficiently solvable while others are **NP**-complete, even when the underlying graph of the dynamical system is a simple path. We identify a fixed parameter tractable problem in this context. We also consider constrained versions of threshold inference problems where the input includes a set of equality or inequality constraints (which specify pairs of nodes which must have the same threshold value or different threshold values). We present algorithmic and complexity results for several constrained threshold inference problems.

© 2016 Elsevier B.V. All rights reserved.

1. Introduction

1.1. Motivation

Many studies have demonstrated the efficacy of using discrete dynamical systems to model population dynamics. Among these are studies of incarceration [1], epidemics and infectious diseases [2,3], and information spread through blogs [4]. It has also been demonstrated that dynamical system models can produce dramatically different results, depending on the parameter values used. For example, in certain disease models, small changes in infection and recovery rates, as quantified by the epidemic threshold, can change the size of an outbreak from large to small [5]. As another example, in complex contagion models, small changes in threshold values can produce large changes in contagion spread [6,7]. These considerations are among the issues within the realm of model verification and validation.

Methods that use observations of systems to calibrate or infer models and model properties arise in many contexts. These properties help explain system behaviors, and parameterized models may be transferable to other contexts [8]. A case in point is a protest in Spain in 2011, in which people demonstrated against economic austerity measures [9]. In that work on information spread via Twitter, threshold behavior was used to model tweeting about the protest. If a person v received

* Corresponding author.
E-mail addresses: abhijin@vbi.vt.edu (A. Adiga), ckuhlman@vbi.vt.edu (C.J. Kuhlman), mmarathe@vbi.vt.edu (M.V. Marathe), ssravi0@gmail.com (S.S. Ravi), drosenkrantz@gmail.com (D.J. Rosenkrantz), thestearns2@gmail.com (R.E. Stearns).

http://dx.doi.org/10.1016/j.tcs.2016.07.008
0304-3975/© 2016 Elsevier B.V. All rights reserved.

t_v tweets (from unique users that she followed) about the event, and then tweeted for the first time before v received the $(t_v + 1)$th tweet, that person's threshold for participation was chosen as t_v. The inference of thresholds enabled the full specification of a model for the underlying socio-technical phenomenon. Several other studies along the same lines have also been carried out (e.g., [10,11]). The threshold model for social networks, introduced in [6], is used in many applications (see e.g., [12]). Discrete dynamical systems [13,14], which generalize cellular automata, represent a rigorous and convenient abstract model to study socio-technical phenomena.

In experimental evaluations of contagion processes, different subpopulations behave differently, suggesting that these subgroups have different thresholds. There is evidence for differences in contagion transmission based on such factors as ethnicity, gender, and academic standing. For example, in [15], it is shown that the probability of students dropping out of school is a function of ethnicity. Since dropping out of school can be modeled as a contagion [15,16], this suggests that a threshold model of dropping out would assign different thresholds to different ethnic groups. Similarly, graduate students have a much higher drop out rate than other students at selective institutions [17], indicating that threshold-based contagion models of dropping out would assign a lesser threshold (to drop out) to graduate students than to undergraduates. As a final example, treating depression as a contagion [18], women are more likely to become depressed from their friends' depression than men [19]. This implies that for a threshold model of depression, females would have a lesser threshold than males. These studies motivate the use of different thresholds for different sets of individuals.

The problem of inferring the components of a system from observed behavior has also received a lot of attention in the theoretical computer science literature. For example, many researchers have studied the problem of learning automata from sets of accepted strings (see e.g., [20–22]). Likewise, the problem of learning CNF and DNF formulas has also been studied extensively in the literature (e.g., [23]). Additional information on these topics will be provided in Section 1.3.

1.2. Problems considered

We consider inference problems that arise in the context of discrete dynamical systems. In particular, we focus on one such model, namely *synchronous* discrete dynamical systems (SyDSs). We provide an informal description of a SyDS here; a formal description is given later. A SyDS consists of an undirected graph whose vertices represent entities (agents) and edges represent local interactions among entities. Each vertex has a state value chosen from a finite domain (e.g., $\{0, 1\}$). In addition, each vertex v also has a local transition function whose inputs are the current state of v and those of its neighbors; the output of this function is the next state of v. The vector consisting of the state values of all the nodes at each time instant is referred to as the **configuration** of the system at that instant. In each time step, all nodes of a SyDS compute and update their states *synchronously*. Starting from a (given) initial configuration, the time evolution of a SyDS consists of a sequence of successive configurations. Models similar to SyDSs have been used in several applications, including the propagation of diseases and social phenomena (e.g., [12,24]).

Several researchers have studied the complexity of various **analysis problems** for discrete dynamical systems (e.g., [25–29]). Informally, the goal of an analysis problem is to predict the behavior of a system given its static description. An example of an analysis problem is that of **reachability**: given a SyDS $\mathcal{S}$ and two configurations $\mathcal{C}_1$ and $\mathcal{C}_2$, will $\mathcal{S}$ reach $\mathcal{C}_2$ starting from $\mathcal{C}_1$? Such analysis questions are studied by considering the **phase space** of the SyDS, which is a directed graph with one vertex for each possible configuration and a directed edge (x, y) from a vertex x to vertex y if the SyDS can transition from the configuration corresponding to x to the one corresponding to y in one time step. In such a case, y is the **successor** of x and x is a **predecessor** of y. Each self loop in the phase space of a SyDS represents a **stable** configuration or a **fixed point** of the system[1] (i.e., a configuration in which the system will stay forever). Any configuration $\mathcal{C}$ whose successor is different from $\mathcal{C}$ is called an **unstable configuration**. A **trajectory** in phase space is a directed path with one or more edges. Also, any vertex in the phase space with no incoming edges represents a **Garden-of-Eden** (GE) configuration which can arise only as the initial configuration of the system.

Here, our focus is on a problem which may be considered as an *inverse* of the analysis problem. In such a problem, the goal is to infer some aspect of the structure of a partially specified system given a description of its observed behavior. In particular, we focus on inferring the local transition functions of a SyDS, given the underlying graph and appropriate behavior patterns. We consider several different behavior patterns (e.g., collection of stable configurations, collection of trajectories, collection of unstable configurations) and study the complexity of identifying the local transition functions that can explain the observed behavior. In particular, we assume that each local transition function f_i is a t_i-threshold function for some non-negative integer t_i (see Section 2 for the definition of threshold functions).

We present results for threshold inference problems for two categories of observed behavior, namely homogeneous collections (e.g., a set of stable configurations, a set of unstable configurations) and heterogeneous collections (e.g., a collection consisting of a set of stable configurations and a set of unstable configurations). Our results establish the complexity of the corresponding decision and optimization problems (see Sections 3 and 4). Motivated by the considerations mentioned in Section 1.1, we also present algorithmic and complexity results for constrained versions of threshold inference problems (Section 5).

[1] We will use the term "stable configuration" instead of "fixed point" throughout this paper since we use the word "fixed" in the context of fixed parameter tractability.

SyDSs are generalizations of cellular automata (CA) since the underlying graphs of SyDSs can be arbitrary while those of CA are generally restricted to geometric structures such as lines and lattices. Many of the results presented in this paper apply directly to CA with threshold functions at the nodes. For example, since all our efficient algorithms for threshold inference for SyDSs hold for arbitrary graphs, they also hold for CA. Further, our hardness results in Sections 3 and 4 which hold for SyDSs when the underlying graph is a simple path can be thought of as results for 1-dimensional CA. Other problems where hardness results for discrete dynamical systems directly imply similar results for CA are discussed in [27].

1.3. Related work

Several inference problems have been explored in the context of disease, information and meme spread. One direction of work considers estimating model parameters given the traces of a diffusion process, network and a class of models (e.g., Bailon et al. [9]). Abraho et al. [30], Gomez et al. [31], Soundarajan and Hopcroft [32] consider the problem of inferring the network structure given the model. Recently, there has been a lot of work on source detection, where the goal is to find the source of infection given limited information about the network, diffusion model and the set of infected nodes (e.g., [33]). Most of the these problems turn out to be hard even for simple models (such as progressive systems [34]) and networks.

Learning finite automata [20,21] and Boolean functions [23] are two rich areas which consider problems with a similar flavor. In the case of learning finite automata, the general problem is to infer a finite (stochastic) automaton given a set of strings labeled as either in the language or not [20–22]. Similarly, in concept learning (or learning Boolean functions), the general task is to infer a Boolean function given information about its values for some inputs, together with the knowledge that it belongs to a particular class of functions [23].

As mentioned earlier, many researchers have addressed various questions regarding phase space properties of discrete dynamical systems. Goles and Martínez [35] provide bounds on the lengths of transients (i.e., trajectories that end in stable configurations) and cycles in threshold dynamical systems. The complexity of determining whether a given configuration y of a deterministic SyDS has a predecessor has been studied in [25]. Problems similar to predecessor existence have also been considered in the context of CA [26,36]. Researchers have also studied various questions for dynamical systems under the sequential update model, where the vertex functions are applied according to a specified order [37,13,28].

Many researchers have explored dynamics on networks. For example, variants of the influence maximization problem initially proposed in [38] are studied in [39], and several fixed parameter tractability results are established. The idea of “harmless individuals” (nodes) in a social network is investigated in [40]. Formally, a set S of nodes is harmless if every node v of the input graph has less than its threshold number of neighbors in S. Hence, a contagion where each initially infected node is in S cannot spread through the network. This is also related to the target set selection problem (see [40]). The firefighting problem, where a maximum number of blocking nodes is specified for each time step to stop the contagion spread, is studied in [41].

Several problems in graph theory, often referred to as re-configuration or re-optimization problems, are related to those we study here. These problem formulations rely on suitable definitions of distances between a pair of instances of a problem and a pair of solutions. Given two problem instances which are within a specified distance and a solution to the first instance, the goal is to determine a solution to the second instance that is within a specified distance of the first solution. In this context, two fixed parameter tractable problems—one for dominating sets and one for vertex covers—are provided in [42]. Other problems have also been solved under essentially this same setting (e.g., [43,44]). Dominating sets, alliances, and a generalizing framework are discussed in [45]. Łacki et al. [46] provide algorithms for maintaining approximations of a minimum Steiner tree of a predefined graph when the vertices of the graph may be deleted or (re)inserted in time. A survey of problems where the goal is to transform one solution into another by a series of intermediate steps, each of which is also a solution, is given in [47].

Beyond the graph theoretic problems discussed above (e.g., [42–44]), fixed parameter tractability and associated results have also been established in other contexts such as grammars and parsing [48,49], logic-based model checking [50], and construction of deterministic finite automata (DFA) [51]. In particular, the problem of finding a smallest deterministic finite automaton consistent with a set of positive and negative examples of the intended language, addressed in [51], has some similarity with our work.

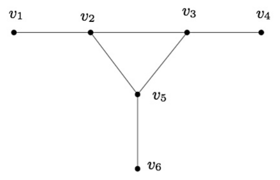
2. Definitions, problem formulations, and summary of results

2.1. Formal definition of the SyDS model

Let $\mathbb{B}$ denote the Boolean domain $\{0, 1\}$. A **Synchronous Dynamical System** (SyDS) $\mathcal{S}$ over $\mathbb{B}$ is specified as a pair $\mathcal{S} = (G, \mathcal{F})$, where

(a) $G(V, E)$, an undirected graph with $|V| = n$, represents the underlying graph of the SyDS, with node set V and edge set E, and

(b) $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$ is a collection of functions in the system, with f_i denoting the **local transition function** associated with node v_i, $1 \le i \le n$.



Initial Configuration:	(1, 0, 0, 1, 0, 1)
Configuration at time 1:	(1, 0, 0, 1, 1, 1)
Configuration at time 2:	(1, 0, 1, 1, 1, 1)
Configuration at time 3:	(1, 1, 1, 1, 1, 1)

Note: Each configuration has the form $(s_1^t, s_2^t, s_3^t, s_4^t, s_5^t, s_6^t)$, where s_i^t is the state of node v_i at time t, $1 \le i \le 6$. The configuration at time 3 is a stable configuration.

Fig. 1. An Example of a SyDS.

Each node of G has a state value from $\mathbb{B}$. Each function f_i specifies the local interaction between node v_i and its neighbors in G. The inputs to function f_i are the state of v_i and those of the neighbors of v_i in G; function f_i maps each combination of inputs to a value in $\mathbb{B}$. This value becomes the next state of node v_i. It is assumed that each local function can be computed efficiently. In a SyDS, all nodes compute and update their next state *synchronously*. Other update disciplines (e.g., sequential updates) for discrete dynamical systems have also been considered in the literature (e.g., [13,27]). At any time t, the **configuration** $\mathcal{C}$ of a SyDS is the n-vector $(s_1^t, s_2^t, \ldots, s_n^t)$, where $s_i^t \in \mathbb{B}$ is the state of node v_i at time t $(1 \le i \le n)$.

The local function f_v associated with node v of a SyDS $\mathcal{S}$ is a t_v-**threshold** function for some integer $t_v \ge 0$ if the following condition holds: the value of f_v is 1 if the number of 1's in the input to f_v is *at least* t_v; otherwise, the value of the function is 0. Thus, the state of a node may change from 0 to 1 or from 1 to 0. Throughout this paper, we assume that the local transition function for each node is a threshold function.

We let d_v denote the degree of node v, and t_v denote the threshold of node v. The number of inputs to the function f_v is $d_v + 1$. We assume without loss of generality that $0 \le t_v \le d_v + 2$. (The threshold values 0 and $d_v + 2$ allow us to realize functions that always output 1 and 0 respectively.)

Example. Consider the graph shown in Fig. 1. Suppose the local transition functions at each of the nodes v_1, v_4, v_5, v_6 is the **1-threshold function**, the function at v_3 is the **2-threshold function** and that at v_2 is the **3-threshold function**. Assume that initially, v_1, v_4 and v_6 are in state 1 and all other nodes are in state 0. During the first time step, the state of node v_5 changes to 1 since its neighbor v_6 is in state 1; the states of other nodes do not change. The configurations at subsequent time steps are shown in the figure. The configuration $(1, 1, 1, 1, 1, 1)$ at time step 3 is a stable configuration for this system. □

2.2. Additional terminology and notation

A **trajectory** of a SyDS $\mathcal{S}$ is a sequence of configurations $\langle C_1, C_2, \ldots, C_r \rangle$, with $r \ge 2$, such that C_{i+1} is the successor of C_i, for $1 \le i \le r - 1$. When the last two configurations in a trajectory are identical, the trajectory ends in a stable configuration. A stable configuration is a special trajectory consisting of two identical configurations. Given a configuration $\mathcal{C}$ and a node v, we let $\mathcal{C}(v)$ denote the value of v in $\mathcal{C}$, and $\mathcal{C}^v$ denote the number of 1's in the input to f_v in $\mathcal{C}$.

A problem is **fixed parameter tractable** (FPT) with respect to a parameter k if there is an algorithm for the problem with a running time of $O(h(k)N^{O(1)})$, where N is the size of the problem instance and the function $h(k)$ depends only on k (see e.g., [52]). In particular, the function h does not depend on N. Many combinatorial problems are known to be fixed parameter tractable [52]. In Sections 4 and 5 we show that two of the inference problems considered in this paper are fixed parameter tractable.

A problem is solvable in **quasi-polynomial time** if there is an algorithm for the problem with a running time of $2^{O((\log N)^c)}$ where N is the size of the problem instance and c is a constant. In Section 5, we present quasi-polynomial time algorithms for two constrained versions of threshold inference problems.

Given a bipartite graph $G(V_1, V_2, E)$, a **matching** in G is a subset M of edges such that no two edges of M share an end point. A matching of largest cardinality is called a **maximum matching**. It is well known that for any bipartite graph with n nodes and m edges, a maximum matching can be found in $O(m\sqrt{n})$ time [53]. We will use this result in Section 4.2.

2.3. Problem formulations and summary of results

2.3.1. Overview

In all of the threshold inference problems considered in this paper, it is assumed that the underlying graph of the SyDS is given and that each local function is a threshold function with an unknown threshold value. We first define unconstrained versions of threshold inference problems considered in this paper. Subsequently, we define constrained versions of threshold inference problems.

Table 1
Table showing results for homogeneous and heterogeneous behavior specifications.

Problem	Results	
ITSC	Efficiently solvable	Theorem 3
ITUC	Efficiently solvable	Theorem 5
ITT	Efficiently solvable	Theorem 7
ITGE	Efficiently solvable	Theorem 9
Max-ITSC	$W[1]$-complete and no $O(n^{1-\epsilon})$ approximation unless **P** = **NP**	Theorem 4
Max-ITUC	Efficiently solvable	Corollary 6
Max-ITT	$W[1]$-complete and no $O(n^{1-\epsilon})$ approximation unless **P** = **NP**	Corollary 8
Max-ITGE	Efficiently solvable	Corollary 10
ITSUC	**NP**-complete but fixed parameter tractable with respect to the number of unstable configurations	Theorems 11 and 14

2.3.2. Unconstrained inference problems

We start with the definitions of threshold inference problems with *homogeneous* behavior specifications. In this case, the input representing behavior is a set Q along with a type tag; all elements of Q are of the type specified by the tag. For example, a set Q with type tag "stable configuration" indicates that each element of Q is a stable configuration. Similar interpretations can be given for the type tags "unstable configuration", "trajectory" and "GE configuration". We say that a SyDS $\mathcal{S}$ **exhibits the behavior specified by** Q if $\mathcal{S}$ satisfies the property specified by the type tag of Q for each element of Q. For example, if the type tag of Q is "stable configuration", then $\mathcal{S}$ exhibits the behavior specified by Q if each element of Q is a stable configuration of $\mathcal{S}$. The general problem formulation for the homogeneous case can now be stated as follows.

Inferring Thresholds from Homogeneous Behavior Specifications

Given: A partially specified SyDS $\mathcal{S}$ over $\{0, 1\}$ consisting of the underlying graph G and a set Q with a type tag.

Question: Is there an assignment of threshold values to the nodes of $\mathcal{S}$ such that the resulting fully specified SyDS $\mathcal{S}$ exhibits the behavior corresponding to the type tag of Q?

A number of specific problems can be derived from the above general definition. For example, when the type tag of Q is "stable configuration", we refer to the resulting problem as **Inferring Thresholds from Stable Configurations** (ITSC). Likewise, when the type tag of Q is "trajectory" or "unstable configuration" or "GE configuration", we refer to the corresponding problems as Inferring Thresholds from Trajectories (ITT), Unstable Configurations (ITUC) and GE Configurations (ITGE) respectively.

When the answer to an instance of a decision problem such as ITSC is "no", it is natural to consider a maximization version where the goal is to find a threshold assignment that makes a largest subset of Q to be stable configurations. We use the prefix "Max" in naming these problems. Thus, we denote the maximization versions of ITSC, ITT, ITUC and ITGE by Max-ITSC, Max-ITT, Max-ITUC and Max-ITGE respectively. At the end of Section 3, we briefly comment on the complementary versions of these problems where the goal is to delete the minimum number of observations so that there is a solution for the remaining set of observations.

When the behavior specification consists of two or more sets, each with a different type tag, we obtain inference problems for *heterogeneous* behavior specifications. Many such inference problems can be formulated by considering combinations of type tags. Here, we focus on the problem where the observed behavior is specified by two sets Q_1 and Q_2 with type tags "stable configuration" and "unstable configuration" respectively. We say that a SyDS $\mathcal{S}$ exhibits the behavior specified by Q_1 and Q_2 if $\mathcal{S}$ exhibits the behavior specified by Q_1 as well as Q_2. We refer to the corresponding problem as **Inferring Thresholds from Stable and Unstable Configurations** (ITSUC). Table 1 shows our results for the above problems.

2.3.3. Inference problems with constraints

As explained in Section 1.1, several application scenarios impose constraints on the threshold values of nodes. In such cases, the solution for a threshold inference problem must also satisfy the specified constraints. This section defines the constrained versions of the threshold inference problems studied in this paper.

We consider two forms of constraints, namely "equal" (denoted by EQ) and "not equal" (denoted by NE) constraints. The constraint EQ(v_i, v_j) requires that nodes v_i and v_j must have the *same* threshold value in the solution to the corresponding inference problem. Similarly, the constraint NE(v_i, v_j) requires that nodes v_i and v_j must have *different* threshold values.

We consider these constraints in conjunction with two of the inference problems, namely ITSC (Inferring Thresholds from Stable Configurations) and ITUC (Inferring Thresholds from Unstable Configurations), defined in Section 2.3.2. The ITSC problem with equality constraints (denoted by ITSC-EQ) is similar to the ITSC problem except that the input includes a set Γ_{EQ} of equality constraints, and the solution to the ITSC problem must also satisfy those constraints. The ITSC problem with

Table 2
Table showing results for constrained versions of inference problems.

Problem	Results	
ITSC-EQ	Efficiently solvable	Theorem 15
ITSC-NE	**NP**-complete but fixed parameter tractable with respect to the number of constraints	Proposition 16 and Theorem 17
ITUC-EQ	Solvable in quasi-polynomial time	Theorem 18
ITUC-NE	**NP**-complete but solvable in quasi-polynomial time when the number of inequality constraints is *fixed*	Proposition 19 and Theorem 21

inequality constraints is denoted by ITSC-NE. The corresponding problems derived from ITUC are denoted by ITUC-EQ and ITUC-NE respectively. Table 2 shows our results for four constrained threshold inference problems.

2.4. Preliminary results

Here, we present some preliminary results which will be used throughout this paper.

Lemma 1. *Every SyDS $\mathcal{S}$ where each local transition function is a threshold function has at least one stable configuration. Furthermore, a stable configuration can be found in polynomial time.*

Proof. Given the SyDS, we say that a given node v is **forced** if either $t_v = 0$ or $t_v = d_v + 2$. Note that if $t_v = 0$, then in every stable configuration, the value (i.e., state) of node v is 1; and if $t_v = d_v + 2$, then in every stable configuration, the value of node v is 0.

We can construct a stable configuration D of the given SyDS $\mathcal{S}$ as follows. As long as there is at least one forced node, we repeatedly delete a forced node, modifying the SyDS each time, as described below. Suppose we select node v as the forced node to be deleted.

1. If $t_v = 0$, we set the value of v in D to 1. We then delete node v from the SyDS. Every neighbor w of v such that $t_w > 0$ has its threshold reduced by one.
2. If $t_v = d_v + 2$, we set the value of v in D to 0. We then delete node v from the SyDS. Every neighbor w of v such that $t_w = d_w + 2$ (where d_w is the degree of node w before the deletion of v) has its threshold reduced by one.

Note that the deletion of a node can cause a neighboring node that was previously unforced to become forced.

If there are no nodes left, a stable configuration has been constructed. If there are any nodes left, they are all unforced. We can extend the partial configuration constructed so far by making all the remaining nodes 0 (or all the remaining nodes 1). The result is a stable configuration. □

Lemma 2. *Given the underlying graph G of a SyDS $\mathcal{S}$ over $\{0, 1\}$ and a configuration D of $\mathcal{S}$, there is a linear time algorithm for constructing an assignment of threshold values to the nodes of $\mathcal{S}$ such that D is the successor of every configuration of $\mathcal{S}$.*

Proof. The thresholds are set as follows.

1. If node v has value 1 in D, we set t_v to be 0, so that the successor for *every* configuration has the value 1 for v.
2. If node v has value 0 in D, we set t_v to be $d_v + 2$, so that the successor for *every* configuration has the value 0 for v. □

3. Threshold inference from homogeneous behavior specifications

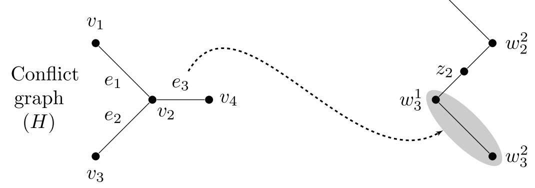
3.1. Inferring thresholds from stable configurations

In this section, we present an efficient algorithm for the problem of inferring thresholds from stable configurations (ITSC). We also present complexity and non-approximability results for Max-ITSC.

Theorem 3. *The ITSC problem can be solved efficiently. When there is a solution, an assignment of threshold values to the nodes can also be obtained efficiently.*

Proof. Let $Q = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_r\}$ be the given set with type tag "stable configuration". We can consider the threshold assignment separately for each node. For any node v, let Q_v^0 (Q_v^1) be the subset of Q such that for each configuration in Q_v^0 (Q_v^1) the value of v is 0 (1). We use f_v to denote the threshold function at node v. Recall that $\mathcal{C}^v$ denotes the number of 1's in the input to f_v in $\mathcal{C}$.

	w_1^1	w_1^2	w_2^1	w_2^2	w_3^1	w_3^2	z_1	z_2
$\mathcal{C}_1$	0	1	0	0	0	0	0	0
$\mathcal{C}_2$	1	0	0	1	0	1	0	0
$\mathcal{C}_3$	0	0	1	0	0	0	0	0
$\mathcal{C}_4$	0	0	0	0	1	0	0	0



Threshold assignments: $t_{w_1^1} = t_{w_2^1} = t_{w_3^1} = 1$; $t_{w_1^2} = t_{w_2^2} = t_{w_3^2} = t_{z_1} = t_{z_2} > 1$.

Fig. 2. An example of Max-ITSC instance and corresponding conflict graph. The shaded configurations are satisfied by the threshold assignments.

1. If Q_v^0 is nonempty, we consider each configuration $\mathcal{C}_i \in Q_v^0$. Since $\mathcal{C}_i$ is a stable configuration, the threshold t_v must satisfy the condition $t_v > \mathcal{C}_i^v$, i.e., $t_v \geq \mathcal{C}_i^v + 1$. Let $t_v^{low} = 1 + \max_{\mathcal{C}_i \in Q_v^0} \mathcal{C}_i^v$. If Q_v^0 is empty, we let $t_v^{low} = 0$.
2. If Q_v^1 is nonempty, we consider each configuration $\mathcal{C}_j \in Q_v^1$. Since $\mathcal{C}_j$ is a stable configuration, the threshold t_v must satisfy the condition $t_v \leq \mathcal{C}_j^v$. Let $t_v^{high} = \min_{\mathcal{C}_j \in Q_v^1} \mathcal{C}_j^v$. If Q_v^1 is empty, we let $t_v^{high} = d_v + 2$.
3. The two steps above provide a collection of constraints that can be satisfied provided $t_v^{low} \leq t_v^{high}$. When there is a solution, any value for t_v can be chosen such that $t_v^{low} \leq t_v \leq t_v^{high}$.

Clearly, the above computations can be done in polynomial time. □

Also note that for a given node v, if Q_v^0 is nonempty, then $t_v^{low} \geq 1$; and if Q_v^1 is nonempty, then $t_v^{high} \leq d_v + 1$. Thus, if there is a solution, then there is a solution where none of the local functions are constant functions; that is, there is a solution where for each node v, $1 \leq t_v \leq d_v + 1$.

Recall that in the Max-ITSC problem, the goal is to choose threshold values so that a maximum number of elements in the set Q are stable points of $\mathcal{S}$. We now establish a hardness result for this problem which holds even when the underlying graph of a SyDS is a simple path.

The idea behind the proof is the following. Let us say that two configurations $\mathcal{C}_i$ and $\mathcal{C}_j$ **conflict** if there is a node v such that $\mathcal{C}_i(v) = 0$, $\mathcal{C}_j(v) = 1$, and $\mathcal{C}_i^v \geq \mathcal{C}_j^v$. Note that if $\mathcal{C}_i$ and $\mathcal{C}_j$ conflict, then there is no assignment of thresholds under which both $\mathcal{C}_i$ and $\mathcal{C}_j$ are stable. On the other hand, if a set of configurations is conflict-free (i.e., no pair of configurations in the set conflict), then thresholds can be assigned so that all the configurations in the set are stable. Given an ITSC problem instance, define the **conflict graph** for the instance to be the undirected graph with a node for each configuration in Q and an edge between each pair of nodes whose corresponding configurations conflict. Each independent set [54] of size r in the conflict graph gives a subset Q' of Q with r configurations that can be made stable. Since the Maximum Independent Set (MIS) problem is in $W[1]$ [55], this reduction from the Max-ITCS problem to the MIS problem shows that Max-ITCS is in $W[1]$, with the parameter being the number of configurations that must be made stable.

Theorem 4. *Max-ITSC is $W[1]$-complete, where the parameter is the number of configurations that must be made stable. Further, for any $\epsilon > 0$, there is no polynomial time $O(n^{1-\epsilon})$ approximation algorithm for Max-ITSC, unless* **P** = **NP**. *(Here n is the number of configurations in the given set Q.) These results hold even when the underlying graph of the SyDS is a simple path.*

Proof. The membership of Max-ITSC in $W[1]$ was discussed just before the statement of the theorem. To establish $W[1]$-hardness, we use a reduction from the MIS problem: given an undirected graph $H(V_H, E_H)$ and an integer $K \leq |V_H|$, does H have an independent set of size at least K? MIS is known to be $W[1]$-complete [55].

The key to the $W[1]$-hardness reduction is to construct an ITSC problem instance from the MIS in such a way that the conflict graph (defined above) for the constructed Max-ITSC problem instance is identical to the graph in the given MIS problem instance. We do this as follows. (Fig. 2 illustrates the construction.) There is a configuration for each node of H, with configuration $\mathcal{C}_i$ corresponding to node $v_i \in V_H$. We let $Q = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$, where $n = |V_H|$. Let $E_H = \{e_1, e_2, \ldots, e_m\}$. For each edge $e_k \in E_H$, the underlying graph G for SyDS $\mathcal{S}$ has two nodes denoted by w_k^1 and w_k^2 along with the edge $\{w_k^1, w_k^2\}$. Suppose the edge $e_k \in E_H$ joins nodes v_i and v_j of H. We set $\mathcal{C}_i(w_k^1) = 0$, $\mathcal{C}_i(w_k^2) = 1$, $\mathcal{C}_j(w_k^1) = 1$, and $\mathcal{C}_j(w_k^2) = 0$. All other configurations in Q have value 0 for both w_k^1 and w_k^2. Nodes w_k^1 and w_k^2 induce a conflict between $\mathcal{C}_i$ and $\mathcal{C}_j$, but do not induce a conflict between any other pair of configurations. Thus, any independent set of V_H corresponds to a conflict-free subset of configurations of $\mathcal{S}$ and vice versa.

To ensure that the underlying graph of SyDS $\mathcal{S}$ is a simple path, we add $m-1$ new nodes denoted by $z_1, z_2, \ldots, z_{m-1}$. For each j, $1 \le j \le m-1$, node z_j is adjacent to w_j^2 and w_{j+1}^1. In each of the n configurations constructed above, the values of the nodes $z_1, z_2, \ldots, z_{m-1}$ are all 0.

It can be verified that H has an independent set of size K if and only if there is a conflict-free subset of Q with K configurations. This proves the $W[1]$-hardness of Max-ITSC.

It is well known that for any $\epsilon > 0$, there is no polynomial time $O(n^{1-\epsilon})$-approximation algorithm for the MIS problem, unless **P** = **NP** [56]. Since our construction preserves approximations (i.e., for each r, any independent set of size r in H leads to a subset of r conflict-free configurations of $\mathcal{S}$ and vice versa), the same negative result holds for Max-ITS as well. □

3.2. Inferring thresholds from unstable configurations

Here, we consider the ITUC problem where the goal is to infer thresholds from a given set Q of unstable configurations. In this case, we show that both ITUC and Max-ITUC can be solved efficiently.

Theorem 5. *The ITUC problem can be solved efficiently. When there is a solution, an assignment of threshold values to the nodes can also be obtained efficiently.*

Proof. If Q contains every possible configuration of $\mathcal{S}$ (i.e., $|Q| = 2^n$, where n is the number of nodes in the underlying graph of $\mathcal{S}$), then, by Lemma 1, there is no solution to the problem.

Suppose that Q excludes at least one configuration of $\mathcal{S}$. Then there is always a solution to the problem, as outlined below. Let $\mathcal{C}$ be a configuration that is not in Q. From Lemma 2, we can set the thresholds so that the successor of *every* configuration is the configuration $\mathcal{C}$; that is, every configuration other than $\mathcal{C}$ is an unstable configuration. □

The following result for Max-ITUC is a simple consequence of the above theorem.

Corollary 6. *Max-ITUC is efficiently solvable.*

3.3. Inferring thresholds from trajectories

Here, we present results for the ITT problem, where the goal is to infer thresholds given a set Q of trajectories.

Theorem 7. *The ITT problem can be solved efficiently. When there is a solution, an assignment of threshold values to the nodes can also be obtained efficiently.*

Proof. The main idea is similar to the one used in establishing the efficient solvability of ITSC (Theorem 3). As before, we can consider the threshold assignment separately for each node v of the underlying graph. Let $T_i = \langle \mathcal{C}_{i,1}, \mathcal{C}_{i,2}, \ldots, \mathcal{C}_{i,r} \rangle$ denote one of the trajectories. Consider any transition $\mathcal{C}_{i,j}$ to $\mathcal{C}_{i,j+1}$ in T_i. We now observe that this transition gives rise to an inequality for the threshold t_v of v, depending on the values of node v in $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i,j+1}$. There are four cases to consider.

Case 1: $\mathcal{C}_{i,j}(v) = 0$ and $\mathcal{C}_{i,j+1}(v) = 0$. Here, $\mathcal{C}_{i,j}^v$ (i.e., the number of 1's in $\mathcal{C}_{i,j}$ which are inputs to the function f_v) is too small to change the value of v from 0 to 1. In other words, we have the inequality $t_v > \mathcal{C}_{i,j}^v$ in this case.

Case 2: $\mathcal{C}_{i,j}(v) = 0$ and $\mathcal{C}_{i,j+1}(v) = 1$. Here, $\mathcal{C}_{i,j}^v$ is large enough to change the value of v from 0 to 1. Thus, we get the inequality $t_v \le \mathcal{C}_{i,j}^v$.

Case 3: $\mathcal{C}_{i,j}(v) = 1$ and $\mathcal{C}_{i,j+1}(v) = 0$. Here, the corresponding inequality is $t_v > \mathcal{C}_{i,j}^v$.

Case 4: $\mathcal{C}_{i,j}(v) = 1$ and $\mathcal{C}_{i,j+1}(v) = 1$. Here, the corresponding inequality is $t_v \le \mathcal{C}_{i,j}^v$.

Thus, each transition in each trajectory gives rise to an inequality for t_v. The number of inequalities for any node v is at most the total number of transitions over all the trajectories. There is a solution to the given ITT instance if and only if for each node v, all the inequalities can be satisfied. Clearly, these computations can be done in polynomial time. □

Since each stable configuration is a special case of a trajectory (containing two identical configurations), the following result is a direct consequence of Theorem 4.

Corollary 8. *Max-ITT is $W[1]$-hard. Further, for any $\epsilon > 0$, there is no polynomial time $O(n^{1-\epsilon})$ approximation algorithm for Max-ITT, unless **P** = **NP**. (Here n is the number of trajectories in the given set Q.) These results hold even when underlying graph of the SyDS is a simple path.*

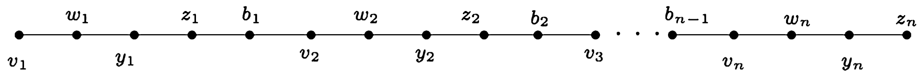


Fig. 3. Reduction used in the Proof of Theorem 11.

3.4. *Inferring thresholds from Garden of Eden configurations*

We now present results for the ITGE problem, where the goal is to infer thresholds given a set Q of Garden of Eden configurations.

Theorem 9. *The ITGE problem can be solved efficiently. When there is a solution, an assignment of threshold values to the nodes can also be obtained efficiently.*

Proof. Suppose that Q contains every possible configuration of $\mathcal{S}$ (i.e., $|Q| = 2^n$, where n is the number of nodes in the underlying graph of $\mathcal{S}$). Since every configuration has a successor, there is at least one configuration with a predecessor, so there is no solution to the problem instance.

Suppose that Q excludes at least one configuration of $\mathcal{S}$. Then there is always a solution to the problem instance, as outlined below. Let $\mathcal{C}$ be a configuration that is not in Q. From Lemma 2, we can set the thresholds so that the successor of *every* configuration is configuration $\mathcal{C}$, so that every configuration other than $\mathcal{C}$ has no predecessor. In other words, every element of Q is a GE configuration. □

The following result for Max-ITGE is a simple consequence of the above theorem.

Corollary 10. *Max-ITGE is efficiently solvable.* □

3.5. *Some observations regarding minimization versions*

When there is no solution to a given threshold inference problem, we considered the problem of finding a maximum subset of configurations for which there is a solution. One can also consider the complementary problem, where the goal is to delete the *minimum* number of configurations so that there is a solution to the remaining set of configurations. Since Max-ITUC and Max-ITGE are efficiently solvable (Corollaries 6 and 10), the corresponding minimum deletion versions are also efficiently solvable. By starting with the minimum vertex cover (MVC) problem [54] and carrying out the same reduction used in the hardness proofs for Max-ITSC and Max-ITT (Theorem 4 and Corollary 8), the corresponding minimum deletion versions can be seen to be **NP**-complete. Also, the minimization versions of these two problems can be reduced to the MVC problem for the corresponding conflict graphs along the lines of the discussion preceding the statement of Theorem 4. Since MVC is fixed parameter tractable [52], this direct connection to MVC points out that the minimization versions of ITSC and ITT are also fixed parameter tractable. Further, all known approximation results for MVC [57] also apply to the minimization versions of ITSC and ITT.

4. Inference from heterogeneous collections of behavior

4.1. *The complexity of ITSUC*

We now consider the ITSUC problem, where there are two sets Q_1 and Q_2 of configurations, and the requirement is to find a threshold value for each node of $\mathcal{S}$ such that each configuration in Q_1 is stable and each configuration in Q_2 is unstable. The following theorem shows the complexity of this decision problem.

Theorem 11. *The ITSUC problem is **NP**-complete even when the underlying graph of the given SyDS is a simple path.*

Proof. It is readily seen that ITSUC is in **NP**. We show **NP**-hardness via a reduction from 3SAT.

Suppose the given 3SAT formula has n variables and m clauses. The reduction constructs a SyDS $\mathcal{S}$ whose underlying graph G is a *simple path* containing $5n - 1$ nodes as follows. For each variable x_i of the formula, graph G contains four nodes, which we denote as v_i, w_i, y_i, and z_i, and the three edges $\{v_i, w_i\}$, $\{w_i, y_i\}$, and $\{y_i, z_i\}$. We also have $n - 1$ additional nodes $b_1, b_2, \ldots, b_{n-1}$ (referred to as “buffer nodes”). For each i, $1 \le i \le n - 1$, we add the two edges $\{z_i, b_i\}$ and $\{b_i, v_{i+1}\}$. Thus, the buffer nodes connect together the subgraphs created from the variables so that the underlying graph of $\mathcal{S}$ is a simple path on $5n - 1$ nodes. (See Fig. 3.)

The set $Q_1 = \{\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5\}$ consists five configurations which are required to be stable. These five configurations are constructed as follows.

1. In $\mathcal{C}_1$, every node v_i and w_i has value 0, every node y_i and z_i has value 1 and all buffer nodes have value 0.
2. In $\mathcal{C}_2$ every node v_i and w_i has value 1, every node y_i and z_i has value 0 and all buffer nodes have value 0.
3. In $\mathcal{C}_3$, every node v_i, w_i, and y_i has value 0, every node z_i has value 1 and all buffer nodes have value 0.
4. In $\mathcal{C}_4$, every node v_i, w_i, y_i and z_i has value 1, and all buffer nodes have value 0.
5. In $\mathcal{C}_5$, all the nodes of $\mathcal{S}$ have value 1.

The set $Q_2 = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$ contains m configurations, with configuration $\mathcal{D}_j$ corresponding to clause c_j of the 3SAT instance, $1 \le j \le m$. Each of these m configurations is required to be unstable. Consider a given clause c_j. If a given variable x_i occurs as a positive literal in c_j, then in $\mathcal{D}_j$, node v_i has value 1, w_i has value 0, y_i has value 0, and z_i has value 0. If variable x_i occurs as a negative literal (i.e., as $\overline{x}_i$) in c_j, then in $\mathcal{D}_j$, node v_i has value 0, w_i has value 1, y_i has value 1, and z_i has value 1. If no literal for variable x_i occurs in c_j, then in D_j, node v_i has value 0, w_i has value 0, y_i has value 1, and z_i has value 1. In all the m configurations, each buffer node has value 0.

Now consider the thresholds of the nodes in G. For each i, $1 \le i \le n$, configuration $\mathcal{C}_1$ imposes the constraints that $t_{v_i} > 0$, $t_{w_i} > 1$, $t_{y_i} \le 2$, and $t_{z_i} \le 2$. For each i, $1 \le i \le n$, configuration $\mathcal{C}_2$ imposes the constraints that $t_{v_i} \le 2$, $t_{w_i} \le 2$, $t_{y_i} > 1$, and $t_{z_i} > 0$. For each i, $1 \le i \le n$, configuration $\mathcal{C}_3$ imposes the constraints that $t_{v_i} > 0$, $t_{w_i} > 0$, $t_{y_i} > 1$, and $t_{z_i} \le 1$. Together, these constraints require that t_{v_i} is 1 or 2, $t_{w_i} = 2$, $t_{y_i} = 2$, and $t_{z_i} = 1$. We view setting $t_{v_i} = 2$ as corresponding to setting Boolean variable x_i to 1, and setting $t_{v_i} = 1$ as corresponding to setting x_i to 0. For each i, $1 \le i \le n-1$, configuration $\mathcal{C}_4$ imposes the constraint that $t_{b_i} > 2$. For each i, $1 \le i \le n-1$, configuration $\mathcal{C}_5$ imposes the constraint that $t_{b_i} \le 3$. Thus, $\mathcal{C}_4$ and $\mathcal{C}_5$ together ensure that $t_{b_i} = 3$ for $1 \le i \le n-1$.

Suppose the given 3SAT formula is satisfiable. Consider a given satisfying assignment α. We construct an assignment of threshold values to the nodes of G, as follows. For each i, $1 \le i \le n$, we set $t_{w_i} = 2$, $t_{y_i} = 2$, and $t_{z_i} = 1$. If a given variable x_i has value 1 in α, we set $t_{v_i} = 2$; and if x_i has value 0 in α, we set $t_{v_i} = 1$. Each buffer node is assigned a threshold value of 3. We now argue that these thresholds ensure that none of the configurations in Q_2 is stable. Suppose x_i has value 1 in α, and a given clause c_j contains the positive literal x_i. Then, node v_i has value 1 in $\mathcal{D}_j$, its only neighbor w_i has value 0 in $\mathcal{D}_j$, and $t_{v_i} = 2$, so configuration $\mathcal{D}_j$ is unstable. On the other hand, suppose x_i has value 0 in α, and a given clause c_j contains the negative literal $\overline{x}_i$. Then, node v_i has value 0 in $\mathcal{D}_j$, its only neighbor w_i has value 1 in $\mathcal{D}_j$, and $t_{v_i} = 1$, so configuration $\mathcal{D}_j$ is unstable. Thus, it can be seen that the answer to the ITSUC problem instance is "Yes".

Now suppose that the answer to the ITSUC problem instance is "Yes". Consider a given assignment of threshold values that makes the answer "Yes". We can construct an assignment α of Boolean values to the variables of the given 3SAT formula, as follows. For $1 \le i \le n$, if $t_{v_i} = 2$, we set x_i to the value 1; and if $t_{v_i} = 1$, we set x_i to the value 0. It can be seen that the constructed assignment α is a satisfying assignment.

Thus, the given 3SAT formula is satisfiable iff the answer to the constructed ITSUC problem instance is "Yes". □

4.2. Fixed parameter tractability of ITSUC

We now show that ITSUC is fixed parameter tractable with respect to the number of unstable configurations specified in the problem instance, with no restrictions on the underlying graph of the given SyDS. Given a set $A = \{a_1, a_2, \ldots, a_r\}$, let $P(A)$ denote a **partition** of A into nonempty subsets. Let each subset in $P(A)$ be called a **block**. We use $\pi(A)$ to denote the collection of all partitions of A. For a set A with r elements, it is known that $|\pi(A)| = O((r/\log r)^r)$ [58].

Let Q_1 and Q_2 denote the set of stable and unstable configurations respectively in the given ITSUC instance. Let $q = |Q_2|$ and let n be the number of nodes in the given SyDS $\mathcal{S}$. From the proof of Theorem 3, it can be seen that the configurations in Q_1 impose constraints on the threshold value of each node of $\mathcal{S}$. Given any configuration $\mathcal{C} = (s_1, s_2, \ldots, s_n)$ in Q_2, we can try to make $\mathcal{C}$ an unstable configuration by choosing a threshold t_{v_i} for node v_i so that in the successor $\mathcal{C}'$ of $\mathcal{C}$, the state of v_i is different from s_i, $1 \le i \le n$. Such a choice must also satisfy the constraints imposed on t_{v_i} by the configurations of Q_1. Given an instance of ITSUC, we say that a node v of $\mathcal{S}$ is **compatible** with a configuration $\mathcal{C} \in Q_2$ if a value for t_v can be chosen so that (i) t_v satisfies all the constraints imposed by the collection Q_1 and (ii) this choice makes $\mathcal{C}$ an unstable configuration (regardless of the threshold values assigned to the other nodes). Extending this definition, we say that a node v is **compatible with a subset** R of Q_2 if v is compatible with every configuration in R.

The above definitions are used in our algorithm (Alg-ITSUC) shown in Fig. 4. The algorithm considers each possible partition of the set Q_2 of unstable configurations. For each partition P, it constructs a bipartite graph H_P with the set V of nodes of the SyDS $\mathcal{S}$ on one side and a set V_P of nodes corresponding to the blocks of P on the other side. Each edge $\{x, y\}$ of H_P, where $x \in V$ and $y \in V_P$ indicates that node x of $\mathcal{S}$ is compatible with the block of P corresponding to node y. (The construction of this bipartite graph is depicted in Fig. 5.) If the algorithm encounters a partition P for which the corresponding bipartite graph H_P contains a matching whose size is equal to the number of blocks in P, the algorithm outputs "Yes"; otherwise, the algorithm outputs "No". We now establish the correctness of the algorithm and its running time.

Lemma 12. *Algorithm Alg-ITSUC given in Fig. 4 correctly decides whether the ITSUC instance has a solution.*

Proof. We will show that the ITSUC instance has a solution iff Alg-ITSUC outputs "Yes".

```
Input: Graph $G(V, E)$ of a SyDS $\mathcal{S}$, and configuration sets $Q_1$ and $Q_2$.

Requirement: Output "Yes" if there is a threshold value $t_v$ for each $v \in V$ such
that in the resulting SyDS, all the configurations in $Q_1$ are stable and all the
configurations in $Q_2$ are unstable. Otherwise, output "No".

Steps:
1. for each partition $P$ in $\pi(Q_2)$ do
   (a) Let $k$ denote the number of blocks in $P$ and let $B_1, B_2, \ldots B_k$ denote the
       blocks themselves.
   (b) Construct the bipartite graph $H_P(V, V_P, E_P)$ where $V_P$ has one node for
       each block in $P$ and $E_P = \{\{x, y\} : x \in V, y \in V_P$ and node $x$ of $\mathcal{S}$ is
       compatible with the block of $P$ represented by node $y\}$.
   (c) if $H_P$ has a matching with $k$ edges then output "Yes" and stop.
   endfor
2. Output "No".
```

Fig. 4. Algorithm Alg-ITSUC to show the fixed parameter tractability of ITSUC.

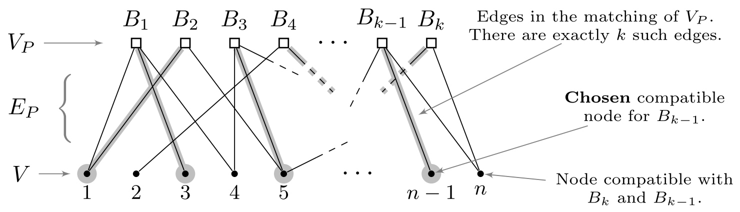

B_1 B_2 B_3 B_4 B_{k-1} B_k
V_P
E_P
V
1 2 3 4 5 $n-1$ n
Edges in the matching of V_P. There are exactly k such edges.
Chosen compatible node for B_{k-1}.
Node compatible with B_k and B_{k-1}.

Fig. 5. The bipartite graph $H_P(V, V_P, E_P)$.

Part 1 (if): Suppose the algorithm outputs the answer "Yes". We will prove that there is a solution to the ITSUC instance.

From the description of the algorithm, we note that in this case, there is a partition P of Q_2 with k (nonempty) blocks such that there is a matching with k edges in the corresponding bipartite graph H_P. In such a matching, each node y that corresponds to block B_y of P is matched to some node x of the SyDS $\mathcal{S}$. By our construction, node x is compatible with block B_y. By the definition of compatibility, there is a threshold value t_x for node x such that t_x satisfies all the constraints imposed by the configurations in Q_1, and further, this value of t_x for x makes all the configurations in B_y unstable. Since every block of P is matched to a distinct node of $\mathcal{S}$, it follows that threshold values can be chosen for each node of $\mathcal{S}$ independently to satisfy the required conditions. In other words, there is a solution to the ITSUC instance.

Part 2 (only if): Suppose there is a solution to the ITSUC instance. We will prove that the algorithm outputs the answer "Yes". The essence of the argument is the following: each solution to the ITSUC instance induces a partition P of the set Q_2 of unstable configurations so that the bipartite graph H_P constructed in Step 1(b) of Alg-ITSUC has a matching whose size is the number of blocks in P.

Consider a solution Γ to the ITSUC instance which assigns the threshold value t_{v_i} for each node v_i of $\mathcal{S}$. Let $Q_2 = \{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m\}$. Since Γ is a solution to the ITSUC instance, for each $\mathcal{D}_j \in Q_2$, there is a node v_r which is compatible with $\mathcal{D}_j$, $1 \le j \le m$; that is, the threshold assignment t_{v_r} to v_r ensures that $\mathcal{D}_j$ is an unstable configuration. (When there are two or more nodes which are compatible with a given configuration $\mathcal{D}_j \in Q_2$, we choose a node arbitrarily.) Thus, the given solution Γ allows us to assign each configuration in Q_2 to some node of $\mathcal{S}$. By this method, we assign a (possibly empty) subset, say D_i, of Q_2 to each node v_i of $\mathcal{S}$ so that v_i is compatible with each configuration in D_i. Since each configuration of Q_2 is assigned to only one node of $\mathcal{S}$, the collection of sets $D_1, \ldots, D_n$ is pairwise disjoint. Thus, the non-empty sets in this collection are the blocks of a partition P of Q_2. Let k denote the number of such blocks in P.

Since Alg-ITSUC considers all the partitions of Q_2, it will surely encounter the partition P. Further, as argued above, for every block B_y of P, there is a node x of $\mathcal{S}$ such that x is compatible with B_y. Moreover, for the partition P, there is an assignment of blocks of P to the nodes of $\mathcal{S}$ such that no two blocks are assigned to the same node of $\mathcal{S}$. In other words, the bipartite graph H_P constructed by the algorithm from P has a matching with k edges, where k is the number of blocks of P. Thus, the algorithm will output the answer "Yes", and this completes the proof. □

Lemma 13. *Algorithm Alg-ITSUC can be implemented to run in time $O(h(q)N^{O(1)})$, where q is the number of unstable configurations, N is the size of the problem instance and the function h depends only on q.*

Proof. Let Q_1 (the set of configurations to be made stable) contain r configurations. We first discuss some preprocessing steps. For any node v and any configuration $\mathcal{C}$ in Q_1, the constraint on the threshold t_v of v imposed by $\mathcal{C}$ can be found in $O(n)$ time (since each configuration has n state values). Thus, the constraints on t_v imposed by all r configurations in Q_1 can be found on $O(nr)$ time. As indicated in the proof of Theorem 3, all of these constraints can be combined into a single constraint of the form $t_v^{low} \le t_v \le t_v^{high}$ for appropriate integers t_v^{low} and t_v^{high} in $O(nr)$ time. Thus, obtaining such a single constraint for each of the n nodes can be done in $O(n^2r)$ time. These preprocessing steps can be done before starting the execution of the **for** loop in Step 1 of the algorithm.

We now estimate the time needed to check whether a node v is compatible with a configuration $\mathcal{C}$ in Q_2. For a node v, the constraint on t_v needed to make $\mathcal{C}$ an unstable configuration (regardless of the threshold values of the other nodes) can be determined in $O(n)$ time. Checking whether this constraint also satisfies the constraint on t_v obtained by considering the configurations in Q_1 can be done in $O(1)$ time. Thus, checking whether a node v is compatible with a configuration in Q_2 can be done in $O(n)$ time. The results from these compatibility checks can be stored in an $n \times q$ Boolean matrix M so that during the execution of the **for** loop of the algorithm, we can determine whether a node v is compatible with a configuration $\mathcal{C}$ of Q_2 in $O(1)$ time.

We now estimate the time used for each iteration of the **for** loop of Fig. 4. For each partition P in $\pi(Q_2)$, the algorithm constructs an appropriate bipartite graph H_P and checks whether the graph has a matching whose size is equal to the number of blocks of P. If P has k blocks, the number of nodes in H_P is $n+k$ and the number of edges is at most nk. For each node $x \in V$, the blocks with which x is compatible can be found in $O(q)$ time from the precomputed matrix M. Thus, constructing the graph H_P can be done in time $O(nkq) = O(nq^2)$ since $k \le q$. Since H_P has $n+k \le n+q$ nodes and at most $nk \le nq$ edges, as mentioned in Section 2.2, a maximum matching in H_P can be found in $O(nq\sqrt{n+q})$ time. Thus, each iteration of the **for** loop can be implemented to run in time $O(nq^2 + nq\sqrt{n+q})$ time.

Since the number of iterations of the **for** loop is at most $|\pi(Q_2)|$, the running time of Step 1 is $O(|\pi(Q_2)|(nq^2 + nq\sqrt{n+q}))$. The overall running time of the algorithm, including the preprocessing steps, is $O(n^2(r+q) + |\pi(Q_2)|(nq^2 + nq\sqrt{n+q}))$. As mentioned earlier, $|\pi(Q_2)| = O((q/\log q)^q)$. Since $n+r+q \le N$, where N is the size of the given ITSUC instance, it follows that the running time of our algorithm for ITSUC has the form $O(h(q)N^{O(1)})$, where $h(q) = (q/\log q)^q$ depends only on q. □

The following theorem is a direct consequence of Lemmas 12 and 13.

Theorem 14. *The ITSUC problem is fixed parameter tractable where the parameter is the number of unstable configurations.* □

5. Threshold inference under constraints

5.1. Overview

Here, we consider threshold inference problems with constraints. Section 5.2 considers constrained inference problems for stable configurations while Section 5.3 considers the corresponding problems for unstable configurations.

Throughout this section, Q denotes the set of configurations (which must be made stable or unstable depending on the problem). For problems involving equality constraints, Γ_{EQ} denotes the given set of constraints. For problems involving inequality constraints, the corresponding set is denoted by Γ_{NE}.

We recall an assumption from Section 2: when a node v of a given SyDS $\mathcal{S}$ has degree d_v, any threshold value t_v assigned to the node satisfies the condition $0 \le t_v \le d_v + 2$. This assumption is used throughout this section.

For any node v, choosing $t_v = d_v + 2$ ensures that the local transition function f_v at v always outputs the value 0. In other words, choosing $d_v + 2$ as the threshold value of v is akin to choosing the threshold value of "infinity" for v. For two nodes u and v whose degrees are different, when we choose $t_u = d_u + 2$ and $t_v = d_v + 2$, the threshold values t_u and t_v are numerically different; however, the values effectively represent the same threshold, namely "infinity". So, in such cases, we regard u and v as having the same threshold value; that is, such a threshold assignment satisfies the condition $t_u = t_v$. Also, to satisfy the constraint $t_u \neq t_v$, at least one of u and v must be assigned a threshold different from "infinity".

For simplicity, we present our algorithms for decision versions of the various problems. It is straightforward to modify the algorithms to find a threshold assignment when one exists.

5.2. Inferring thresholds with stable configurations and constraints

Here, we consider the constrained versions of the ITSC problem, namely ITSC-EQ and ITSC-NE. We first show that ITSC-EQ can be solved in polynomial time. Then we show that ITSC-NE is **NP**-complete and also that it is fixed parameter tractable with respect to the number of inequality constraints.

5.2.1. A polynomial time algorithm for ITSC-EQ

Our polynomial time algorithm for the ITSC-EQ problem relies on the ideas developed in the algorithm for the ITSC problem (Theorem 3). In particular, we recall that for each node $v \in V$, the set Q gives rise to lower and upper bounds t_v^{low} and t_v^{high} respectively on the threshold value of v. These bounds have the property that all configurations in Q can be made stable if and only if for each node $v \in V$, a threshold value t_v can be chosen so that $t_v^{low} \le t_v \le t_v^{high}$. Thus, if there is any node v for which $t_v^{low} > t_v^{high}$, then there is no solution to the ITSC-EQ instance. Therefore, for the remainder of this section, we assume that for each node v, $t_v^{low} \le t_v^{high}$.

```
Input: Graph $G(V,E)$ of a SyDS $\mathcal{S}$, a set $Q$ of configurations $\mathcal{S}$ and a collection
$\Gamma_{EQ}$ of constraints of the form $t_{v_i} = t_{v_j}$.

Requirement: Output "Yes" if there is a threshold value $t_v$ for each $v \in V$ such
that in the resulting SyDS, all the configurations in $Q$ are stable and all the
constraints in $\Gamma_{EQ}$ are satisfied. Otherwise, output "No".

Steps:
1. Using the configurations in $Q$, for each node $v \in V$, find lower ($t_v^{low}$) and
   upper ($t_v^{high}$) bounds on the threshold value $t_v$ of $v$ using the method given
   in the proof of Theorem 3.
2. If there is a node $v \in V$ such that $t_v^{low} > t_v^{high}$ then output "No" and stop.
3. Find the partition $P = \{V_1, V_2, \ldots, V_r\}$ of $V$ using the equivalence relation
   $\Gamma_{EQ}$. (Nodes which do not appear in any constraint form singleton blocks
   in $P$.)
4. for $i = 1$ to $r$ do
   (a) Determine whether there is a value $\alpha_i$ such that assigning the threshold
       value $\alpha_i$ to each node in $V_i$ satisfies all the bounds on the threshold val-
       ues of each node in $V_i$. (Note that $0 \le \alpha_i \le n+1$.)
   (b) If there is no such $\alpha_i$, then output "No" and stop.
5. Output "Yes".
```

Fig. 6. A polynomial time algorithm for ITSC-EQ.

Let $V' \subseteq V$ be the set of nodes which appear in one or more constraints of Γ_{EQ}. Since equality of threshold values is an equivalence relation, the set Γ_{EQ} induces a partition of V' which can be constructed in polynomial time. We augment this partition by creating $|V - V'|$ additional blocks, with each such block containing just one node of $V - V'$. Let V_1, V_2, …, V_r denote the blocks in the resulting partition of V. Thus, in any solution to the given ITSC-EQ instance, all the nodes in block V_i must have the same threshold value, $1 \le i \le r$. Recall that for each node v, the threshold values must be in the range $[0 \,..\, d_v + 2]$; since $d_v \le n-1$, the number of possible threshold values for any node is $O(n)$. So, by trying each such threshold value, it is possible to determine in polynomial time whether there is a single threshold value α_i that satisfies all the bounds (determined by Q) for each node in block V_i, $1 \le i \le r$. Clearly, there is a solution to the ITSC-EQ instance if and only if the answer is "Yes" for each block.

An outline of our algorithm for ITSC-EQ appears in Fig. 6. The correctness of the algorithm and the fact that it runs in polynomial time readily follow from the above discussion. The following theorem summarizes our result for the ITSC-EQ problem.

Theorem 15. *The ITSC-EQ problem can be solved in polynomial time.* □

5.2.2. Results for ITSC-NE

We begin by showing that ITSC-NE is **NP**-complete. The proof is by a straightforward reduction from the 3-Coloring problem for undirected graphs: given a graph $H(V_H, E_H)$, can the nodes of H be colored using at most three colors so that no pair of adjacent nodes receives the same color? The 3-Coloring problem is known to be **NP**-complete [54]. In this problem, we may assume without loss of generality that the input graph H requires at least 3 colors since the 2-coloring problem can be solved in polynomial time [54]. The main idea used in our reduction is that each edge of the graph representing an instance of 3-Coloring becomes an inequality constraint for the resulting ITSC-NE instance.

Proposition 16. *The ITSC-NE problem is **NP**-complete.*

Proof. It is easy to see that ITSC-NE is in **NP**. The proof of **NP**-hardness is through a reduction from the 3-Coloring problem. Let the graph $H(V_H, E_H)$ represent the given instance of the 3-Coloring problem. Let $V_H = \{w_1, w_2, \ldots, w_n\}$ so that $|V_H| = n$. We construct an instance of the ITSC-NE problem as follows. For each node $w_i \in V_H$, the underlying graph $G(V,E)$ of the SyDS $\mathcal{S}$ has two nodes denoted by v_i and x_i, $1 \le i \le n$. Let $V = \{v_1, v_2, \ldots, v_n\}$ and $X = \{x_1, x_2, \ldots, x_n\}$. Node v_i corresponds to node w_i of H, $1 \le i \le n$. The edge set E of G consists of n edges, namely $\{v_1, x_1\}$, $\{v_2, x_2\}$, …, $\{v_n, x_n\}$. The only configuration $\mathcal{C}$ of $\mathcal{S}$ that must be made stable has the state value 1 for all the nodes of G. For each edge $\{w_i, w_j\} \in E_H$, the constraint $t_{v_i} \ne t_{v_j}$ is added to the set Γ_{NE} of inequality constraints. (There are no constraints on the threshold value of the nodes in X.) This completes the construction, which can obviously be carried out in polynomial time.

Suppose H is 3-colorable. Let the three colors be denoted by 0, 1 and 2. Further, let V_H^i denote the color class i (i.e., the set of nodes of H assigned color i), $i = 0, 1, 2$. Consider the following threshold assignment: all the nodes of V corresponding to the color class V_H^i are assigned the threshold value i ($i = 0, 1, 2$), and each node in X is assigned the threshold value 1. It can be verified that under this assignment, $\mathcal{C}$ is a stable configuration and that all the constraints in Γ_{NE} are satisfied.

Now, suppose there is a solution to the ITSC-NE instance. Since the degree of each node in G is 1, the solution must assign the threshold value 0, 1, 2 or 3 to each node of G. Also, since the configuration $\mathcal{C}$ (in which the state of every node is 1) is required to be stable, no node of G can be assigned the threshold of 3; that is, each node of G must be assigned a

Input: Graph $G(V, E)$ of a SyDS $\mathcal{S}$, a set Q of configurations and a collection Γ_{NE} of constraints of the form $t_{v_i} \neq t_{v_j}$. Let $k = |\Gamma_{NE}|$.

Requirement: Output "Yes" if there is a threshold value t_v for each $v \in V$ such that in the resulting SyDS, all the configurations in Q are stable and all the constraints in Γ_{NE} are satisfied. Otherwise, output "No".

Steps:

1. Using the configurations in Q, for each node $v \in V$, find lower (t_v^{low}) and upper (t_v^{high}) bounds on the threshold value t_v of v using the method given in the proof of Theorem 3.
2. If there is a node $v \in V$ such that $t_v^{low} > t_v^{high}$ then **output** "No" and **stop**.
3. Construct graph $G_\Gamma(V_1, E_1)$: $V_1 \subseteq V$ is the set of nodes involved in one or more constraints in Γ_{NE} and for each constraint $t_{v_i} \neq t_{v_j}$ in Γ_{NE}, E_1 contains the edge $\{v_i, v_j\}$.
4. For each node $v \in V_1$, construct a list L_v containing each integer value α such that $t_v^{low} \leq \alpha \leq t_v^{high}$.
5. Let $V_1' \subseteq V_1$ be the set of nodes such that for each node $w \in V_1'$, the list L_w has at least $k+1$ values. Let G_Γ^1 be the subgraph of G_Γ induced on the node set $V_1 - V_1'$.
6. Consider each combination β of values in the lists for the nodes of G_Γ^1. If some combination β yields a valid list coloring of G_Γ^1, then **output** "Yes"; otherwise, **output** "No".

Fig. 7. Algorithm to show the fixed parameter tractability of ITSC-NE.

threshold value of 0, 1 or 2. Consider the following assignment of colors to the nodes of H: if node v_i is assigned threshold α (for some $\alpha \in \{0, 1, 2\}$), assign color α to the corresponding node w_i, $1 \leq i \leq n$. It can be verified that this is a valid 3-coloring of H, and this completes the proof. □

Our next result shows that the ITSC-NE problem is fixed parameter tractable with respect to the number of inequality constraints. Let $k = |\Gamma_{NE}|$ and let V_1 denote the set of nodes which appear in at least one constraint in Γ_{NE}. Since the number of constraints is k, we have $|V_1| \leq 2k$. The nodes in $V - V_1$ are not involved in any of the constraints in Γ_{NE}.

As in the algorithm for ITSC-EQ, the set of configurations Q (which must be made stable by the chosen threshold assignment) can be used to construct lower (t_v^{low}) and upper (t_v^{high}) bounds on the threshold value of each node $v \in V$. If there is a node v for which $t_v^{low} > t_v^{high}$, then there is no solution to the ITSC-NE instance. So, for the remainder of this discussion, we assume that for each node v, $t_v^{low} \leq t_v^{high}$.

To solve the ITSC-NE problem, we need to determine whether a threshold value for each node $v \in V_1$ can be chosen so that the constraints in Γ_{NE} are also satisfied. Consider the graph $G_\Gamma(V_1, E_1)$ whose node set is V_1 and whose edge set E_1 represents the constraints in Γ_{NE}; that is, for each constraint $t_{v_i} \neq t_{v_j}$ in Γ_{NE}, E_1 has the edge $\{v_i, v_j\}$. For each node $v \in V_1$, let L_v denote the list of integer values in the (closed) interval $[t_v^{low} \,..\, t_v^{high}]$; thus, L_v is the list of all allowable threshold values for v, as determined by the set of configurations Q. Thus, our goal is to determine whether for each node $v \in V_1$, a value can be chosen from its list L_v so that no pair of adjacent nodes in G_Γ has the same value. This is the well known **list coloring problem** in graph theory [59].

Since $|E_1| = k$, the maximum degree of any node in G_Γ is k. Let V_1' denote the set of all nodes v of G_Γ such that L_v contains $k+1$ or more values. We note that each node $v \in V_1'$ (and the edges incident on v) can be removed from G_Γ. This is because after obtaining a valid list coloring for the nodes in $V_1 - V_1'$, for each node $v \in V_1'$, a threshold value α from L_v can be chosen so that α differs from the values chosen for the (at most k) neighbors of v in G_Γ. Let G_Γ^1 denote the subgraph of G_Γ induced on $V_1 - V_1'$. Thus, the list for each node in G_Γ^1 has at most k values. Since G_Γ^1 has at most $2k$ nodes and each list has at most k values, we need to try only k^{2k} combinations of values in an exhaustive search to determine whether there is a valid list coloring of G_Γ^1; if none, there is no solution to the ITSC-NE instance. Otherwise, any valid list coloring of G_Γ^1 can be extended to a valid list coloring of G_Γ. This coloring of G_Γ along with a choice of threshold values for nodes in $V - V_1$ (i.e., nodes which do not appear in any constraint) constitutes a solution to the ITSC-NE instance. The above discussion leads to the outline of our algorithm for the ITSC-NE problem presented in Fig. 7.

Theorem 17. *The ITSC-NE problem is fixed parameter tractable with respect to the number of inequality constraints.*

Proof. The correctness of the algorithm for the ITSC-NE problem presented in Fig. 7 follows from the above discussion. We now show that the running time has the form $O(f(k) + N^{O(1)})$, where N is the size of the problem instance and the function f depends only on k, the number of inequality constraints. In the ensuing discussion, we refer to the steps of the algorithm presented in Fig. 7.

Step 1 can be done in $O(n^2|Q|)$ time by going through each configuration in Q and finding the number of appropriate input values for each of the n nodes of the SyDS $\mathcal{S}$ (as discussed in the proof of Theorem 3). Step 2 can be done in $O(n)$

```
Input: Graph $G(V,E)$ of a SyDS $\mathcal{S}$, a set $Q$ of configurations and a collection
$\Gamma_{EQ}$ of constraints of the form $t_{v_i} = t_{v_j}$. Let $q = |Q|$.

Requirement: Output "Yes" if there is a threshold value $t_v$ for each $v \in V$ such
that in the resulting SyDS, all the configurations in $Q$ are unstable and all the
constraints in $\Gamma_{EQ}$ are satisfied. Otherwise, output "No".

Steps:
1. Find the partition $P = \{V_1, V_2, \ldots, V_r\}$ of $V$ using the equivalence relation
   $\Gamma_{EQ}$. (Nodes which do not appear in any constraint form singleton blocks in
   $P$.)
2. if $q < 2^r$ then output "Yes" and stop.
3. Comment: Here, $q \geq 2^r$.
   (a) Consider each combination $\beta = (\alpha_1, \alpha_2, \ldots, \alpha_r)$ of threshold values to the
       $r$ blocks, where $\alpha_i$ is the value for block $i$, $1 \leq i \leq r$.
   (b) If some combination $\beta$ makes each configuration in $Q$ unstable,
       output "Yes"; otherwise, output "No".
```

Fig. 8. A quasi-polynomial time algorithm for ITUC-EQ.

time. Step 3 can be done in $O(k)$ time by going through each constraint in Γ_{NE}. Since $k = O(n^2)$, the time for Step 3 is $O(n^2)$. Since the size of the list L_v for each node v is $O(n)$, Step 4 and Step 5 can be done in time $O(nk) = O(n^3)$. So, the time for Steps 1 through 5 is $O(n^3 + n^2|Q|) = O(N^3)$, where N is the size of the ITSC-NE instance.

Step 6 checks all combinations of values in the lists for the nodes of G^1_Γ. Since G^1_Γ has at most $2k$ nodes and each list has at most k values, the number of possible combinations is k^{2k}. For each combination, we can determine whether the resulting coloring of G^1_Γ is valid in $O(k)$ time since G^1_Γ has at most k edges. Therefore, Step 6 can be implemented to run in $O(k^{2k+1})$ time.

Hence, the overall running time of the algorithm is $O(k^{2k+1} + N^3)$; that is, the ITSC-NE problem is fixed parameter tractable with respect to k. □

5.3. Inferring thresholds with unstable configurations and constraints

In this section, we focus on threshold inference problems given a set of unstable configurations along with constraints. We show that the ITUC-EQ problem can be solved in quasi-polynomial time. We also show that, in general, ITUC-NE is **NP**-complete. However, when the number of inequality constraints is *fixed*, we again show that ITUC-NE can be solved in quasi-polynomial time.

5.3.1. An algorithm for ITUC-EQ

We begin by discussing the ideas behind the algorithm. As in the case of the algorithm for ITSC-EQ (Section 5.2), we first construct a partition $\{V_1, V_2, \ldots, V_r\}$ of the node set V using the constraints in Γ_{EQ}. In any solution to the ITUC-EQ instance, all nodes in block V_i must have the same threshold value, $1 \leq i \leq r$.

We say that a configuration $\mathcal{C}$ of $\mathcal{S}$ is **block uniform** if the following condition holds: for each block V_i, each node in V_i has the same state value in $\mathcal{C}$, $1 \leq i \leq r$. Since there are r blocks, the number of block uniform configurations of $\mathcal{S}$ is 2^r.

Let $Q = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_q\}$, where $q = |Q|$. We first consider the case where $q < 2^r$. In this case, at least one block uniform configuration, say $\mathcal{C}$, does *not* appear in Q. We note that such a configuration can be found in polynomial time by first sorting the configurations of Q using radix sort [53] and carrying out a simple linear scan of the resulting sorted list of configurations. If we make $\mathcal{C}$ the successor of each configuration in Q, then each configuration in Q is rendered unstable. Since $\mathcal{C}$ is block uniform, we can achieve this and ensure that all the nodes in each block V_i have the same threshold value by the following method (which is similar to the one used in the proof of Lemma 2 in Section 2): for each block V_i, if all the nodes in V_i have state value 1 in $\mathcal{C}$, set the threshold of each node in V_i to 0; otherwise, set the threshold of each node $v \in V_i$ to $d_v + 2$ (which corresponds to the value "infinity"). Thus, when $q < 2^r$, there is always a solution to the ITUC-EQ instance.

Now, consider the other case where $|Q| = q \geq 2^r$. Since Q is part of the input, in this case, the size N of the ITUC-EQ instance is at least 2^r. For any block V_i, the possible threshold values are in the interval $[0 \,..\, n+1]$; thus, the number of possible values is at most $n+2$. Since there are r blocks, the number of combinations of threshold values for all the blocks is at most $(n+2)^r$. Our algorithm carries out an exhaustive search over all these combinations of threshold values. If some combination makes each configuration in Q unstable, then there is a solution to the ITUC-EQ instance; otherwise, there is no solution. In the proof of Theorem 18, we will show that the running time for this case is $O(N^{c \log N})$ for some constant c.

An outline of our algorithm for ITUC-EQ appears in Fig. 8. Correctness of the algorithm follows from the above discussion. The following theorem establishes the running time of the algorithm.

Theorem 18. *The ITUC-EQ problem can be solved in quasi-polynomial time. More specifically, the running time of the algorithm in Fig. 8 is $O(N^{c \log N})$ where N is the size of the problem instance and c is a constant.*

Proof. When $|Q| < 2^r$, the running time of the algorithm in Fig. 8 can be seen to be a polynomial in N, the size of the problem instance. So, we need to consider only the case when $|Q| \geq 2^r$. We show that the running time in this case is $O(N^{c \log N})$ for some constant c.

There are r blocks, and the number of threshold values to be considered for each block is at most $n+2$. Thus, the number of combinations of threshold values to be considered is at most $(n+2)^r = O(N^r)$. For each combination, we can determine in $O(|Q|n^2) = O(N^3)$ time whether the combination causes each configuration in Q to be unstable. Therefore, the total time spent in Step 3(b) of the algorithm is $O(N^{r+3})$. Since $|Q| \geq 2^r$, we have $N \geq 2^r$ or $r \leq \log N$. Therefore, the overall running time is $O(N^{\log N + 3})$ which is $O(N^{c \log N})$ for a constant c. Since $O(N^{c \log N}) = 2^{O((\log N)^2)}$, the running time is quasi-polynomial. □

5.4. Results for ITUC-NE

We first show that ITUC-NE is **NP**-complete through a reduction similar to the one used in the proof of Proposition 16.

Proposition 19. *The ITUC-NE problem is* ***NP****-complete.*

Proof. It is easy to verify that ITUC-NE is in **NP**. We establish **NP**-hardness using a reduction from 3-Coloring. Let the graph $H(V_H, E_H)$ represent the given instance of the 3-Coloring problem. Let $V_H = \{w_1, w_2, \ldots, w_n\}$ so that $|V_H| = n$. We construct an instance of the ITUC-NE problem as follows. For each node $w_i \in V_H$, the underlying graph $G(V, E)$ of the SyDS $\mathcal{S}$ has one corresponding node denoted by v_i, $1 \leq i \leq n$. Thus, $V = \{v_1, v_2, \ldots, v_n\}$. The edge set E of G is empty. The only configuration $\mathcal{C}'$ of $\mathcal{S}$ that must be made unstable has the state value 0 for all the nodes of G. For each edge $\{w_i, w_j\} \in E_H$, the constraint $t_{v_i} \neq t_{v_j}$ is added to the set Γ_{NE} of inequality constraints. This completes the construction, which can obviously be carried out in polynomial time.

Suppose H is 3-colorable. Let the three colors be denoted by 0, 1 and 2. Further, let V_H^i denote the color class i (i.e., the set of nodes of H assigned the color i), $i = 0, 1, 2$. Since H is assumed to require at least three colors, each of these color classes is nonempty. Consider the following threshold assignment: all the nodes of V corresponding to the color class V_H^i are assigned the threshold value i, $i = 0, 1, 2$. It is easy to see that this assignment satisfies all the constraints in Γ_{NE}. Further, since at least one node of G is assigned the threshold value 0, $\mathcal{C}'$ is an unstable configuration. Thus, we have a solution to the ITUC-NE instance.

Now, suppose there is a solution to the ITUC-NE instance. Since the degree of each node in G is zero, the solution must assign the threshold value 0, 1 or 2 to each node of G. Consider the following assignment of colors to the nodes of H: if node v_i is assigned threshold α (for some $\alpha \in \{0, 1, 2\}$), assign color α to the corresponding node w_i, $1 \leq i \leq n$. It can be verified that this is a valid 3-coloring of H, and this completes the proof. □

We now show that when the number of inequality constraints is *fixed*, the ITUC-NE problem can be solved in quasi-polynomial time. The approach is similar to the one used for the ITUC-EQ problem. Let $k = |\Gamma_{NE}|$ and let V_1 denote the subset of nodes which appear in one or more constraints in Γ_{NE}. Let $V_2 = V - V_1$ be the set of nodes which are not involved in any constraint. Let $p = |V_1|$; thus, $|V_2| = n - p$. As before, we know that $p \leq 2k$. As a first step towards our algorithm for the ITUC-NE problem, we have the following simple lemma.

Lemma 20. *The problem of determining whether there is a threshold assignment to the nodes of V_1 that satisfies all the constraints in Γ_{NE} is in the complexity class* ***XP****, with $k = |\Gamma_{NE}|$ as the parameter. In particular, the problem can be solved in $O(kn^{2k})$ time, which is polynomial when k is fixed.*

Proof. A straightforward algorithm with a running time of $O(kn^{2k})$ can be obtained as follows. For each node in V_1, there are only $O(n)$ possible threshold values. Since $|V_1| \leq 2k$, the number of combinations of threshold values to the nodes in V_1 is $O(n^{2k})$. For each combination, checking whether all the constraints in Γ_{NE} are satisfied can be done in $O(k)$ time. Therefore, the overall time is $O(kn^{2k})$, which is a polynomial since k is fixed. Since $O(kn^{2k}) = O(n^{2k+1})$, the running time is of the form $O(n^{f(k)})$, where $f(k)$ depends only on k. Thus, the problem is in **XP** [55]. □

Our algorithm proceeds by considering two cases. For the first case, we let $q = |Q| < 2^{n-p}$. For each configuration $\mathcal{C}_j \in Q$, let $\mathcal{C}'_j$ be the subconfiguration obtained by considering only the state values of nodes in V_2, $1 \leq j \leq q$. Let $Q' = \{\mathcal{C}'_1, \mathcal{C}'_2, \ldots, \mathcal{C}'_q\}$. Let Π denote the set of all 2^{n-p} subconfigurations of $\mathcal{S}$ obtained by assigning state values 0 or 1 to the $n-p$ nodes of V_2. Since $|Q| < |\Pi|$, there is at least one subconfiguration $\mathcal{C}'$ in Π which does not appear in Q'. As in the algorithm for ITUC-EQ, such a subconfiguration can be found in polynomial time by first constructing Q' and sorting the subconfigurations in Q'. The subconfiguration $\mathcal{C}'$ can be extended into a configuration $\mathcal{C}$ of $\mathcal{S}$ by choosing arbitrary values for the nodes in V_1. As in the algorithm for ITUC-EQ, we can make $\mathcal{C}$ the successor for all the configurations in Q by choosing appropriate threshold values for the nodes in V_2 (which are not involved in any constraint). This ensures that each configuration in Q is unstable. Therefore, there is a solution to the ITUC-EQ instance if and only if there is a threshold

```
Input: Graph $G(V, E)$ of a SyDS $\mathcal{S}$, a set $Q$ of configurations and a collection
$\Gamma_{NE}$ of constraints of the form $t_{v_i} \neq t_{v_j}$. Let $q = |Q|$.

Requirement: Output "Yes" if there is a threshold value $t_v$ for each $v \in V$ such
that in the resulting SyDS, all the configurations in $Q$ are unstable and all the
constraints in $\Gamma_{NE}$ are satisfied. Otherwise, output "No".

Steps:
1. Let $V_1$ be the set of nodes that appear in one or more constraints of $\Gamma_{NE}$. Let
   $p = |V_1|$ and let $V_2 = V - V_1$.
2. Case 1: $q < 2^{n-p}$.
   (a) Use the algorithm presented in the proof of Lemma 20 to check whether
       there is a threshold assignment to the nodes in $V_1$ that satisfies all the
       constraints in $\Gamma_{NE}$.
   (b) If a threshold assignment is found in Step 2(a), then output "Yes" and
       stop; otherwise, output "No" and stop.
3. Case 2: Here, $q \geq 2^{n-p}$.
   (a) Consider each combination $\beta = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ of threshold values to the
       $n$ nodes, where $\alpha_i$ is the value for node $v_i$, $1 \leq i \leq n$. (Note that $0 \leq \alpha_i \leq$
       $n+1$, $1 \leq i \leq n$.)
   (b) If some combination $\beta$ makes each configuration in $Q$ unstable and sat-
       isfies all the constraints in $\Gamma_{NE}$, output "Yes"; otherwise, output "No".
```

Fig. 9. A quasi-polynomial time algorithm for ITUC-NE when the number of constraints is fixed.

assignment for the nodes in V_1 that satisfies all the constraints in Γ_{NE}. From Lemma 20, the latter task can be done in polynomial time. Therefore, when $|Q| < 2^{n-p}$, the algorithm runs in polynomial time when k is fixed.

For the second case, we have $|Q| \geq 2^{n-p}$. Therefore, in this case, the size of the problem instance N is at least 2^{n-p}. The algorithm uses a simple exhaustive search over all the $O(n^n)$ possible assignments of threshold values. As in the case of the algorithm for ITSC-EQ, this running time can be shown to be quasi-polynomial in the size of the problem instance.

An outline of our algorithm for ITUC-NE for the case when $|\Gamma_{NE}|$ is fixed appears in Fig. 9. Since the correctness of the algorithm follows from the above discussion, the following theorem establishes that the running time is quasi-polynomial.

Theorem 21. *When the number of inequality constraints is* fixed, *the ITUC-NE problem can be solved in time $O(N^{c \log N})$ time, where N is the size of the problem instance and c is a constant.*

Proof. Let $k = |\Gamma_{NE}|$ and let p be the number of nodes appearing in one or more constraints. We note that $p \leq 2k$.

From the preceding discussion, when $|Q| < 2^{n-p}$, the running time of the algorithm in Fig. 9 can be seen to be a polynomial in N, the size of the problem instance. When $|Q| \geq 2^{n-p}$, we show that the running time is $O(N^{c \log N})$ for some constant c.

There are n nodes, and the number of threshold values to be considered for each node is at most $n+2$. Thus, the number of combinations of threshold values to be considered is at most $(n+2)^n = O(N^n)$. For each combination, we can determine in $O(|Q|n^2) = O(N^3)$ time whether the combination causes each configuration in Q to be unstable. Therefore, the total time spent in Step 3 of the algorithm is $O(N^{n+3})$. Since $|Q| \geq 2^{n-p}$, we have $N \geq 2^{n-p}$ or $n \leq \log_2 N + p$. Since $p \leq 2k$, we have $n \leq \log_2 N + 2k$. Therefore, the overall running time is $O(N^{\log N + 3 + 2k})$ which is $O(N^{c \log N})$ for a constant c since k is fixed. □

6. Summary and future research directions

We considered many versions of threshold inference problems for deterministic SyDSs and presented complexity results, fixed parameter tractability results as well as polynomial and quasi-polynomial time algorithms. We conclude by mentioning some specific open problems and a few general directions for future work.

It is of interest to investigate whether the running times of our algorithms that show the fixed parameter tractability of some inference problems can be improved. Alternatively, it may also be of interest to investigate whether lower bounds can be established on the running times under some complexity theoretic assumptions such as strong exponential time hypothesis [60,61]. When considering constrained versions of threshold inference problems, we obtained quasi-polynomial time algorithms for two problems, namely ITUC-EQ and ITUC-NE when the number of constraints is fixed. It is of interest to investigate whether there are polynomial time algorithms for these problems.

Our work also suggests several general directions for future work. One direction is to consider inference problems for other forms of observed behavior such as a collection of snapshots of the system, where each snapshot specifies a time and the configuration of the system at that time. A second direction is to consider inference problems assuming more powerful local functions. Finally, it is also of interest to consider inference problems for stochastic SyDSs, whose local transition functions are probabilistic threshold functions.

Acknowledgements

It is our pleasure to thank the reviewer for carefully reading the manuscript and providing valuable suggestions. This work has been partially supported by DTRA Grant HDTRA1-11-1-0016 and DTRA CNIMS Contract HDTRA1-11-D-0016-0010, NSF NetSE Grant CNS-1011769, NSF SDCI Grant OCI-1032677 and NIH MIDAS Grant 5U01GM070694-11.

References

[1] K. Lum, S. Swarup, S. Eubank, J. Hawdon, The contagious nature of imprisonment: an agent-based model to explain racial disparities in incarceration rates, J. R. Soc. Interface 11 (98) (2014), arXiv:2014.0409.

[2] M.E. Halloran, N.M. Ferguson, S. Eubank, I.M. Longini, D.A. Cummings, B. Lewis, S. Xu, C. Fraser, A. Vullikanti, T.C. Germann, et al., Modeling targeted layered containment of an influenza pandemic in the United States, Proc. Natl. Acad. Sci. USA 105 (12) (2008) 4639–4644.

[3] O. Diekmann, H. Heesterbeek, T. Britton, Mathematical Tools for Understanding Infectious Disease Dynamics, Princeton University Press, 2012.

[4] D. Gruhl, R. Guha, D. Liben-Nowell, A. Tomkins, Information diffusion through blogspace, in: Proceedings of the 13th International Conference on World Wide Web, ACM, 2004, pp. 491–501.

[5] B.A. Prakash, D. Chakrabarti, N.C. Valler, M. Faloutsos, C. Faloutsos, Threshold conditions for arbitrary cascade models on arbitrary networks, Knowl. Inf. Syst. 33 (3) (2012) 549–575.

[6] M. Granovetter, Threshold models of collective behavior, Amer. J. Sociol. (1978) 1420–1443.

[7] C.J. Kuhlman, V.A. Kumar, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, Inhibiting diffusion of complex contagions in social networks: theoretical and experimental results, Data Min. Knowl. Discov. 29 (2) (2015) 423–465.

[8] T.G. Trucano, L.P. Swiler, T. Igusa, W.L. Oberkampf, M. Pilch, Calibration, validation, and sensitivity analysis: what's what, Reliab. Eng. Syst. Saf. 91 (10) (2006) 1331–1357.

[9] S. González-Bailón, J. Borge-Holthoefer, A. Rivero, Y. Moreno, The dynamics of protest recruitment through an online network, Sci. Rep. 1 (2011), 7 pages.

[10] D.M. Romero, B. Meeder, J. Kleinberg, Differences in the mechanics of information diffusion across topics: idioms, political hashtags, and complex contagion on twitter, in: Proceedings of the 20th International Conference on World Wide Web, ACM, 2011, pp. 695–704.

[11] J. Ugander, L. Backstrom, C. Marlow, J. Kleinberg, Structural diversity in social contagion, Proc. Natl. Acad. Sci. USA 109 (16) (2012) 5962–5966.

[12] D. Easley, J. Kleinberg, Networks, Crowds, and Markets: Reasoning About a Highly Connected World, Cambridge University Press, 2010.

[13] H. Mortveit, C. Reidys, An Introduction to Sequential Dynamical Systems, Springer Science & Business Media, New York, NY, 2007.

[14] C. Barrett, H.B. Hunt, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, R.E. Stearns, Modeling and analyzing social network dynamics using stochastic discrete graphical dynamical systems, Theoret. Comput. Sci. 412 (30) (2011) 3932–3946.

[15] J. Crane, The epidemic theory of ghettos and neighborhood effects on dropping out and teenage childbearing, Amer. J. Sociol. (1991) 1226–1259.

[16] A. Gaviria, S. Raphael, School-based peer effects and juvenile behavior, Rev. Econ. Stat. 83 (2) (2001) 257–268.

[17] Y.J. Xu, Advance to and persistence in graduate school: identifying the influential factors and major-based differences, J. Coll. Stud. Ret., Res. Theory Pract. 16 (3) (2014) 391–417.

[18] M.H.W. v. Zalk, M. Kerr, S.J. Branje, H. Stattin, W.H. Meeus, Peer contagion and adolescent depression: the role of failure anticipation, J. Clin. Child Adolesc. Psychol. 39 (6) (2010) 837–848.

[19] E.A. Stevens, M.J. Prinstein, Peer contagion of depressogenic attributional styles among adolescents: a longitudinal study, J. Abnorm. Child Psychol. 33 (1) (2005) 25–37.

[20] C. De la Higuera, Grammatical Inference: Learning Automata and Grammars, Cambridge University Press, 2010.

[21] J. Heinz, C. De la Higuera, M. van Zaanen, Grammatical inference for computational linguistics, Synth. Lect. Hum. Lang. Technol. 8 (4) (2015) 1–139.

[22] K.P. Murphy, Passively learning finite automata, Tech. rep. 96-04-017, Santa Fe Institute, Santa Fe, NM, 1996.

[23] M.J. Kearns, U.V. Vazirani, An Introduction to Computational Learning Theory, MIT Press, Cambridge, MA, 1994.

[24] M.W. Macy, R. Willer, From factors to actors: computational sociology and agent-based modeling, Annu. Rev. Sociol. 28 (2002) 143–166.

[25] C. Barrett, H.B. Hunt, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, R.E. Stearns, M. Thakur, Predecessor existence problems for finite discrete dynamical systems, Theoret. Comput. Sci. 386 (1) (2007) 3–37.

[26] F. Green, NP-complete problems in cellular automata, Complex Systems 1 (3) (1987) 453–474.

[27] C.L. Barrett, H.B. Hunt, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, R.E. Stearns, Complexity of reachability problems for finite discrete dynamical systems, J. Comput. System Sci. 72 (8) (2006) 1317–1345.

[28] S. Kosub, C.M. Homan, Dichotomy results for fixed point counting in Boolean dynamical systems, in: Proceedings of the 10th Italian Conference on Theoretical Computer Science, 2007, pp. 163–174.

[29] K. Sutner, Computational classification of cellular automata, Int. J. Gen. Syst. 41 (6) (2012) 595–607.

[30] B. Abrahao, F. Chierichetti, R. Kleinberg, A. Panconesi, Trace complexity of network inference, in: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2013, pp. 491–499.

[31] M. Gomez Rodriguez, J. Leskovec, A. Krause, Inferring networks of diffusion and influence, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2010, pp. 1019–1028.

[32] S. Soundarajan, J.E. Hopcroft, Recovering social networks from contagion information, in: Proceedings of the 7th Annual Conference on Theory and Models of Computation, Springer, 2010, pp. 419–430.

[33] D. Shah, T. Zaman, Rumors in a network: who's the culprit?, IEEE Trans. Inform. Theory 57 (8) (2011) 5163–5181.

[34] J. Kleinberg, Cascading behavior in networks: algorithmic and economic issues, in: Algorithmic Game Theory, Cambridge University Press, UK, 2007, pp. 613–632, Ch. 24.

[35] E. Goles, S. Martínez, Neural and Automata Networks: Dynamical Behavior and Applications, Kluwer, Dordrecht, The Netherlands, 1990.

[36] B. Durand, A random NP-complete problem for inversion of 2D cellular automata, Theoret. Comput. Sci. 148 (1) (1995) 19–32.

[37] C.L. Barrett, H.B. Hunt III, M.V. Marathe, S. Ravi, D.J. Rosenkrantz, R.E. Stearns, P.T. Tosic, Gardens of Eden and fixed points in sequential dynamical systems, in: Proceedings of the Discrete Mathematics and Theoretical Computer Science Conference, 2001, pp. 95–110.

[38] D. Kempe, J. Kleinberg, E. Tardos, Maximizing the spread of influence through a social network, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 137–146.

[39] C. Bazgan, M. Chopin, A. Nichterlein, F. Sikora, Parameterized approximability of maximizing the spread of influence in networks, J. Discrete Algorithms 27 (2014) 54–65.

[40] C. Bazgan, M. Chopin, The complexity of finding harmless individuals in social networks, Discrete Optim. 14 (2014) 170–182.

[41] C. Bazgan, M. Chopin, M. Cygan, M.R. Fellows, F.V. Fomin, E.J. van Leeuwen, Parameterized complexity of firefighting, J. Comput. System Sci. 80 (7) (2014) 1285–1297.

[42] F.N. Abu-Khzam, J. Egan, M.R. Fellows, F.A. Rosamond, P. Shaw, On the parameterized complexity of dynamic problems with connectivity constraints, in: Combinatorial Optimization and Applications, Springer, 2014, pp. 625–636.
[43] N. Boria, J. Monnot, V.T. Paschos, Reoptimization under vertex insertion: max P_k-free subgraph and max planar subgraph, Discrete Math. Algorithms Appl. 5 (2) (2013), 25 pages.
[44] T. Ito, E.D. Demaine, N.J. Harvey, C.H. Papadimitriou, M. Sideri, R. Uehara, Y. Uno, On the complexity of reconfiguration problems, in: Proceedings of the 19th International Symposium on Algorithms and Computation, Springer, 2008, pp. 28–39.
[45] H. Fernau, J.A. Rodriguez-Velazquez, A survey on alliances and related parameters in graphs, Electron. J. Graph Theory Appl. 2 (1) (2014) 70–86.
[46] J. Łacki, J. Oćwieja, M. Pilipczuk, P. Sankowski, A. Zych, The power of dynamic distance oracles: efficient dynamic algorithms for the Steiner tree, in: Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, ACM, 2015, pp. 11–20.
[47] J. van den Heuvel, The complexity of change, in: S.R. Blackburn, S. Gerke, M. Wildon (Eds.), Surveys in Combinatorics, in: London Math. Soc. Lecture Note Ser., vol. 409, Cambridge University Press, 2013, pp. 127–160.
[48] M. Berglund, H. Björklund, F. Drewes, On the parameterized complexity of linear context-free rewriting systems, in: Proceedings of the 13th Meeting on the Mathematics of Language, Association for Computational Linguistics, 2013, pp. 21–29.
[49] C.C. Florêncio, H. Fernau, On families of categorical grammars of bounded value, their learnability and related complexity questions, Theoret. Comput. Sci. 452 (2012) 21–38.
[50] R.G. Downey, M.R. Fellows, B.M. Kapron, M.T. Hallett, H.T. Wareham, The parameterized complexity of some problems in logic and linguistics, in: Proceedings of the Third International Symposium on Logical Foundations of Computer Science, in: Lecture Notes in Comput. Sci., vol. 813, Springer, 1994, pp. 89–100.
[51] H. Fernau, P. Heggernes, Y. Villanger, A multi-parameter analysis of hard problems on deterministic finite automata, J. Comput. System Sci. 81 (4) (2015) 747–765.
[52] R. Niedermeier, Invitation to Fixed Parameter Algorithms, Oxford University Press, New York, NY, 2006.
[53] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, second edition, MIT Press and McGraw–Hill, Cambridge, MA, 2009.
[54] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., San Francisco, 1979.
[55] J. Flum, M. Grohe, Parameterized Complexity Theory, Springer, Heidelberg, Germany, 2006.
[56] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$, Acta Math. 182 (1999) 105–142.
[57] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation: Combinatorial Problems and Their Approximability Properties, Springer, Berlin, Germany, 1999.
[58] R. Graham, D. Knuth, O. Patashnik, Concrete Mathematics, Addison–Wesley, Reading, MA, 1994.
[59] D.B. West, Introduction to Graph Theory, Prentice–Hall, Inc., Englewood Cliffs, NJ, 2003.
[60] R. Impagliazzo, R. Paturi, F. Zane, Which problems have strongly exponential complexity?, in: Proceedings of the 39th Annual Symposium on Foundations of Computer Science, IEEE, 1998, pp. 653–662.
[61] D. Lokshtanov, D. Marx, S. Saurabh, Lower bounds based on the exponential time hypothesis, Bull. Eur. Assoc. Theor. Comput. Sci. EATCS 105 (2011) 41–72.