Local Information in Influence Networks

Yuezhou Lv* and Thomas Moscibroda

* Tsinghua University, Microsoft Research
totolv@126.com, *moscitho@microsoft.com*

Abstract. We study how multi-hop information impacts convergence in social influence networks. In influence networks, nodes have a choice between two options A and B, and each node prefers to end up choosing the option that a majority of its neighbors choose. We consider the case of innovation adoption in which nodes can only change from A to B, but not backwards. For this model, we ask the question, when is it safe for a node to switch from A to B? If nodes have multi-hop information about the network, rather than knowing only the state of their immediate neighbors, the answer to this question becomes complex. The reason is that a node needs to recursively reason about what its neighbors know, and whether given their knowledge they will also upgrade to B.
In this paper, we assume that each node has complete knowledge about its k-hop neighborhood, but does not know anything about the network beyond k-hops. We study how different local decision algorithms achieve different properties in terms of *safety* and *conversion ratio* (how many nodes ultimately upgrade to B). We characterize the possible algorithms by classifying them into a *hierarchy of algorithms*. Each class of algorithms in this hierarchy is distinguished by a natural safety property that it guarantees. For each class, we give an optimal algorithm in terms of conversion ratio, and we show that each class is fully contained in the class of lower safety level. Conversely, each lower-safety class can achieve strictly higher conversion ratio than any algorithm in the safer class. Thus, our hierarchy reveals a strict trade-off between safety and conversion ratio. Finally, we show that each class of algorithms satisfies two natural closure properties.

Keywords: Influence Networks, Multi-hop Information, Hierarchy of Algorithms, Distributed Algorithms

1 Introduction

Influence networks in all their variants are important in the study of many natural phenomena. In an influence network, each node is an agent and its action in some round T, depends on the state of its neighbors. Influence network models have been used in the study of diffusion of innovation, social networks, belief propagation, spring embedders, cellular automata, traffic networks, biological cell systems, etc.

In studies on influence networks, it has been implicitly assumed that not only does the *utility* of a node depend on the state of its neighbors, but – importantly – that each

* This work was supported in part by the National Basic Research Program of China Grant 2011CBA00300, 2011CBA00302, the National Natural Science Foundation of China Grant 61073174, 61033001, 61061130540, and the Hi-Tech research and Development Program of China Grant 2006AA10Z216.

node *only knows the state of its immediate neighbors*. That is, it is assumed that nodes only have information about their neighborhood, or that they do not make sophisticated use of any additional information about the state of nodes outside their 1-hop neighborhood. However, having such additional non-local information can be of critical importance both in terms of convergence speed (i.e., how fast the system converges to its final equilibrium) as well as in terms of convergence ratio (i.e., the overall utility or social welfare achieved by the system in the equilibrium). Consider the following scenario, which we will use as a baseline for our model in this paper: Assume that each node in a network represents a participant that has a choice between staying with a current, older operating system, or upgrading to a newer version of the operating system. Some participants may be early adopters and decide to upgrade regardless of the actions of its neighbors; while others may be wary of change and never want to upgrade. For most participants, however, the utility is such that they will upgrade if the majority of their neighbors also upgrade; and they will stay if the majority of their neighbors stay.

The question is, for such a regular participant – when is it safe to upgrade to the new operating system? If every node only knows its neighbors, (i.e., without any knowledge about its neighbors' neighbors), then it is possible that all nodes will "wait" until a majority of its neighbors have decided to upgrade, and since every node behaves in this same way, no node will ever upgrade. That is, the system is stuck in a suboptimal configuration, simply due to the constraint of having only local information. If, however, each node knows about the state of nodes in its 2-hop or 3-hop neighborhood, then a node could compute locally that it is safe to upgrade, because it can be sure that a majority of its neighbors will ultimately benefit from upgrading, and thus they *will* upgrade. Thus, simply by increasing the amount of local information, the entire system ends up in a better global equilibrium.

To make the example concrete, suppose each node has two states $\mathbf{original}$ and $\mathbf{upgraded}$, and each node is better off by changing from $\mathbf{original}$ to $\mathbf{upgraded}$ if and only if *all* its neighbors are upgraded. Consider a line with 5 nodes, $x_1, \ldots, x_5$, in which x_1 and x_5 are upgraded while the others are original. If each node has only 1-hop information, x_2, x_3, x_4 will never upgrade. But, if x_3 has 2-hop information, x_3 can upgrade first, because x_2, x_4 will then also upgrade which finally makes x_3 better off. The same happens in a system with 3 nodes that form a triangle. If every node *knows* that the three nodes form a triangle, all of them can upgrade in one step. However, if the nodes do not know whether their neighbors are mutually connected (they may have an independent neighbor each), then none of the nodes will upgrade.

More generally, the study of how multi-hop information impacts the dynamics and convergence of influence networks and leads to fascinating questions. These questions are of the following nature: A node x would like to upgrade, but doing so is only "safe" (i.e., guaranteed to lead to a higher utility), if its neighbors also upgrade. However, in order for x to know whether its neighbors will upgrade, it must determine whether it is "safe" for them to do so. Thus, the problem is recursive: In order for x to determine whether it is "safe" for its neighbor y to upgrade, it must determine what y knows about its neighbors, i.e., including what y knows about what x knows about y, etc.

Thus, the extent to which a node knows more than simply its immediate neighborhood fundamentally changes how the system behaves. In this paper, we study how

multi-hop local information impacts the convergence of influence networks. We provide a theory of local decision algorithms based on an analysis of the kind of decisions that nodes can safely take, if they are given multi-hop information. Naturally, the answer to the question of "when is it safe to upgrade" fundamentally depends on what each node assumes about the behavior of the other nodes within its vicinity. In other words, whether an action is safe for a node x depends i) on what x knows about its local neighborhood (and recursively , what the nodes in x's neighborhood know about their respective neighborhood, etc), and ii) on the extent to which x can rely on the nodes in its vicinity to behave rationally, trustworthily, conservatively, etc.

We show that these assumptions about the neighbors' behavior imply a natural hierarchy of safety-properties. Each stronger safety property ensures a stronger guarantee that the node's action will be the right choice in the equilibrium. Intriguingly, this hierarchy of safety properties also implies a *hierarchy of local decision algorithms*, and every local decision algorithm can be categorized into one of the classes of the hierarchy.

What is fascinating about this hierarchy of local decision algorithm is that each of its classes is defined along a natural safety concept, satisfies closure properties, and is strictly separated from both the next higher (=safer) class and the next lower (=less safe) class. Specifically, we prove the following results:

- The class of algorithms $\mathcal{C}_i$ is a proper superset of every safer class $\mathcal{C}_j$, $i > j$.
- For every class $\mathcal{C}_i$, we derive the optimal algorithm $OPT(\mathcal{C}_i)$ among all local decision algorithms contained in this class. Here, optimal is with regard to the social welfare (conversion ratio) achieved in the final equilibrium.
- We prove that for every class $\mathcal{C}_i$, its optimal algorithm $OPT(\mathcal{C}_i)$ can achieve a strictly better conversion ratio than any local decision algorithm contained in any safer class $\mathcal{C}_j$, $i > j$. That is, each class in the hierarchy is strictly more efficient in terms of conversion ratio. Thus, the hierarchy captures the trade-off between safety and resulting social welfare.
- Finally, we prove that three of the classes ($\mathcal{C}_{Byz}$, $\mathcal{C}_{Rat}$, $\mathcal{C}_{Pos}$) satisfy two natural closure properties: i) *subset-closure* (every subset of an algorithm in the class is included in the class), and ii) *union-closure* (the union of two algorithms in the class is included in the class). For the fourth class, $\mathcal{C}_{Pro}$, the properties do not hold in general, but only for an important subclass of $\mathcal{C}_{Pro}$.

In some classes of the hierarchy, the optimal algorithm is simple and natural, while for others (e.g., $\mathcal{C}_{Pro}$), the optimal algorithm exhibits a complicated recursive structure that may be of independent interest.

1.1 Related Work

There is vast literature on graph-based problems in which a node's decision depends on its neighbors' state, including famous examples such as the "game-of-life" simulations [4]; the classic "democrats-vs-republicans" problem [21]; influence maximization problems [2] [8][9][10], or spread (diffusion) of innovation problems[1][16][19]. The process of local majority voting in graphs, and its basic properties has been reviewed in [14]. In economics, this class of problems can be regarded generically as binary decisions with externalities [17]. It has been widely used in sociology and economics[11][12][18][22]. Models in which nodes can only change from "inactive" to

"active" have been studied in [7][13][19][20]. The linear threshold model proposed in [8][9] is more closely related to ours. In contrast, an alternative setting allows nodes to change states freely, leading to problems such as stability, periodicity [5][6][15] or convergence time [3]. In all these works, the information used by nodes for decision making is restricted to the nodes' immediate neighborhood, i.e., no multi-hop information.

2 Model and Definition

2.1 Influence Network Model

We model an Influence Network as a graph $G = (V, E, \Phi)$. Two nodes connected by an edge are *neighbors* – their utility determines each other. A node $x \in V$ is in one of two possible states: the "original" state A or the "upgraded" state B. A node has a type $\Phi(x)$: stubborn (unchangeable) or changeable. A *stubborn node* always remains in state A. A *changeable node* can change to B. Once a node has upgraded to B, it cannot switch back to A. A special case among the changeable nodes are *early-adopters*, i.e., nodes that start in state B at the beginning. All other (regular) changeable nodes start in state A and switch to B under certain conditions as described below. We denote a changeable node initially in state A as a *regular node*.

The network evolves over a series of rounds. We write $\gamma_T(x)$, $\gamma_T(S)$, and $\gamma_T(G)$ to denote the state of a node x, a set of nodes S, and the state of the network G in round T. At the beginning, the network can include nodes in both state A and B (early-adopters). The evolving *process* on network G is the sequence of network states $\gamma_0(G), \gamma_1(G), ...$ The network is in an equilibrium, if there is no further change happening in the process, i.e., no node changes its state.

Definition 1. *A network G is in an* equilibrium *in round T iff $\gamma_T(G) = \gamma_{T+1}(G)$.*

The next definition captures whether a regular node is *stable*, i.e., whether it is satisfied with its current state. Let $N(x)$ be the set of neighbors of x, and $N_T^B(x)$ be the set of neighbors in state B. For a given threshold $q \in [0, 1]$, if at least a q-fraction of a node x's neighbors are in state B, then x is stable if it is also in state B, otherwise it is stable if it is in state A.

Definition 2. *A regular node x is* stable *in round T if $\gamma_T(x) = B$ and $|N_T^B(x)|/|N(x)| \geq q$, or if $\gamma_T(x) = A$ and $|N_T^B(x)|/|N(x)| < q$.*

2.2 *k*-hop Influence Network & Local Decision Algorithm

In this paper, we extend the basic 1-hop influence network setting to a multi-hop setting. In a *k-hop influence network*, each node has complete information about the topology, and the state and type of all nodes up to a distance of k in the network. Let $V_k(x), E_k(x)$ be the set of nodes and edges within the k-hop neighborhood of node x. The *view* of a node x is defined as follows:

Definition 3. *In a k-hop network, the* view *$\Gamma(x)$ of node x is a 4-tuple $\Gamma(x) = (V_k(x), E_k(x), \Phi(V_k(x)), \gamma(V_k(x)))$.*

Local Decision Algorithm: The only information available to a node is its view. Therefore, the decision of whether a regular node upgrades from state A to state B in a round

T depends entirely on its current view. Therefore, we can define a *Local Decision Algorithm* (or short, algorithm) as a mapping of a node's view to a binary decision, whether or not the node changes its state from A to B. That is, an algorithm can be seen as defining for which views a regular node decides to upgrade. With this in mind, we characterize a Local Decision Algorithm as *the set of views that lead the node to upgrade*. That is, an algorithm is equivalent to a set of views, called the *Changing View Set* S_{ALG}, which cause a regular node x in state A to upgrade to state B. Thus, in this paper, we reason about changing view sets when defining properties of algorithms.

Every node's decisions are based on its Local Decision Algorithm. We denote by $\Pi = (S_1, \ldots, S_n)$ the set of algorithms of all regular nodes $x_1, \ldots, x_n$. Let $\Pi_{x_i} = S_i$, and $\Pi_{-x_i} = (S_1, \ldots, S_{i-1}, S_{i+1}, \ldots, S_n)$. We use upper script to indicate that every regular node executes the same algorithm, e.g., $\Pi^{ALG} = (S_{ALG}, \ldots, S_{ALG})$. Since a process depends only on the initial state $\gamma(G)$ and the algorithms of each regular node, we can use the notation $P[\gamma(G), \Pi]$ to characterize a process.

We use the natural notion of social welfare to evaluate the performance of algorithms: How many nodes end up upgrading to state B in the equilibrium. We call this metric the *conversion ratio* $|\Omega|$, where the conversion set Ω is the set of all nodes in state B in the equilibrium. When comparing algorithms in terms of their conversion ratio, we use the following three definitions.

Definition 4. *An algorithm S_α is (strictly) more efficient than an algorithm S_β in some network setting $\gamma(G)$ if in $P_\alpha[\gamma(G), \Pi^{S_\alpha}]$ and $P_\beta[\gamma(G), \Pi^{S_\beta}]$, it holds that $|\Omega^{P_\alpha}| \geq |\Omega^{P_\beta}|$ ($|\Omega^{P_\alpha}| > |\Omega^{P_\beta}|$).*

Definition 5. *We define $S_\alpha \succeq S_\beta$ ($S_\alpha \succ S_\beta$) if S_α is more efficient than S_β in **any** network setting (and is strictly more efficient than S_β in at least one network setting).*

Definition 6. *S is* optimal *within a class of algorithms $\mathcal{C}$, denoted $S = OPT(\mathcal{C})$, if $S \in \mathcal{C}$ and for any algorithm $S_\alpha \in \mathcal{C}$, it holds that $S \succeq S_\alpha$.*

3 Hierarchy of Algorithms

In this section, we construct a hierarchy of local decision algorithms. In a k-hop influence network, two key factors that determine a node's decision making are i) what it can assume about its neighbor's behavior (i.e., its neighbors' algorithms); and ii) how much "risk" the node is willing to take; how surely the node expects to end up in a stable configuration in the equilibrium (i.e., a node's safety guarantee). There is a fundamental tradeoff between these two factors. The more likely a node upgrades to B based on a belief that some of its neighbors will also upgrade to B in subsequent rounds, the more opportunity the node itself has to take the plunge and upgrade itself, and thus the higher the global conversion ratio will be. Consider two extreme local decision algorithms: In the first algorithm, each regular node in G directly upgrades to B in the very first round. This algorithm clearly leads to the highest possible conversion ratio, but many nodes may end up being unstable in the equilibrium, i.e., they upgrade even though they should not have. At the other extreme end of the spectrum is the standard local 1-hop decision algorithm studied in the existing literature: A node upgrades to B only if a q-fraction of its neighbors are B. This algorithm guarantees stability for every node, but it is inefficient in terms of conversion ratio. In many situations, few nodes will

upgrade, even though all nodes would collectively benefit from doing so. Our hierarchy of local decision algorithms captures this trade-off, showing how different classes of algorithms achieve different natural safety-properties and conversion ratios, depending on the nodes' beliefs about other nodes' algorithms.

Since a local decision algorithm is equivalent to a set of changing views, *any property for an algorithm is entirely a function of its changing view set*. The first property we introduce defines the set of *rational algorithms*. A local decision algorithm is *rational* if it lets a node upgrade when there are a sufficient number of upgraded nodes in its 1-hop neighborhood. That is, an algorithm is rational if it includes all views $\Gamma(x)$ in its changing view set in which $|N^B(x)|/|N(x)| \geq q$ holds.

Definition 7 (Rational). *An algorithm S is* rational *if $S_r \subseteq S$, where $S_r = \{\Gamma(x) || N^B(x)|/|N(x)| \geq q$ holds in $\Gamma(x)\}$.*

Any natural algorithm should be rational as for a regular node, if a q-fraction of its neighbors have upgraded to B, it is sure to be stable by also upgrading. We call the set of all algorithms (including rational and non-rational) *arbitrary*. Our hierarchy includes arbitrary algorithms, but for simplicity we often implicitly assume algorithm to be rational unless we explicitly state that it is arbitrary.

Safety Properties: Local decision algorithms can be characterized according to the safety guarantees they achieve. We define an algorithm to be *safe* if when executing this algorithm, a node ends up being stable in the equilibrium. As a weaker version of safety, we consider the concept of *possible-safety*. A local decision algorithm is *possible-safe* if a node–when executing this algorithm–has a chance to end up being stable in the equilibrium. Following these ideas, we classify safety properties for local decision algorithms in sequence from safest to least safe.

The highest level of safety guarantee is *Byzantine-safety*. An algorithm is *Byzantine-safe* if in any network, when executing this algorithm, a node is guaranteed to end up being stable in the equilibrium, regardless of what arbitrary algorithms the other nodes in the network run. While Byzantine-safe algorithms ensure safety even in the presence of irrationally operating neighbors, nodes running a Byzantine-safe algorithm have few opportunities to upgrade to state B and often get stuck unnecessarily in state A.

Definition 8 (Byzantine-safe). *An algorithm S is* Byzantine-safe *if for any $\gamma(G)$, $x \in G$ and arbitrary Π_{-x}, in process $P[\gamma(G), (\Pi_{-x}, \Pi_x = S)]$, x is stable in the equilibrium.*

The next lower degree of safety is called *Rational-safety*. It is defined similarly, except that each node assumes that the other nodes in the network execute at least a *rational* (instead of arbitrary) algorithm.

Definition 9 (Rational-safe). *An algorithm S is* rational-safe *if for any $\gamma(G)$, $x \in G$ and rational Π_{-x}, in process $P[\gamma(G), (\Pi_{-x}, \Pi_x = S)]$, x is stable in the equilibrium.*

The third level of safety is *Protocol-safety*. A node x executing a protocol-safe local decision algorithm is guaranteed to end up being stable in the equilibrium, under the assumption that all other nodes will execute *the same algorithm* as x does. In other words, protocol-safety captures the safety guarantee we can achieve if every node in the network follows a given distributed protocol designed by some global algorithm designer, and every node faithfully executes this common protocol.

Definition 10 (Protocol-safe). *An algorithm S is* protocol-safe *if for any $\gamma(G)$, $x \in G$, in process $P[\gamma(G), \Pi^S]$, x is stable in the equilibrium.*

Finally, the lowest level of safety is *Possible-safety*. If a node executes a possible-safe algorithm, then there exists a set of specific algorithms for the other nodes such that if each node executes this specific algorithm, every regular node in the graph ends up being stable in the equilibrium. In other words, an algorithm is possible-safe, if there at least *exists a possibility that it leads to all nodes becoming stable in the equilibrium*. Another way to interpret possible-safe local decision algorithms is that these are set of algorithms for "optimists". A node will decide to upgrade, if there is a chance that upgrading will lead to a stable outcome for everyone.

Definition 11 (Possible-safe). *An algorithm S is* possible-safe *if for any $\gamma(G)$, $x \in G$, there exists a Π_{-x} such that in process $P[\gamma(G), (\Pi_{-x}, \Pi_x = S)]$, each regular node is stable in the equilibrium.*

Notice that the definition of Possible-safety rules out trivial algorithms such as the algorithm in which every node always upgrades to B. Indeed, such an algorithm is not possible-safe the neighbors of a node executing such algorithm could all be stubborn.

Thus, we have four classes of algorithms, denoted by $\mathcal{C}_{Byz}, \mathcal{C}_{Rat}, \mathcal{C}_{Pro}$, and $\mathcal{C}_{Pos}$, each of which corresponds to a particular safety-guarantee the algorithm ensures. The following theorem shows that each class of algorithms is entirely contained in the next lower class. A class contains all algorithms that are contained in the next safer class.

Theorem 1. $\mathcal{C}_{Byz} \subseteq \mathcal{C}_{Rat} \subseteq \mathcal{C}_{Pro} \subseteq \mathcal{C}_{Pos}$.

We also study closure properties of the different classes. Specifically, a class of algorithms is *union-closed* if the union of any two algorithms in such class is included in it. A class of algorithms is *subset-closed* if the subset of any algorithm in such class is included in the class.

Definition 12. *A class of algorithms $\mathcal{C}$ is*

1. union-closed if for any $S_\alpha, S_\beta \in \mathcal{C}$, it holds that $(S_\alpha \cup S_\beta) \in \mathcal{C}$.

2. subset-closed if for any $S_\beta \in \mathcal{C}$ and any $S_\alpha \subseteq S_\beta$, it holds that $S_\alpha \in \mathcal{C}$.

4 Algorithms

In the section, we study each class of algorithms. For each class, we derive an algorithm that is optimal among all algorithms within this class. Also, for each class, we verify whether it is subset- and union-closed. Finally, we compare the optimal algorithm in each class in terms of its conversion ratio to all algorithms in the next safer class. Our results imply that the hierarchy is "complete" in the sense that the algorithms in each less safe hierarchy class are strictly more efficient than the algorithms in the safer class.

4.1 Preliminaries

We begin with two concepts that are useful across all classes. The first one, *eligible*, describes a set of nodes that can make each other stable by collectively changing to B.

Definition 13. *A set of nodes W is* eligible *if,*

1. W is a non-empty set including only changeable nodes in state A;

2. when upgrading each node in W to state B, each node in W is stable.

Secondly, we derive an important structural characterization of algorithms. We say an algorithm is INCREMENTAL if it is more efficient than any of its own subsets.

Definition 14. *S_β is INCREMENTAL if $S_\beta \succeq S_\alpha$ for any $S_\alpha \subseteq S_\beta$.*

Intuitively, we would assume that every algorithm is INCREMENTAL. Surprisingly, it turns out that even rational algorithms may not be INCREMENTAL, i.e., an algorithm with fewer changing views that are a proper subset of another algorithm can be more efficient. As a simple example, consider a line network consisting of 5 nodes, with $k = 5$ and $q = 1$. Consider two algorithms S_α and S_β. Both algorithms include all views in which all neighbors are in state B (which is rational for $q = 1$), and in addition they contain a set of additional changing views $\Gamma^1(x), \ldots, \Gamma^4(x)$ as shown in Figure 1. Let $S_\alpha = S_r \cup \{\Gamma^1(x), \Gamma^2(x), \Gamma^3(x)\}$ and $S_\beta = S_r \cup \{\Gamma^1(x), \Gamma^2(x), \Gamma^3(x), \Gamma^4(x)\}$, i.e., the two algorithms are identical except that S_β includes one more changing view $\Gamma^4(x)$. However, in spite of this extra changing view, S_β is actually strictly worse than S_α on the 5-node chain. Assume that initially, only the left-most node is in state B; the others are regular nodes in state A. It can be verified that with S_α, all nodes will upgrade to B in the equilibrium; while with S_β, Nodes 2 and 5 upgrade to B in the first round, but Nodes 3 and 4 are then stuck – they have no chance to upgrade to B. Thus, even though S_β is rational and a strict superset of S_α, S_α is more efficient. Thus, the rational algorithm S_β is not INCREMENTAL.

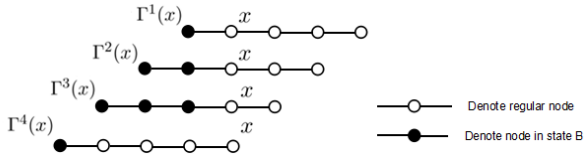

Fig. 1. Views $\Gamma^1(x), \Gamma^2(x), \Gamma^3(x), \Gamma^4(x)$.

The reason S_β is not INCREMENTAL is that it violates a natural property we call *Augmentation-Completeness*. For a view $\Gamma(x)$, we say a view $\Gamma'(x)$ *augments* $\Gamma(x)$ if $\Gamma'(x)$ is identical to $\Gamma(x)$ except that i) some stubborn nodes in $\Gamma(x)$ are changeable nodes in $\Gamma'(x)$, and ii) some regular nodes in $\Gamma(x)$ are in state B in $\Gamma'(x)$. Clearly, it should always be easier for a node to upgrade to B in a view $\Gamma'(x)$ that augments $\Gamma(x)$ than in $\Gamma(x)$. The augments relationship thus implies a partial order of views in terms of the extent to which nodes are willing to change. Denote by $F^+(.)$ a mapping from an algorithm S to the set of all views that augment at least one view in S. I.e., for a set S and any view $\Gamma(x) \in S$, $F^+(S)$ includes all views that augment $\Gamma(x)$. Using this definition, we can show that for two algorithms S_α and S_β such that S_β is a superset of $F^+(S_\alpha)$, it holds for any G and any initial state, the set of regular nodes changing to state B in P_α is a subset of that in P_β.

We say an algorithm S is Augmentation-Complete, AUG-COMPLETE, if for any of its views $\Gamma(x) \in S$, S contains all the views that augment $\Gamma(x)$. Formally, S is AUG-COMPLETE if $S = F^+(S)$. Augmentation-Completeness is a powerful tool that enables us to compare the efficiency among different algorithms, and we will use it extensively when constructing the optimal protocol-safe algorithm. With this definition, we can prove the following theorem.

Theorem 2. *Any AUG-COMPLETE algorithm is INCREMENTAL.*

4.2 Byzantine-safe Algorithm

It is trivial to see that the only Byzantine-safe algorithm is the simple 1-hop algorithm; nodes upgrade to B is a sufficient number of neighbors have upgraded to B. It is the only algorithm that ensure stability even in the presence of irrational neighbors. Since $\mathcal{C}_{Byz}$ contains only a single algorithm, the class is clearly subset-closed and union-closed.

Theorem 3. $\mathcal{C}_{Byz} = \{S_{1-hop}\}$.

Algorithm 1: S_{1-hop}

$S_{1-hop} := \{\Gamma(x) || N^B(x)|/|N(x)| \geq q\}$.

4.3 Rational-safe Algorithms

In this subsection, we devise an optimal rational-safe algorithm S_{fore}, called *foresee*. S_{fore} works as follows: In order to check whether it should upgrade given its current view, a node x temporarily "assumes" its state to be B, and under this assumption repeatedly finds eligible nodes in its view and "upgrades" these node to B. If the outcome of this local simulation is such that at least a q-fraction of x's neighbors are in state B, x upgrades to B. The key of S_{fore} is that x assumes itself to be in state B at the outset.

Algorithm 2: S_{fore}

Given a view $\Gamma(x)$. Set the state of x to B;
while *There exists a node* $y \in V_{k-1}(x)$ *such that* $\{y\}$ *is eligible* **do**
  Set the state of y to B.
if $|N^B(x)|/|N(x)| \geq q$ **then**
  $\Gamma(x)$ is a changing view. (i.e., add $\Gamma(x)$ into S_{fore})

Algorithm S_{fore} can be much more efficient than the 1-hop algorithm S_{1-hop}. Consider a graph G with only regular nodes and threshold q $q < 1/d_{max}$, where d_{max} is the maximum degree of graph G. If each node uses S_{1-hop}, no node changes to B. In contrast, if every regular node uses S_{fore}, every node will assume itself to be in state B and execute the while loop. For each of its neighbors y, as $q < \frac{1}{d_{max}}$, it holds that $N^B(y)/N(y) = 1/N(y) > 1/d_{max} > q$, i.e., y is eligible and can upgrade to B. That is, x knows that each of its neighbors will upgrade to B and it can safely upgrade itself. Therefore, with S_{fore}, all nodes simultaneously upgrade to B in the very first round.

Studying S_{fore}, we find that each subset of S_{fore} is a rational-safe algorithm and each rational-safe algorithm is a subset of S_{fore}.

Lemma 1. $S \subseteq S_{fore} \iff S \in \mathcal{C}_{Rat}$.

Thus, S_{fore} is the rational-safe algorithm with the maximum set of views and it is the superset of each algorithm in $\mathcal{C}_{Rat}$. Therefore, we can infer that $\mathcal{C}_{Rat}$ is subset-closed and union-closed by definition. Furthermore, as S_{fore} is AUG-COMPLETE (we can check any view in S_{fore} according to the definition of AUG-COMPLETE), from Theorem 2, we can infer that S_{fore} is more efficient than any of its subset. Therefore, S_{fore} is optimal within the class of rational-safe algorithms.

Theorem 4. *The class of rational-safe algorithms* $\mathcal{C}_{Rat}$ *is subset-closed and union-closed. Furthermore,* $S_{fore} = OPT(\mathcal{C}_{Rat})$.

4.4 Protocol-safe Algorithms

In many cases, S_{fore} is still very inefficient. Protocol-safe algorithms can be more aggressive and efficient, because they can consider *eligible sets* rather than only eligible individual nodes. We begin with a simple algorithm S_{trust}, which is intuitive but not optimal. Let $D(W)$ be the diameter of a node set W in G. We can easily infer that S_{trust} is protocol-safe because it only considers eligible sets W with diameter less than $k-1$. The small diameter ensures that each node in W can see W as an eligible set in its own view. Thus, each node in W can simultaneously upgrade to B together.

Algorithm 3: S_{trust}–Finding eligible sets in view $\Gamma(x)$ with restricted diameter

$S_{trust} := \{\Gamma(x) | \exists W \subseteq V_{k-1}(x)$ such that W is eligible, $D(W) \leq k-1$ and $x \in W\}$.

To see that S_{trust} can be much more efficient than any rational-safe algorithm including S_{fore}, consider a network with diameter less than k and threshold $q > 1/d_{min}$ (d_{min} is the minimum degree in G), e.g., a complete graph G and $q = 1$. It is easy to see that if each node executes S_{fore}, no node changes to B. On the other hand, if each node executes S_{trust}, every node will find the entire graph as an eligible set and thus all nodes simultaneously upgrade to B in round 1.

However, S_{trust} is not optimal. To see why S_{trust} is not optimal, we show that another protocol-safe algorithm S^+_{trust} which is the union of S_{trust} plus a special changing view $\Gamma^*(x)$ is more efficient than S_{trust}. Consider a 5-node chain of regular nodes, and assume $k = 3$ and $q = 0.1$. In this example, S_{trust} cannot find any eligible set with diameter at most than 3, thus each regular node is stuck in A. On the other hand, if the extra changing view $\Gamma^*(x)$ is the 5-node chain of regular nodes, then the middle node can upgrade to B in Round 1; and all other nodes will also upgrade in subsequent rounds. We now derive two optimal protocol-safe algorithms - one non-constructive and one constructive with an additional assumption.

Non-constructive Optimal Protocol-safe Algorithm: We give an optimal protocol-safe algorithm S^*. To describe S^*, we introduce a class of algorithms $\mathcal{C}^+_{Pro} = \{S | S \in \mathcal{C}_{Pro}$ and S is AUG-COMPLETE$\}$ with all AUG-COMPLETE algorithms in $\mathcal{C}_{Pro}$. Using the definition of $\mathcal{C}^+_{Pro}$, we define $S^* := \bigcup_{S \in \mathcal{C}^+_{Pro}} S$. We show that S^* is optimal within the class of protocol-safe algorithms.

Theorem 5. $S^* = OPT(\mathcal{C}_{Pro})$.

The proof is mainly based on three structural lemmas describing protocol-safe algorithms. The first one states that $\mathcal{C}^+_{Pro}$ is union-closed (Intriguingly, we later show that $\mathcal{C}_{Pro}$ itself is not union-closed).

Lemma 2. *$\mathcal{C}^+_{Pro}$ is union closed.*

Since S^* is defined as a union of all algorithms in $\mathcal{C}^+_{Pro}$, from Lemma 2, we can infer that $S^* \in \mathcal{C}^+_{Pro}$. i.e., S^* is protocol-safe and AUG-COMPLETE, and it is trivial to see that S^* is the superset of any algorithm in $\mathcal{C}^+_{Pro}$. Since we want to show S^* is optimal in $\mathcal{C}_{Pro}$, we need to build a connection between $\mathcal{C}_{Pro}$ and $\mathcal{C}^+_{Pro}$. In the following lemma, we find that for any protocol-safe algorithm S, $F^+(S) \in \mathcal{C}^+_{Pro}$.

Lemma 3. *For any $S \in \mathcal{C}_{Pro}$, $F^+(S) \in \mathcal{C}^+_{Pro}$.*

As S^* is the superset of any algorithm in $\mathcal{C}^+_{Pro}$, from Lemma 3, we can get that for any protocol-safe algorithm S, $F^+(S)$ is a subset of S^*. For any algorithm S, it holds $S \subseteq F^+(S)$ (from the definition of $F^+(\cdot)$). We conclude in the next lemma that any protocol-safe algorithm S is a subset of S^*. (Interestingly, the reverse does not hold, i.e., there exists $S \subseteq S^*$ such that $S \notin \mathcal{C}_{Pro}$.)

Lemma 4. *If $S \in \mathcal{C}_{Pro}$, then $S \subseteq S^*$.*

On the other hand, we know that S^* is AUG-COMPLETE and every AUG-COMPLETE algorithm is INCREMENTAL (Theorem 2). That is, S^* is more efficient than any subset of itself. We can infer that S^* is optimal within the class of protocol-safe algorithms.

Finally, we show that the class of protocol-safe algorithms is not union-closed and subset-closed. It is different from the other three classes of the hierarchy.

Theorem 6. *The class of protocol-safe algorithms $\mathcal{C}_{Pro}$ is not union-closed and subset-closed.*

Constructive Optimal Protocol-safe Algorithm: Algorithm S^* is optimal, but it is non-constructive and it is entirely unclear how to apply this algorithm in a real network setting. In this section, we give a constructive optimal algorithm S_Δ. The algorithm is based on techniques similar to dynamic programming: we inductively construct a maximal AUG-COMPLETE set of changing views by enumeration in a systematic manner that additionally satisfy a so-called UNIFORM constraint. To do so, a node checks all possible views, according to the total number of nodes and the total number of regular nodes in each view in an increasing order, and adds the valid ones into S_Δ's changing view set. Ultimately, we can prove that S_Δ is optimal, but only under the assumption that S^* satisfies the UNIFORM property. We conjecture that this is true, but we do not currently have a formal proof. Therefore, we only claim the weaker theorem that S_Δ is optimal among all UNIFORM AUG-COMPLETE protocol-safe algorithms.

4.5 Possible-safe Algorithms

For the class of possible-safe algorithms, we show that S_{hope} is optimal. The algorithm includes the finding of an eligible set W.

Algorithm 4: S_{hope}

$S_{hope} := \{\Gamma(x) | \exists W \subseteq V_{k-1}(x)$ such that W is eligible and $x \in W\}$.

Again, we show that S_{hope} can be much more efficient than an optimal protocol-safe algorithm S^*. Specifically, consider a network of 5 nodes, four of which form a square, and one node is in the middle linking to the other 4 nodes. Suppose $k = 2$ and $q = 1$. With S^*, no node will upgrade. With S_{hope}, the center node will can see all four neighbors and it knows that the set of all neighbors plus itself is eligible. Thus, the center node upgrades to B, which is more efficient albeit unstable, because the corner nodes do not know the topology of the opposite corner node and will remain in A.

An defining structural property of the class of possible-safe algorithms (and S_{hope}) is that each subset of S_{hope} is a possible-safe algorithm and each possible-safe algorithm is a subset of S_{hope}.

Lemma 5. $S \subseteq S_{hope} \iff S \in \mathcal{C}_{Pos}$.

Thus, S_{hope} plays the same central role for $\mathcal{C}_{Pos}$ as S_{fore} played for $\mathcal{C}_{Rat}$. Indeed, the rest of the argument follows along the same lines. According to Lemma 5, we know that S_{hope} is the possible-safe algorithm with the maximum set of views and it is the superset of each algorithm in $\mathcal{C}_{Pos}$. Therefore, we can infer that $\mathcal{C}_{Pos}$ is subset-closed and union-closed by definition. Furthermore, as S_{hope} is AUG-COMPLETE, from Theorem 2, we can infer that S_{hope} is more efficient than any of its subset. Therefore, we can conclude that S_{hope} is optimal within the class of possible-safe algorithms.

Theorem 7. *The class of possible-safe algorithms $\mathcal{C}_{Pos}$ is subset-closed and union-closed. Furthermore, $S_{hope} = OPT(\mathcal{C}_{Pos})$.*

4.6 Putting Everything Together

Combining all the above results, we now show a strict order in terms of conversion ratio among all the optimal algorithms in the four classes of local decision algorithms. The optimal algorithm of a safer class is strictly less efficient than the optimal algorithm in the less safe class. That is, *the achievable safety guarantee of these algorithms precisely corresponds to their performance efficiency in terms of conversion ratio*: A beautiful finding. We construct a family of graphs in which the respectively safer optimal algorithm will have fewer nodes upgrade to B, than the respective less safe optimal algorithm.

Theorem 8. *It holds $OPT(\mathcal{C}_{Pos}) \succ OPT(\mathcal{C}_{Pro}) \succ OPT(\mathcal{C}_{Rat}) \succ OPT(\mathcal{C}_{Byz})$, for any $k \geq 2, q > 0$.*

Proof. We first show that the order of efficiency holds. As $\mathcal{C}_{Byz} \subseteq \mathcal{C}_{Rat} \subseteq \mathcal{C}_{Pro} \subseteq \mathcal{C}_{Pos}$ (Theorem 1), we can infer that $OPT(\mathcal{C}_{Pos}) \succeq OPT(\mathcal{C}_{Pro}) \succeq OPT(\mathcal{C}_{Rat}) \succeq OPT(\mathcal{C}_{Byz})$. We know that $S_{1-hop} = OPT(\mathcal{C}_{Byz})$, $S_{fore} = OPT(\mathcal{C}_{Rat})$, $S^* = OPT(\mathcal{C}_{Pro})$ and $S_{hope} = OPT(\mathcal{C}_{Pos})$. Thus, it holds $S_{hope} \succeq S^* \succeq S_{fore} \succeq S_{1-hop}$.

Next, we need to show that the strict order of efficiency holds for any $k \geq 2, q > 0$. In the following context, consider any $k \geq 2, q > 0$.

To show that $OPT(\mathcal{C}_{Rat}) \succ OPT(\mathcal{C}_{Byz})$, we show $S_{fore} \succ S_{1-hop}$. Consider a graph G with 2 regular nodes connected. If each regular node executes S_{1-hop}, neither changes to state B. If each regular node executes S_{fore}, both change to state B. Therefore, $S_{fore} \succ S_{1-hop}$.

To show that $OPT(\mathcal{C}_{Pro}) \succ OPT(\mathcal{C}_{Rat})$, as $S^* \succ S_{trust}$, we only need to show that $S_{trust} \succ S_{fore}$. Consider a complete graph G with n regular nodes such that $(n-1) > 1/q$. If each regular node executes S_{fore}, no regular node changes to state B. If each regular node executes S_{trust}, each regular node changes to state B. Therefore, $S_{trust} \succ S_{fore}$.

To show that $OPT(\mathcal{C}_{Pos}) \succ OPT(\mathcal{C}_{Pro})$, we show that $S_{hope} \succ S^*$. As we do not know the explicit form of S^*, our idea is to construct a graph in which there is one regular node x that can see all the nodes in the graph and only when all nodes change to state B, all regular nodes are stable in the equilibrium; then only x changes to state B by running S_{hope} and other nodes will keep state A since they cannot find an eligible set in their view. Moreover, if each regular node executes S^*, since every regular node should be stable in the equilibrium, we can infer that no regular node changes to B.

We build the construction step by step. Recall that we have shown in Section 4.5 an example in which S_{hope} is more efficient than S^* for $k = 2, q = 1$. Using a similar technique, we extend the case $k = 2, q = 1$ to the general case. In the following example, we show that for any $k \geq 2$ and $q = 1$, it holds that $S_{hope} \succ S^*$.

Suppose $q = 1$ and any $k \geq 2$. Construct graph $G = (V, E)$ as follows: Construct $2k + 2$ chains, where the ith chain ($i = 1, 2, ..., 2k + 2$) includes regular nodes $x_{1i}, x_{2i}, ..., x_{(k-1)i}$. In the ith chain, x_{ji} links to $x_{(j+1)i}$ ($j = 1, 2, ..., k - 2$). There is a regular node x_{00} linking to $x_{11}, x_{12}, ..., x_{1(2k+2)}$. For each i ($i = 1, 2, ..., 2k + 1$), $x_{(k-1)i}$ links to $x_{(k-1)(i+1)}$ and $x_{(k-1)(2k+2)}$ links to $x_{(k-1)1}$. See Figure 2. We call such G a "cage".

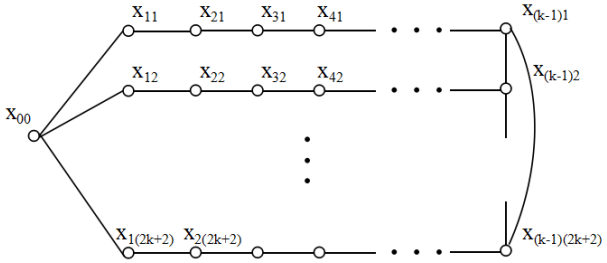

Fig. 2. The "Cage" Graph

In process P_1, suppose each regular node in G executes S^*. As S^* is protocol-safe, we can infer that each node is stable in the equilibrium. As $q = 1$, we can infer all nodes in the equilibrium in P_1 are in the same state, namely either all nodes are in state A or all nodes are in state B. Otherwise, a node in state B that has any neighbor in state A is not stable. We then show by contradiction that in P_1, all nodes are in state A. Suppose all nodes are in state B. By symmetry, $x_{(k-1)1}, S_{(k-1)2}, ..., S_{(k-1)(2k+2)}$ should change to B in the same round T^*. Thus we know that in $T^* - 1$, nodes $x_{(k-1)1}, S_{(k-1)2}, ..., S_{(k-1)(2k+2)}$ are all in state A. Denote by $\Gamma^*(x_{(k-1)1})$ the view of $x_{(k-1)1}$ in $T^* - 1$. We can infer that $\Gamma^*(x_{(k-1)1}) \in S^*$. Consider another graph G' that has the same topology and type as G and the initial state of G' is the same as G in $T^* - 1$ except that $x_{(k-1)(k+2)}$ is a stubborn node. We still consider that each node in G' executes S^*. As $x_{(k-1)1}$ cannot see the state or type of $x_{(k-1)(k+2)}$ (due to k-hop information restriction), we can infer that the initial view of $x_{(k-1)1}$ in G' is the same as $\Gamma^*(x_{(k-1)1})$ which means $x_{(k-1)1}$ changes to state B in round 1 in G'. Then in G', we get that $x_{(k-1)1}$ is in state B and $x_{(k-1)(k+2)}$ is a stubborn node. It is easy to see that at least one regular node in $\{x_{(k-1)1}, x_{(k-1)2}, ..., x_{(k-1)(k+1)}, x_{(k-1)(k+3)}, ..., x_{(k-1)(2k+2)}\}$ is not stable. This contradicts our assumption that S^* is protocol-safe. Therefore, we know that in P_1 in which each regular node executes S^*, each regular node is in A.

In process P_2, suppose each regular node in G executes S_{hope}. Denote by $S_0 = \{\text{each view}\}$. As x_{00} knows the whole graph G, the entire set of nodes V is an eligible set. Therefore, according to the definition of S_{hope}, x_{00} changes to state B in round 1. We can see that at least one node in P_2 changes to state B. Thus, S_{hope} is more efficient than S^*.

Using the "cage" graph, we can extend the specific threshold q to the general case. Suppose $q > 0$ and each node has $(k+1)$-hop information $k \geq 1$. We construct a "cage"

graph G'' like above and additionally link c_{ij} stubborn nodes to x_{ij} (c_{ij} will be assigned in the following context). We can get the similar proof of the general case in G'' to that above in G with $q = 1$ by achieving the following two rules: 1) only x_{00} can see the whole graph (This can be done since x_{00} in G' has $(k+1)$-hop information with which it can see all the stubborn nodes and each regular node can not see the whole graph.) and 2) all regular nodes are stable in the equilibrium if and only if all of them are in state A or state B. In order to achieve the second rule, for a regular node x_{ij} with b_{ij} regular neighboring nodes, the following two equalities should hold, i) $b_{ij}/(b_{ij} + c_{ij}) \geq q$ and ii) $(b_{ij} - 1)/(b_{ij} + c_{ij}) < q$. Rearranging these equations, we can derive

$$b_{ij}\frac{1-q}{q} - \frac{1}{q} < c_{ij} \leq b_{ij}\frac{1-q}{q}.$$

Since $1/q > 1$, we know that there must be an integer in the range $(b_{ij}\frac{1-q}{q} - \frac{1}{q}, b_{ij}\frac{1-q}{q})$, and hence, S_{hope} is more efficient than S^* for any $k \geq 2, q > 0$.

Thus, we have shown $OPT(\mathcal{C}_{Pos}) \succ OPT(\mathcal{C}_{Pro}) \succ OPT(\mathcal{C}_{Rat}) \succ OPT(\mathcal{C}_{Byz})$.

Also, note again that $\mathcal{C}_{Pro}$ differs from $\mathcal{C}_{Rat}$ with regard to the union-closed and subset-closed property (Thm 6 vs Thm 4). The reason for this difference is that in a rational-safe algorithm, each node assumes every other node being rational, where such assumption is static since all the rational algorithms are known in advance. But in protocol-safe, it is no longer true. In contrast, for a protocol-safe algorithm, it cannot do the same because a node needs to recursively consider what its neighboring algorithm might do. The class of protocol-safe algorithm is in this sense "dynamic".

Local vs Global Decision Algorithms All algorithms in this paper are *local* decision algorithms based on k-hop of multi-hop information. This means that all of these algorithms are non-optimal compared to a global optimal decision algorithm that has complete information of the network. Thus, even the most efficient of our local decision algorithms, S_{hope} is not globally optimal. Indeed, a globally optimal algorithm can be regarded as S_{hope} with k being infinitely large. To see that S_{hope} with k-hop information can be suboptimal, consider a ring network G with $2k + 2$ regular nodes, and $q > 0.5$. With global view, all nodes should upgrade to B, rendering all nodes stable. However, if each node uses algorithm S_{hope} with local view, no node upgrades to B.

5 Conclusion

In this paper, we have derived a hierarchy of local decision algorithms in a basic influence network setting with multi-hop information. Giving nodes multi-hop information renders the problem more complex since nodes now need to reason about other nodes' behaviors and views. We have shown that the classes of algorithms that achieve different safety properties are strictly separated from each other in terms of efficiency, thus capturing the underlying trade-off between safety-guarantee and ability to "take action". The hierarchy thus disentangles and categorizes the questions raised by the typical recursive distributed problems such as, "I will take action, if my neighbor takes action; and to determine whether he will take action, I need to know whether my neighbor thinks I take action, etc." It is intriguing that such complicated recursive multi-hop patterns give raise to a natural hierarchy of classes of local decision algorithms.

References

1. W. B. Arthur and D. A. Lane. Information contagion. *Structural Change and Economic Dynamics*, 4(1):81–104, 1993.
2. W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1029–1038. ACM, 2010.
3. S. Frischknecht, B. Keller, and R. Wattenhofer. Convergence in (social) influence networks. In *Distributed Computing*, pages 433–446. Springer, 2013.
4. M. Gardner. Mathematical games: The fantastic combinations of john conways new solitaire game life. *Scientific American*, 223(4):120–123, 1970.
5. E. Goles and J. Olivos. Periodic behaviour of generalized threshold functions. *Discrete Mathematics*, 30(2):187–189, 1980.
6. E. Goles and M. Tchuente. Iterative behaviour of generalized majority functions. *Mathematical Social Sciences*, 4(3):197–204, 1983.
7. M. Granovetter. Threshold models of collective behavior. *American journal of sociology*, 83(6):1420, 1978.
8. D. Kempe, J. Kleinberg, and É. 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*, pages 137–146. ACM, 2003.
9. D. Kempe, J. Kleinberg, and É. Tardos. Influential nodes in a diffusion model for social networks. In *IN ICALP*, 2005.
10. J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 420–429. ACM, 2007.
11. M. W. Macy. Chains of cooperation: Threshold effects in collective action. *American Sociological Review*, pages 730–747, 1991.
12. M. W. Macy and R. Willer. From factors to actors: Computational sociology and agent-based modeling. *Annual review of sociology*, pages 143–166, 2002.
13. S. Morris. Contagion. *The Review of Economic Studies*, 67(1):57–78, 2000.
14. D. Peleg. Local majorities, coalitions and monopolies in graphs: a review. *Theoretical Computer Science*, 282(2):231–257, 2002.
15. S. Poljak and M. Sra. On periodical behaviour in societies with symmetric influences. *Combinatorica*, 3(1):119–121, 1983.
16. M. Rogers Everett. Diffusion of innovations. *New York*, 1995.
17. T. C. Schelling. Hockey helmets, concealed weapons, and daylight saving: A study of binary choices with externalities. *Journal of Conflict Resolution*, pages 381–428, 1973.
18. T. C. Schelling. *Micromotives and macrobehavior*. WW Norton & Company, 2006.
19. T. W. Valente. Network models of the diffusion of innovations. *Computational & Mathematical Organization Theory*, 2(2):163–164, 1996.
20. S. Wasserman. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
21. P. Winkler. Puzzled delightful graph theory. *Communications of the ACM*, 51(8):104–104, 2008.
22. H. P. Young. *Individual strategy and social structure: An evolutionary theory of institutions*. Princeton University Press, 2001.