On the Convergence Time in Graphical Games: A Locality-Sensitive Approach

Juho Hirvonen
Aalto University, Finland
Helsinki Institute for Information Technology (HIIT), Espoo, Finland

Laura Schmid
Kim Jaechul Graduate School of AI, KAIST, Seoul, Republic of Korea

Krishnendu Chatterjee
IST Austria, Klosterneuburg, Austria

Stefan Schmid
TU Berlin, Germany
Weizenbaum Institute, Berlin, Germany

Abstract

Graphical games are a useful framework for modeling the interactions of (selfish) agents who are connected via an underlying topology and whose behaviors influence each other. They have wide applications ranging from computer science to economics and biology. Yet, even though an agent's payoff only depends on the actions of their direct neighbors in graphical games, computing the Nash equilibria and making statements about the convergence time of "natural" local dynamics in particular can be highly challenging. In this work, we present a novel approach for classifying complexity of Nash equilibria in graphical games by establishing a connection to local graph algorithms, a subfield of distributed computing. In particular, we make the observation that the equilibria of graphical games are equivalent to locally verifiable labelings (LVL) in graphs; vertex labelings which are verifiable with constant-round local algorithms. This connection allows us to derive novel lower bounds on the convergence time to equilibrium of best-response dynamics in graphical games. Since we establish that distributed convergence can sometimes be provably slow, we also introduce and give bounds on an intuitive notion of "time-constrained" inefficiency of best responses. We exemplify how our results can be used in the implementation of mechanisms that ensure convergence of best responses to a Nash equilibrium. Our results thus also give insight into the convergence of strategy-proof algorithms for graphical games, which is still not well understood.

2012 ACM Subject Classification Theory of computation → Network games; Theory of computation → Algorithmic game theory

Keywords and phrases distributed computing, Nash equilibria, mechanism design, best-response dynamics

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2023.11

Related Version *Full Version*: `https://arxiv.org/abs/2102.13457`

Funding This work was partially funded by the Academy of Finland, grant 314888, the European Research Council CoG 863818 (ForM-SMArt), and the Austrian Science Fund (FWF) project I 4800-N (ADVISE). LS was supported by the Stochastic Analysis and Application Research Center (SAARC) under National Research Foundation of Korea grant NRF-2019R1A5A1028324.

Acknowledgements We thank the anonymous reviewers for their feedback.

1 Introduction

Modeling the interactions of multiple selfish agents, whose decisions and behavior influence each other and are in some way dependent on an underlying topology, is an important aspect of solving problems from a wide range of diverse fields. Game theory offers a natural approach to this issue in the form of multiplayer network games [28, 9]. Instances of such

network games are ubiquitous [3, 40, 28, 2, 43, 1]. They all have in common that interactions and players' decisions are in some way governed by the underlying graph and that players' payoffs usually depend on the network that links them. Considering their heterogeneity and complexity, general network games, such as the rich class of congestion games [40], can be difficult to analyze. A considerable amount of research has thus been dedicated to more restricted versions of such games, such as those where each player's utility is solely determined by their own and their direct neighbors' actions. These so-called *graphical games* (see [31] for a comprehensive introduction and survey) capture the *locality* of effects on players, and are useful for settings where there are only a few strong direct influences on every agent [9].

Computing the Nash equilibria of graphical games or proving results about their properties or convergence is challenging in general. Previous work has dealt with aspects such as deciding whether a given strategy profile is a Nash equilibrium or whether equilibria exist, and classifying the complexity of these tasks [41]. Algorithms for computing (approximate) equilibria, often for various specific types of graphs, have also been a focus of a considerable number of studies [37, 20, 21, 44, 27, 22]; however, it is important to note that these algorithms usually do not reflect natural game dynamics. Hence, they are not guaranteed to be efficient or strategy-proof: players can have incentives to deviate from the algorithm. Additionally, we also still lack a systematic understanding and classification in terms of computational complexity. In particular, the convergence time of strategy-proof algorithms and local dynamics, such as best-response dynamics, to Nash equilibria is still not fully understood, except for special cases [32, 26, 42].

In this work, we approach graphical games from a new perspective, demonstrating that the theory of distributed computing is particularly well-suited as a tool to study the behavior and computational power of decentralized game-theoretic systems with multiple players. In particular, distributed computing offers the tools to prove unconditional and general impossibility results, which demonstrate natural limits to the behavior of systems. One of the most significant limiting factors in distributed computing is *locality* [39], or how far information must propagate in a system to solve a particular task. Advantageously, impossibility results based on locality then apply independently of more specific factors, such as the model of strategic behavior considered. This makes it possible to analyse games without a fixed model of play.

Distributed computing further provides a benchmark for optimal cooperation. By comparing the outcome of models such as best-response dynamics to the best efficient distributed algorithm, we can assess the inefficiency of systems that are not perfectly coordinated. We also get natural algorithmic tools for mechanism design: properly incentivised, distributed algorithms can potentially be run by the agents without the need for centralised control. We also note that the tools of distributed computing theory help us understand games with multiple equilibria. We can consider questions such as which subset of equilibria can be computed efficiently, and what their properties are. Additionally, we can study how likely *undesirable* behavior is, noting that bad equilibria are also often hard to compute.

Investigating graphical games as distributed systems may seem very natural, but has only been rarely explored [19]. In order to showcase the rich set of tools resulting from this perspective, we show how Nash equilibria and other stable state concepts in graphical games correspond to a family of computational problems in distributed computing called *locally verifiable labellings (LVLs)* [36]. The computational complexity of these problems is well studied (see e.g. [36, 16, 11, 17, 5]). Using the properties of LVLs, we can then classify graphical games based on how hard it is to compute a Nash equilibrium, i.e. based on the distributed hardness of computing a stable state. We show how this implies lower bounds

on the convergence time of best-response dynamics in network games: these dynamics can converge only as fast as a distributed algorithm can compute solutions to the equivalent graph problem. We find three types of graphical games in the process: (1) Games where Nash equilibria are easy to compute and best responses also converge fast; (2) games where computing a Nash equilibrium is hard, and thus best responses cannot converge fast; and (3) games where Nash equilibria are easy to compute, but best responses may fail to converge fast. We exemplify our approach with three corresponding fundamental graphical games that have been studied in the literature [8, 9, 30].

Games of the third type are particularly interesting, as the existence of fast distributed algorithms offers possibilities for mechanism design. We show a way of constructing games where best responses simulate distributed algorithms, thus ensuring convergence in games of type 3. Since best-response dynamics are rational with respect to the restricted local knowledge of agents, our work also connects to the open question of strategy-proof algorithms for distributed Nash equilibria computation that is posited by [29].

Finally, we define a notion called inefficiency of best responses: It is a time-constrained analogue of the Price of Anarchy that compares the evolution of the system state over a time period with the best solution that can be computed with the same time complexity. We demonstrate that Price of Anarchy can be an unfair measure by comparing strategic behavior with perfect *distributed* cooperation.

Overall, our approach of classifying games based on the computational hardness of their equilibria provides a powerful new perspective on graphical games that allows for a deeper understanding of strategic behavior, and contributes to a strengthened connection between game theory and distributed computing.

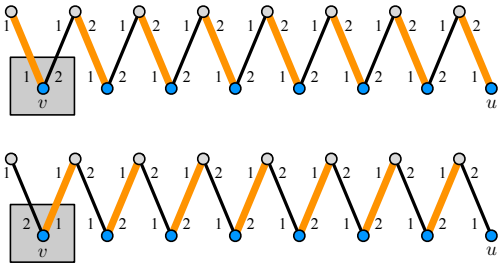

Figure 1 Two simple networks with preferences indicated by numbers: each agent prefers the neighbour labelled 1 over the neighbour labelled 2. The two networks only differ in the preferences of the agent labelled v. Still, the unique stable matchings (marked with bold edges) are disjoint.

Stable matching [24] is a classic example of a graphical game: we have a bipartite network where each agent on the left is connected to some subset of agents on the right. Agents have some preference relation over the other agents they are connected to, and the goal is to find a matching such that no pair of connected agents prefer each other over their current matches. While there is a polynomial-time truthful *centralised* mechanism for finding a stable matching, the problem becomes hard when agents form a decentralized distributed system [23]: in the worst case, the output of an agent depends on the preferences of other agents at linear distance (see Figure 1). This example highlights the goal of our work: we study games that are considered "easy" is the literature of game theory and provide a more fine-grained view of their hardness from the perspective of decentralised systems.

1.1 Organization of the paper

We begin with a primer on graphical games and locally verifiable labelings in Section 2. We proceed to show in Section 3 how Nash equilibria of graphical games correspond to a family of computational tasks called locally verifiable labellings. Then, we show how to apply these results to best-response game dynamics. In Section 4 we study and classify three fundamental graphical games. In Section 5 we study how distributed algorithms can be used to incentivize desired agent behavior. In Section 6, we use theory of distributed computing to study how inefficient best-response dynamics are compared to distributed algorithms. We conclude with an outlook on the connection between theory of distributed computing and game theory. To improve readability and provide intuition, we, as indicated, defer technical details and full proofs to the Appendix.

2 Model

2.1 Graphical games

Define a multiplayer game in its general form as consisting of n players $i \in \mathcal{I}$, each equipped with a pure strategy space or action space A_i. The cartesian product $A_1 \times A_2 \times ... \times A_n$ of the action spaces of individual agents is denoted by $\mathcal{A}$. Every player has a utility function $u_i(a)$ mapping each strategy profile to a value: $u_i \colon \mathcal{A} \times \mathcal{I} \to \mathbb{R}$. We use a_{-i} to denote the joint strategy profile of all players except for player i.

Throughout the paper, our examples will focus on pure strategies, but all of the general results also apply to mixed strategies. A strategy profile a^* is a (pure) Nash equilibrium if for all players i it holds that $u_i(a_i^*, a_{-i}^*) \geq u_i(a_i, a_{-i}^*) \ \ \forall a_i \in A_i$. In other words, no player in a Nash equilibrium can gain a higher utility by unilaterally deviating. We say that the strategy a_i^* is a *best response* to the rest of the strategy profile a_{-i}^*.

We can now define *graphical games*, given by a triple $(\mathcal{A}, u, N)$. As introduced in [31], graphical games are a concisely representable form of *multiplayer games* on networks. A *network* or a *graph* $N = (V, E)$ consists of a set of n *nodes* V and a set E of *edges* between pairs of nodes. The nodes of the network N then represent the agents, or players, in the graphical game. We define the local neighborhood of a node v as $B(v) \subseteq \{1, .., n\} = \{j \in V, (v, j) \in E\}$, with $v \in B(v)$ as well. A player's *utility function* $u_v(a)$ now depends only on a strategy profile restricted to their local neighborhood $B(v)$, i.e. the partial strategy profile $a^{(v)} = (a_i : i \in B(v))$. We denote the product of the individual agents' utility functions by u.

A classic example of graphical games are coloring games [30]. In a k-coloring game, the agents must choose resources (colors) that differ from the resources chosen by their neighbors. The action set $A_i = \{1, 2, \ldots, k\}$ for all agents. Utility is 1 if the strategy a_i is a color that is not chosen by its neighbors, and 0 otherwise. A strategy profile is a Nash equilibrium if none of the agents with a similarly colored neighbor can choose a free color.

In this work, we consider the question of *convergence* to Nash equilibria via *best-response dynamics*, a specific example of *local dynamics*. We assume that players can update their strategy in between gameplay. In one step of the dynamics, one player is selected and this player then updates her strategy with the best response to her neighborhood's strategy profile. That is, the action of a node v is a best response to the partial strategy profile $a_v(t-1)$. There is some fixed order on the actions that is used to break ties. We assume that nodes only have local information and cannot look beyond their neighborhood. Their restricted knowledge makes such local dynamics rational. In this work, we will subsequently only consider best responses. However, our results also hold for more general local dynamics.

In order to have a reasonable definition of a running time for local dynamics, we define a model of *fair best responses*. The play consists of *fair rounds*. During each round an adversary schedules all agents to act exactly once and one at a time. The convergence time of best-response dynamics on a fixed network is the maximum number of fair rounds until all players have reached a Nash equilibrium over, all possible orders of play. For random initial strategy profiles we say that best-response dynamics converge if they converge with high probability over the initial strategy assignment.

We choose the liveness criterion of each player playing each round for two reasons. First, if there is no bound on how often each player is guaranteed to play, then best responses will not converge, simply because the adversary can stop some fixed subset of players from playing. Second, the effect of e.g. each player playing every k rounds simply appears as a multiplicative factor in the convergence times.

2.2 Distributed complexity theory and locally verifiable labelings

Our work establishes a connection between graphical games and the *LOCAL* model of computation [35, 39]). We are given a fixed network $N = (V, E)$ (a graph) connecting $n = |V(N)|$ nodes. To identify the nodes, each node gets an $O(\log n)$-bit unique name as an input. We will also consider the randomized LOCAL model, where instead each node has access to its own private source of random bits. The nodes then collaboratively aim to solve a given task, but can only communicate with their neighbors. The computation proceeds in *synchronous* rounds. In each round each node in the graph can (1) send a message to each of its neighbors (there is no bound on the message size), (2) receive a message from each neighbor, and (3) perform local computations (there is no bound on the complexity of these local computations). Initially the nodes do not know anything about the input network. Each node is responsible for computing its own part of the output.

Running time is measured in the number of communication rounds until all nodes have stopped and announced their outputs. The *running time* of a distributed algorithm (over a family $\mathcal{N}$ of networks) is the maximum running time over all networks and inputs (identifiers, random bits). The *distributed complexity* of a problem P for a family $\mathcal{N}$ of networks is the running time of the optimal algorithm for solving P over $\mathcal{N}$.

We note that the crucial property of the LOCAL model is that in T communication rounds each node can gather information from only its T-hop neighborhood in the network. Information outside this radius cannot affect the actions of a node, due to requiring a longer time to propagate. We denote this T-hop neighborhood of a node v by $B(v, T)$, and the radius-1 neighborhood simply by $B(v)$.

We study a class of problems called *locally verifiable labelling (LVL)* problems. This is a generalization of *locally checkable labellings (LCLs)* [36], a family of problems that has been studied extensively in recent years (for example [11, 16, 17, 5, 14, 10, 4]). These are the class of problems that can be *verified* efficiently with a distributed algorithm: a solution is correct if and only if it is correct in the constant-radius neighborhood of each node. Conversely, if a solution is not correct, at least one node can detect this by looking at its constant-radius neighborhood. For example, coloring is locally verifiable, as each node can compare its color with its neighbors and verify that the colors differ.

As we will show in Section 3, LVLs also capture the Nash equilibria of graphical games (as computational problems). This follows naturally from the fact that the stability of each agent only depends on the strategies of its neighbors.

An important result in the theory of distributed computing is a "complexity gap" of LVLs [16]. It allows us to classify LVLs into *easy* and *hard* problems:

▶ **Fact 1** (Chang et al. [16], Informal). *There is a superexponential gap in the possible distributed complexities of LVL problems. Easy problems below this gap can be solved in almost constant time, and hard problems above this gap require logarithmic time.*

The running time for easy problems is at most the iterated logarithm of the size of the network – an extremely slowly growing function. We say that an algorithm with this running time is *fast*. Hard problems require at least logarithmic time in the size of the network. We call such running times *slow*. For more details, see Appendix A.2.

3 Distributed computing and graphical games

Our work is motivated by our observation that graphical games and local algorithms are fundamentally connected. In particular, we will show that all Nash equilibria (and in fact all equilibria that are based on local information only) are LVLs. Thus, if we assume that agents playing a game converge to an equilibrium, they are implicitly solving the corresponding computational task.

We note that the LOCAL model of distributed computing is particularly well-suited for realizing this connection. First, it is possible to prove *unconditional* impossibility results in the LOCAL model. Existing results cover many LVLs that are potentially interesting from the perspective of game theoretical applications (see e.g. [11, 35, 15, 6]). Second, any such impossibility results proven in the LOCAL model apply to a wide range of more realistic settings. In particular, they apply to game models where the play of the agents is constrained by the available information. This is due to the fact that algorithms in the LOCAL model are only limited by information propagation.

We show how to transfer existing impossibility results for computational tasks in the distributed setting to a model of games. To do this, we first establish an equivalence between the Nash equilibria of graphical games and locally verifiable labellings. Then, to transfer impossibility results to the model of fair sequential best responses, we show that the LOCAL model can simulate best responses. This implies that if all Nash equilibria of a game are hard to compute as LVLs, then the best-response dynamics cannot converge fast to these equilibria. Note that we study a notion of *families of graphical games* as an asymptotic generalisation of single games. These are defined formally in Appendix A.3. In what follows, when referring to a graphical game, we actually mean a family of graphical games.

Theorem 1 then states the foundational observation for our work: the Nash equilibria of graphical games can be seen as locally verifiable labelling problems.

▶ **Theorem 1** (Informal). *The Nash equilibria of a family of graphical games uniquely define an LVL.*

The LVL is constructed by taking all locally stable strategy assignments: Each node accepts a labelling if and only if it has no better response with respect to the utility function. We illustrate the correspondence in Figure 2.

Theorem 1 means that we can study LVLs to understand graphical games: We can, e.g., ask if all Nash equilibria of a game are hard to compute. If they are, this fact has implications for the underlying game. A formal version of the theorem is given in Appendix A.3.

3.1 Bounding convergence time of best responses

Next we discuss how impossibility results about LVLs can be turned into lower bounds for the convergence time of best responses. We show that best responses can be simulated in the LOCAL model. Formal versions of these results with their proofs are given in Appendix A.4.

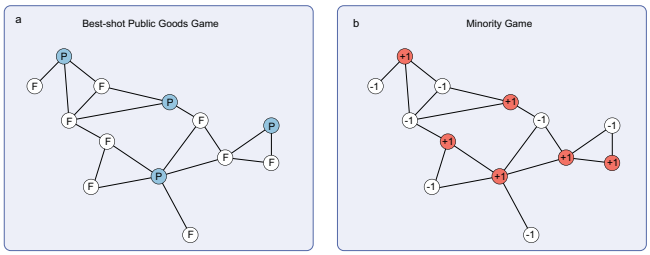

Figure 2 We illustrate the correspondence between Nash equilibria of graphical games and LVLs for the games described in Section 4. **a,** In the best-shot public goods game, nodes can pay a cost to produce a good (P) or to forego producing (F), relying on neighbors to produce. The Nash equilibria of this game are maximal independent sest of producers. **b,** The minority game gives players a choice between two resources. Their goal is to choose the opposite of the majority of their neighbors. The Nash equilibrium is a locally optimal cut: players have at least as many neighbors playing the opposite strategy as the same strategy.

▶ **Theorem 2** (Informal). *Best-response dynamics can be efficiently simulated in the LOCAL model.*

Proof sketch. The high-level idea is to compute an order of play such that many agents can move in parallel without affecting each other. Such an order is given by a suitable coloring of the underlying network. Each color class can move in parallel and the outcome is the same as if they had moved one by one. A coloring can be computed fast, it needs to be computed only once, and given a e.g. k-coloring one round of fair best responses can be simulated in k rounds in the LOCAL model. ◀

Since convergence time of best-response dynamics is defined as the worst case over all orders of play, Theorem 2 implies that if best-response dynamics converge fast, then there is a fast distributed algorithm as well. Conversely, if no fast algorithm exists, best-response dynamics cannot converge fast either.

▶ **Corollary 3** (Informal). *If the Nash equilibria of a family of games, as an LVL, have distributed complexity $\Omega(T)$, then best-response dynamics require $\Omega(T) - O(\log^* n)$ fair rounds to converge for that game.*

We want to highlight a particular implication of Corollary 3: together with the complexity gap of LVLs (Lemma 1) it implies a gap in the possible convergence times of best-response dynamics as well.

▶ **Corollary 4** (Informal). *Best-response dynamics of a graphical game either converge fast or take at least logarithmic time to converge.*

We finish this section by noting that in fact the applicability of impossibility results from distributed computing is much more general. Consider the following model of *local* dynamics: agents play one at a time, and when it is an agent's turn to move it decides its strategy based on all information inside its constant-radius neighborhood, for an arbitrarily large constant. There are no restrictions on the algorithm the agent uses to compute its decision, except that it is a function of the information available to it.

4 Classification of graphical games

In this section we describe a classification of families of graphical games based on Corollary 4, and then study examples of graphical games from three different classes.

We use two dimensions for this classification. The first dimension is computational hardness. Consider a graphical game family: computing a Nash equilibrium is either an easy or a hard problem. The second dimension is the convergence time of best-response dynamics: they also either converge fast or slow (at least logarithmic time). From this, we get the following types of game families:

1. There is a graphical game family such that a Nash equilibrium can be computed fast and best-response dynamics converge in constant number of rounds.
2. There is a graphical game family such that all Nash equilibria are hard to compute and therefore best-response dynamics converge slow.
3. There is a graphical game family such that a Nash equilibrium can be computed fast but best response dynamics converge slowly.

However, note that one combination is not possible: if computing a Nash equilibrium is hard, then best-response dynamics cannot converge fast by Corollary 3: distributed algorithms are at least as powerful as best-response dynamics.

4.1 Best-shot public goods game

This game family deals with the provision of public goods such as finding a cure for a disease or filling an important supply, assuming that only the maximal contribution counts towards the provision level instead of the sum of all players' contributions [18, 9]. In contrast to public goods games often studied as social dilemmas, i.e. where the social optimum is reached by all agents cooperating (producing) despite the incentive to do nothing, the problem here is not only one of free-riding, but also one of coordination. Agent groups have to figure out which of them should optimally be the one to provide the good, in order to avoid redundant costs. Here we consider the simplest version of the game where agents only have two choices, to provide a good, or not to provide it. More formally, each agent has two possible actions, i.e. $A_i = \{P, F\}$. Agents' utilities u_i are as following: if a focal agent plays F and one of their neighbors plays P, the utility is $u_i = 1$. If the agent plays P, $u_i = 1 - c$ (for some $0 < c < 1$), but if they and all their neighbors play F, the utility is 0. Simply put, providing the public good is costly, and it is preferable for an agent to have a neighbor do so; however, they are still better off providing it themselves than if nobody in their neighborhood does so.

The correspondence of this game family with distributed graph problems is a prominent one: the Nash equilibria of the best-shot public goods game correspond to *maximal independent sets* of agents playing P [9]. The set is independent (i.e. no two agents with P are adjacent), as two adjacent agents with the strategy P would have incentive to choose F. On the other hand the set is maximal (i.e. each agent plays P or has a neighbor that plays P) as otherwise such an agent would have incentive to play P.

Maximal independent set is an "easy" LVL [38]. Correspondingly, best responses converge in two fair rounds for the best-shot public goods game. We can compare this with the complexity analysis of best-shot public goods games by [32], and point out that our approach considers a natural concept of *distributed* complexity whereas previous work usually takes a different, more centralized view.

▶ **Theorem 5.** *Fair best responses in the best-shot public goods game converge in two rounds from any initial configuration.*

Proof. Assume that the system starts with some arbitrary strategy profile. We claim that after the first round the set of agents playing P is independent and after the second round it is maximal.

Assume that after the first round neighboring agents u and v play P. Then the one that played last would have seen that the other plays P, and their best response is F. Therefore the set of agents playing P is independent. In the second round no agent will switch from P to F, as all of their neighbors play F. If an agent plays F and their neighbors play F it will switch to P. After two rounds the agents playing P form a maximal independent set, which is a Nash equilibrium. ◀

The best-shot public goods game falls into the first category of our classification: Nash equilibria are easy to compute and best-response dynamics converge fast.

4.2 Minority game

Minority game is a basic anti-coordination and potential game (also called social game) [13, 8] where agents attempt to do the opposite of what their neighbors are doing. That is, they attempt to anti-coordinate with what the majority of their surrounding co-players do. This can be seen e.g. as a model competition for limited resources.

Agents have two possible actions, i.e. $A_i = \{-1, 1\}$. A focal player's utility u_i is defined as $u_i(a) = 1 + |\{j \in B(i) \setminus \{i\} \colon a_j \neq a_i\}| - |\{j \in B(i) \setminus \{i\} : a_j = a_i\}|$, i.e. the difference between the number of neighbors with a different label and the same label, plus 1 (for technical reasons to avoid Nash equilibria with utility 0).

The Nash equilibria correspond to strategy profiles where each agent has at least as many neighbors playing the opposite strategy as the same strategy. In this game, we again have a correspondence with a prominent graph problem: in distributed computing, the corresponding LVL is known as locally optimal cut. A *cut* is a binary labelling of the nodes, where an edge is said to be cut if its endpoints have different labels. A maximum cut maximises the number of cut edges, and a cut is locally optimal if no node can flip its label to improve the cut. Locally optimal cut is known to be a hard problem [6]. This immediately implies that best-response dynamics must also converge slow in minority games (Corollary 3). Minority games therefore fall into the second category of our classification: computing a Nash equilibrium is hard and best-response dynamics converge slowly.

4.3 Coloring game

Coloring games [30] are another family of basic anti-coordination games: each agent must choose a color from some palette of size k and it gains utility if and only if none of its neighbours choose the same color. Coloring games can be used to model scheduling and other resource allocation scenarios.

For a positive integer k, define the family of k-coloring games as follows: the action set is $A_i = \{1, 2, \ldots, k\}$ for each agent i. The utility function is $u_i(a) = 1$ if $a_i \neq a_j$ for all neighbours j of i, and 0 otherwise. That is, an agent receives utility 1 if it is properly colored, and 0 otherwise.

An important threshold for coloring games happens with $k = \Delta$ or $k = \Delta + 1$, where Δ is the maximum degree of the underlying network. If $k \geq \Delta + 1$, a k-coloring can be computed fast [7] and best responses also converge in one fair round. To see the latter, consider an arbitrary player making its move: since it has at most Δ neighbors, these neighbors can use at most Δ colors and it can always find a *free color* that gives it utility 1. After each agent has played once, everyone has chosen a free color, and this is a Nash equilibrium.

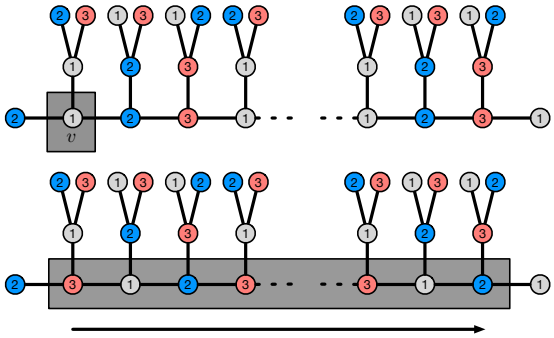

Figure 3 A lower bound construction for the convergence time of best-response dynamics for the Δ-coloring game. Top: initial configuration. Bottom: final stable configuration. Initially only the highlighted agent v is unstable: it has 3 as a free color. When it changes its strategy, it causes a cascade of agents, highlighted on the bottom, to change their strategies.

In contrast, finding a Δ-coloring is hard [11, 16]. This, however, does not necessarily mean that k-coloring games cannot converge fast. A greedy solution is no longer guaranteed, as the Δ neighbors of an agent could use all Δ colors. Thus the Nash equilibria of the coloring game do not match one to one with proper colorings. There exist Nash equilibria that are not proper colorings. This observation has an important implication, as there is a simple fast algorithm for finding a Nash equilibrium of any Δ-coloring game.

▶ **Theorem 6.** *There is fast algorithm that computes a Nash equilibrium of the Δ-coloring game, and the total utility of the Nash equilibrium is within factor $(1-\varepsilon)$ of the optimum.*

We describe and analyse the algorithm in the full version of the paper. In contrast, best-response dynamics may take up to linear time to converge:

▶ **Theorem 7.** *When $k \leq \Delta$, the worst-case convergence time of best responses for the k-coloring game is $\Omega(n)$.*

The idea for the proof is illustrated in Figure 3. The full proof is given in the full version of the paper.

This example shows that best-response dynamics can be computationally much less powerful than even simple distributed algorithms. In the next section we will show how distributed algorithms can be used as a tool to incentivise coordination and fast convergence for the coloring game.

5 Simulation games: incentivizing algorithmic cooperation

In this section, we introduce a novel way of incentivizing desired behavior in best-response dynamics. We show how to construct *simulation games*, where the best responses simulate a distributed algorithm. The Nash equilibria of simulation games then correspond to the outputs of the algorithm.

We illustrate two potential use cases for such games:

1. If a game has efficiently computable Nash equilibria, but best-response dynamics fail to converge, simulation games can be used to ensure convergence.
2. When best-response dynamics may converge to undesirable equilibria, simulation games can be used to select the Nash equilibria computed by an efficient distributed algorithm.

To incentivise the agents we assume they are guaranteed a certain utility if they follow the simulation game. This is the cost of the game, and do not try to optimise it. It is an interesting open question when simulation games could be made *self-sufficient*: the cost of incentive payments would be less than the utility gained from computing a better outcome. We present a case study in Section 5.2.

Simulation games always converge fast, requiring just one fair round. The drawback is that each agent needs to know the structure of the network and the strategies of its neighbours up to some constant distance, requiring additional means of communication.

The strategic guarantee of simulation games is that at each step, the best response of each agent is to continue to run the simulation, making these games strategy-proof. We note that even though there are multiple equilibria, we define them such that they all have the same value for a single agent. Therefore at no point should the agent take any action but their best response. Now if e.g. the agents would also get the utility from the outcome computed by the simulation game, this would no longer hold: there can be multiple equilibria, and some of these might be better than others. Agents could potentially have incentive to do something else than best response in order to affect the actions of the other agents.

5.1 Simulation games: overview

Simulation games are always based on the existence of a fast distributed algorithm. The algorithm defines the action sets and utility functions of the agents. Given an underlying network, these define a game where best responses converge to a simulation of the algorithm.

▶ **Theorem 8** (Informal). *Assume a fast distributed algorithm solves an LVL P for family of networks $\mathcal{N}$. Then for each network there exists a simulation game where best-responses converge in one round and Nash equilibria correspond to solutions computed by the algorithm.*

Simulation games are based on a special property of fast distributed algorithms: they can always be decomposed into a normal form with two parts [16]. In the first part, the input network is colored. In the second part, a specific version of the algorithm is applied with the coloring as input. The role of the coloring is to insulate the algorithm from the size of the input: we can use the same, constant-time algorithm on all networks. This makes it possible to encode the algorithm into a game.

5.1.1 Construction of simulation games

On a high level, simulation games are constructed as follows. We are given a network N and a fast distributed algorithm $\mathcal{A}$, with some constant running time T, as input. While T is a constant independent of the size of N, it can be larger than 1, the distance to which we assume agents in best-response dynamics "see". To simulate a distributed algorithm, the agents must receive information from some larger distance that is linear in T. This requires additional communication. To be consistent with the definition of graphical games, we model this communication by adding edges to N: the simulation game G is played on a network N' that is obtained by connecting all nodes in N within some specific constant distance $R = O(T)$. In addition, each agent initially knows the structure of its R-neighborhood in N.

The action sets are defined in two parts. The first part corresponds to all possible colorings of the neighbors of the agent. These are used to encode the first part in the normal form of the fast algorithm. The second part corresponds to the possible outputs of the algorithm. In addition there is an empty action, and we assume all agents start with this as their strategy.

The utility function is then used to incentivise correct simulations. For the first part, agents receive utility only if they choose a coloring of their R-neighborhood that is (1) a proper coloring (all agents have different color, a technical requirement) and (2) the coloring is consistent with the colorings chosen by other agents – all neighbors of a specific agent v choose the same color for v, and v agrees. The first part ensures that there is a properly defined notion of coloring for the T-neighborhood of each agent. For the second part, agents receive utility if they choose the output label that the algorithm $\mathcal{A}$ would choose with the coloring as input. This ensures that the agents choose an outcome $\mathcal{A}$ could have computed.

From this description we can see how the strategies will encode the output of $\mathcal{A}$. In particular, if $\mathcal{A}$ computes a Nash equilibrium of some game, then we can also directly project the Nash equilibria of the simulation game into Nash equilibria of the original game by considering only the output label.

Proof of Theorem 8, sketch. The most important properties of simulation games are that (1) they converge to Nash equilibria that correspond to outputs of the algorithm, and (2) they converge fast.

Initially all agents start with the empty strategy. When it is an agents time to move, it always gains utility by choosing an action that corresponds to a proper coloring of its neighbors. We ensure that there are enough colors so that this is always possible. Additionally, since the initial state is chosen to be empty, there are no conflicts with respect to the colors and agents always prefer to choose a coloring that is consistent with the colorings already chosen by other agents.

This ensures that in one fair round agents have chosen a proper coloring of the network. Given the coloring, the output labels are already defined: the agents only gain utility if they choose the output chosen by $\mathcal{A}$ given the input coloring. This ensures that the Nash equilibrium of the simulation game encodes a possible execution of the algorithm $\mathcal{A}$. Once the agents have reached one simulation, they have no incentive to change: all equilibria yield the same utility. Choosing the output label does not require additional communication: it can be chosen based on the coloring chosen by the agent. Therefore the best responses converge in one round. ◀

A formal description of simulation games and their construction is given in Appendix B.

5.2 Case study: coloring game

In this section we show how simulation games can be used to both ensure convergence and select equilibria in coloring games (see Section 4.3). Formal proofs are given in the full version.

First, consider the Δ-coloring game on general networks of maximum degree Δ. As shown in Section 4.3, this is a game where best-response dynamics require $\Omega(n)$ fair rounds to converge (Theorem 7). There is, however, an algorithm for efficiently computing a Nash equilibrium where $(1-\varepsilon)$-fraction of optimum total welfare (Theorem 6).

This shows that by constructing the simulation game of the algorithm from Theorem 6, we guarantee both fast convergence and almost optimal total utility.

▶ **Theorem 9** (Informal). *There is a simulation game that computes a Nash equilibrium of the Δ-coloring game with utility $(1-\varepsilon)$-fraction from the optimum.*

If we restrict the graph class to d-dimensional toruses, we can do even better. By previous work, it is known that for any constant d, there exists a fast algorithm for 4-coloring of d-dimensional toruses [12].

Consider the coloring game on two-dimensional n-by-n torus networks. That is, the set of nodes consists of $v_{i,j}\colon i,j \in \{0,1,\ldots,n-1\}$, there is an edge between nodes $v_{i,j}$ and $v_{i,k}$ if $k = (j+1) \bmod n$, and there is an edge between nodes $v_{i,j}$ and $v_{k,j}$ if $k = (i+1) \bmod n$. The complexity of k-coloring is completely understood in this setting:

▶ **Theorem 10** ([12]). *Computing a k-coloring on d-dimensional torus is easy when $k \geq 4$ and hard when $k = 2$ or $k = 3$.*

Both the 2-coloring problem and the 3-coloring problem are global (i.e. require $\Omega(n)$ rounds to compute). By Theorem 16 this implies that best responses also require at least $\Omega(n)$ rounds to converge. Even when best-response dynamics converge, they may converge to a Nash equilibrium where some agents are not properly colored.

In contrast, by Theorems 10 and 8, when $k \geq 4$ for each d-dimensional torus there exists a simulation game such that best-response dynamics converge in one round to an optimal solution: all agents are properly colored.

▶ **Theorem 11** (Informal). *There is a simulation game that computes a Nash equilibrium with optimal total utility for the 4-coloring game on any d-dimensional torus.*

Theorems 9 and 11 demonstrate how simulation games can be applied to games that have efficiently computable Nash equilibria but best-response dynamics do not converge efficiently.

6 Inefficiency of best responses

In this section we introduce and study a distributed analogue of Price of Anarchy [33]: what is the price of greedy behavior compared to optimal *distributed* coordination? The classic price of anarchy compares the total welfare of the optimal assignment to the worst Nash equilibrium. In a distributed setting, in particular, both of these are typically hard to compute. We want to replace these measures with more reasonable distributed analogues: what can be computed efficiently with locality T? In this way, our notion can be viewed as a refinement of the classic PoA concept from a distributed perspective.

Importantly, as we have seen in Section 3, efficient convergence is not possible in all graphical games. This means that an analogue of worst Nash equilibrium does not necessarily even exist! We consider a specific model of play, best-response dynamics, and compare the evolution of the state over time with respect to the best solution that can be computed with the same complexity. This leads to the notion of *inefficiency of best responses (IoBR)*.

This notion of inefficiency is computational inefficiency with respect to a time bound T. As a computational notion, it stands in contrast to the Price of Anarchy, which is an existential notion. Hence, it must be defined over families of games: a single game can always be solved perfectly by an algorithm that specifically encodes a solution to that game.

We show that there exist games such that we can bound the inefficiency of best responses away from the Price of Anarchy. This illustrates that the Price of Anarchy does not always fairly reflect the quality of solutions computed by best responses when time constraints are taken into account.

6.1 Defining inefficiency of best responses

To measure the performance of best responses, we compare the total welfare of a solution produced by T fair rounds of best responses to the best solution that can be computed in T rounds. The running time bound T is a function of the size of the game. We assume that the system starts from a random initial strategy profile. We consider random initial strategy

profiles, as worst-case strategy profiles make any computational arguments moot: the initial strategy profile could be the worst Nash equilibrium, even if it is hard to compute. To have a fair comparison, we consider what can be computed by randomised distributed algorithms.

Consider a family of games $\mathcal{G}$ and a distributed algorithm F with running time $T = T(n)$. For each $G \in \mathcal{G}$, let $F(G)$ denote total utility of the solution computed by F on G. Let $\mathrm{BR}(G, T)$ denote total utility of the solution produced in G by T rounds of best responses.

We define the T-inefficiency of best responses as

$$\mathrm{IoBR}(\mathcal{G}, T) = \max_{F \in \mathcal{F}_T} \min_{G \in \mathcal{G}} \frac{F(G)}{E[\mathrm{BR}(G, T)]},$$

where $\mathcal{F}_T$ denotes the family of all distributed algorithms with running time at most T. The definition compares the worst relative performance (over games of $\mathcal{G}$) of the best algorithm to the expected performance of best responses. The idea is that $\mathcal{G}$ represents some family of "similar" games: they are an asymptotic analogue of a single game in the definition of the Price of Anarchy.

To estimate the quantity $\mathrm{IoBR}(\mathcal{G}, T)$ we will bound all $F(G)$ from above using computational arguments, and bound $\mathrm{BR}(G, T)$ from below using both arguments about the behavior of best-response dynamics *and* computational arguments.

Note that when we consider the best strategy profile that is computable in T communication rounds, we consider distributed algorithms for *optimization* problems. That is, the algorithms might not compute a solution corresponding to some Nash equilibrium, but more generally any strategy assignment that tries to optimize the total welfare of all agents.

In the next section we show how the inefficiency of best responses can be estimated using tools from distributed computing.

6.2 Case study: best-shot public goods game

We begin by analysing the inefficiency of best responses in the public goods game. We show that the Price of Anarchy can be bounded away from the inefficiency of best responses. We prove that there exists an infinite family of best-shot public goods games such that even though good and bad solutions exist, no distributed algorithm can compute them efficiently. Therefore best responses cannot produce the bad solutions either. The following theorem states the outcome of our analysis for the best-shot public goods game.

▶ **Theorem 12** (Informal). *There exists a family $\mathcal{G}$ of public goods games such that for all $G \in \mathcal{G}$:*

1. *Price of Anarchy is the same for all $G \in \mathcal{G}$, and*
2. *Inefficiency of best responses, compared to fast algorithms, for $\mathcal{G}$ is strictly less than Price of Anarchy.*

The good solutions correspond to small *dominating sets* of producing agents: each agent has exactly one adjacent producer. The bad solutions are very redundant dominating sets: each non-producing agent has only producing neighbors.

To construct these games, we argue that there exist two families of networks:

1. In the first family, networks have both good and bad solutions, as described above.
2. In the second family, the networks are locally indistinguishable from networks of the first family, but do not have the bad and good solutions.

No algorithm can find the good or bad solutions in the second family, as they don't exist. Fast algorithms cannot distinguish between the two families. This means that fast algorithms cannot produce the good or bad solutions in the first family: if such algorithms existed, they would imply good and bad solutions for the second family as well, a contradiction.

We give a formal version of Theorem 12 and its proof in in the full version of this paper. There we also study the minority game and show a similar separation result for it: there exists a family of games such that Price of Anarchy is bounded away from the inefficiency of best responses.

7 Conclusion

In this work we introduced a novel approach to classifying the complexity of Nash equilibria in graphical games, and to understanding the convergence behavior of best-response and even general local dynamics. By establishing a connection to the analysis of distributed graph problems, we showed that the Nash equilibria of graphical games correspond to locally verifiable labelings: solutions to graph problems which are verifiable with constant round algorithms. Impossibility and complexity results provably transfer from the distributed setting to the game setting. In particular, the complexity gap observed in LVL problems also directly translates to the classification of equilibrium convergence times.

Thus, we can leverage this very natural connection to give lower bounds for convergence of best-response dynamics, to quantify the time-constrained inefficiency of best responses when convergence is slow or even absent, and to present how these results can be used for implementing mechanisms where best responses converge to a Nash equilibrium that is a solution of the corresponding graph problem. We also want to highlight once more that while our lower bounds are only strengthened by the simplicity of our examples, our results are relevant beyond the limited instances we have presented.

Our findings also relate to the open question of strategy proof algorithms for reaching equilibria in graphical games, as posed by Kearns in his 2007 survey. We note that in contrast to algorithms as in [29], our discussion indeed pertains to how agents *reach* equilibria while playing the game in a way that is rational with respect to their locally restricted knowledge. This poses an interesting avenue for further research.

In this work, we have only considered pure Nash equilibria. However, we note that our approach is much more general than that, since it first and foremost depends on information being locally restricted without making additional assumptions like the game being a potential game. One can prove similar results as in this work for *mixed Nash equilibria*, but even for different equilibrium concepts altogether, such as *correlated equilibria*. The latter correspond to a model where information is local, but can in some sense be exchanged between neighboring nodes, introducing correlated strategy distributions. A further simple extension of our model could see different local strategy update dynamics (e.g. ficitious play) at work, instead of restricting the analysis to best response dynamics only. This also highlights the connection of this research direction with evolutionary graph theory, which as a generalized approach to evolutionary dynamics features players with similarly bounded rationality [34]. Furthermore, it is also easily conceivable to analyze a far wider range of graphical games or LVLs with our approach, and to even extend the analysis to infinite graphs.

Naturally, there are limitations to our approach. Our techniques are relevant only in relatively sparse networks. Graphs of low diameter do not give impossibility results based on information propagation, as information can spread quickly. The results for such a setting would look quite different; furthermore, computation in a setting that comes closer to a centralized one where nodes do not have as strongly limited information is not well understood. Another issue lies with the fact that we only show lower bounds for convergence. This means that for instances that are not efficiently solvable, it could be the case that true convergence is far slower than what our results give. For further understanding, a complexity theory of the convergence of best-response dynamics is needed.

What we have considered in this paper should only represent a first taste for how powerful the connection between game theory and distributed computing is. We also highlight at this point that interpreting agents interacting in games on networks as a distributed system is highly intuitive. There are many possibilities to harness this, and we have only explored a very limited number of them. We hope that this work can serve as a proof of concept and be the starting point of exciting further research.

References

1 Guillermo Abramson and Marcelo Kuperman. Social games in a social network. *Physical Review E*, 63(3):030901, 2001. doi:10.1103/PhysRevE.63.030901.

2 James Aspnes, Kevin Chang, and Aleksandr Yampolskiy. Inoculation strategies for victims of viruses and the sum-of-squares partition problem. *Journal of Computer and System Sciences*, 72(6):1077–1093, 2006. doi:10.1016/J.JCSS.2006.02.003.

3 Guy Avni, Thomas A. Henzinger, and Orna Kupferman. Dynamic resource allocation games. In *Algorithmic Game Theory*, pages 153–166. Springer Berlin Heidelberg, 2016. doi:10.1007/978-3-662-53354-3_13.

4 Alkida Balliu, Sebastian Brandt, Yuval Efron, Juho Hirvonen, Yannic Maus, Dennis Olivetti, and Jukka Suomela. Classification of distributed binary labeling problems. *CoRR*, abs/1911.13294, 2019. arXiv:1911.13294.

5 Alkida Balliu, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Dennis Olivetti, and Jukka Suomela. New classes of distributed time complexity. In *Proc. 50th ACM Symposium on Theory of Computing (STOC 2018)*, pages 1307–1318. ACM Press, 2018. doi:10.1145/3188745.3188860.

6 Alkida Balliu, Juho Hirvonen, Christoph Lenzen, Dennis Olivetti, and Jukka Suomela. Locality of not-so-weak coloring. In *Proc. 26th International Colloquium on Structural Information and Communication Complexity (SIROCCO 2019)*, volume 11639 of *Lecture Notes in Computer Science*, pages 37–51. Springer, 2019. doi:10.1007/978-3-030-24922-9_3.

7 Leonid Barenboim. Deterministic $(\Delta + 1)$-coloring in sublinear (in Δ) time in static, dynamic, and faulty networks. *J. ACM*, 63(5):47:1–47:22, 2016. doi:10.1145/2979675.

8 Yann Bramoullé. Anti-coordination and social interactions. *Games and Economic Behavior*, 58(1):30–49, 2007. doi:10.1016/J.GEB.2005.12.006.

9 Yann Bramoullé and Rachel Kranton. Public goods in networks. *Journal of Economic Theory*, 135(1):478–494, 2007. doi:10.1016/J.JET.2006.06.006.

10 Sebastian Brandt. An automatic speedup theorem for distributed problems, 2019. arXiv:1902.09958.

11 Sebastian Brandt, Orr Fischer, Juho Hirvonen, Barbara Keller, Tuomo Lempiäinen, Joel Rybicki, Jukka Suomela, and Jara Uitto. A lower bound for the distributed Lovász local lemma. In *Proc. 48th ACM Symposium on Theory of Computing (STOC 2016)*, pages 479–488. ACM Press, 2016. doi:10.1145/2897518.2897570.

12 Sebastian Brandt, Juho Hirvonen, Janne H Korhonen, Tuomo Lempiäinen, Patric R J Östergård, Christopher Purcell, Joel Rybicki, Jukka Suomela, and Przemysław Uznański. LCL problems on grids. In *Proc. 36th ACM Symposium on Principles of Distributed Computing (PODC 2017)*, pages 101–110. ACM Press, 2017. doi:10.1145/3087801.3087833.

13 Damien Challet and Y-C Zhang. Emergence of cooperation and organization in an evolutionary game. *Physica A: Statistical Mechanics and its Applications*, 246(3-4):407–418, 1997. doi:10.1016/S0378-4371(97)00419-6.

14 Yi-Jun Chang, Qizheng He, Wenzheng Li, Seth Pettie, and Jara Uitto. The Complexity of Distributed Edge Coloring with Small Palettes. In *Proc. 29th ACM-SIAM Symposium on Discrete Algorithms (SODA 2018)*, pages 2633–2652. Society for Industrial and Applied Mathematics, 2018. doi:10.1137/1.9781611975031.168.

15 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An Exponential Separation between Randomized and Deterministic Complexity in the LOCAL Model. In *Proc. 57th IEEE Symposium on Foundations of Computer Science (FOCS 2016)*, pages 615–624. IEEE, 2016. doi:10.1109/FOCS.2016.72.

16 Yi-Jun Chang, Tsvi Kopelowitz, and Seth Pettie. An exponential separation between randomized and deterministic complexity in the LOCAL model. *SIAM J. Comput.*, 48(1):122–143, 2019. doi:10.1137/17M1117537.

17 Yi-Jun Chang and Seth Pettie. A time hierarchy theorem for the LOCAL model. *SIAM J. Comput.*, 48(1):33–69, 2019. doi:10.1137/17M1157957.

18 Todd L Cherry, Stephen J Cotten, and Stephan Kroll. Heterogeneity, coordination and the provision of best-shot public goods. *Experimental Economics*, 16(4):497–510, 2013. doi:10.1007/s10683-012-9349-1.

19 Simon Collet, Pierre Fraigniaud, and Paolo Penna. Equilibria of games in networks for local tasks. In *Proc. 22nd International Conference on Principles of Distributed Systems (OPODIS 2018)*, volume 125 of *LIPIcs*, pages 6:1–6:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPICS.OPODIS.2018.6.

20 Constantinos Daskalakis and Christos H Papadimitriou. Computing pure nash equilibria in graphical games via markov random fields. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 91–99, 2006. doi:10.1145/1134707.1134718.

21 Constantinos Daskalakis and Christos H. Papadimitriou. On a network generalization of the minmax theorem. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming*, pages 423–434, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi:10.1007/978-3-642-02930-1_35.

22 Edith Elkind, Leslie Ann Goldberg, and Paul Goldberg. Nash equilibria in graphical games on trees revisited. In *Proc. 7th ACM Conference on Electronic Commerce*, pages 100–109. Association for Computing Machinery, 2006. doi:10.1145/1134707.1134719.

23 Patrik Floréen, Petteri Kaski, Valentin Polishchuk, and Jukka Suomela. Almost stable matchings by truncating the gale-shapley algorithm. *Algorithmica*, 58(1):102–118, 2010. doi:10.1007/S00453-009-9353-9.

24 David Gale and Lloyd S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962. doi:10.2307/2312726.

25 Mohsen Ghaffari and Hsin-Hao Su. Distributed Degree Splitting, Edge Coloring, and Orientations. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA 2017)*, pages 2505–2523. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974782.166.

26 Samuel Ieong, Robert McGrew, Eugene Nudelman, Yoav Shoham, and Qixiang Sun. Fast and compact: A simple class of congestion games. In *Proc. 20th National Conference on Artificial Intelligence – Volume 2 (AAAI 2005)*, pages 489–494. AAAI Press, 2005. URL: http://www.aaai.org/Library/AAAI/2005/aaai05-077.php.

27 Matthew O Jackson and Leeat Yariv. Diffusion of behavior and equilibrium properties in network games. *American Economic Review*, 97(2):92–98, 2007. doi:10.1257/aer.97.2.92.

28 Matthew O Jackson and Yves Zenou. Games on networks. In *Handbook of game theory with economic applications*, volume 4, pages 95–163. Elsevier, 2015. doi:10.1016/B978-0-444-53766-9.00003-3.

29 Michael Kearns. Graphical games. *Algorithmic game theory*, 3:159–180, 2007. doi:10.1017/CBO9780511800481.009.

30 Michael Kearns, Siddharth Suri, and Nick Montfort. An experimental study of the coloring problem on human subject networks. *Science*, 313(5788):824–827, 2006. doi:10.1126/science.1127207.

31 Michael J. Kearns, Michael L. Littman, and Satinder P. Singh. Graphical models for game theory. In *Proc. 17th Conference in Uncertainty in Artificial Intelligence (UAI 2001)*, pages 253–260, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. URL: https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=107&proceeding_id=17.

32 Zohar Komarovsky, Vadim Levit, Tal Grinshpoun, and Amnon Meisels. Efficient equilibria in a public goods game. In *Proceedings of the 2015 IEEE / WIC / ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT) – Volume 01*, WI-IAT '15, pages 214–219. IEEE Computer Society, 2015. doi:10.1109/WI-IAT.2015.91.

33 Elias Koutsoupias and Christos H. Papadimitriou. Worst-case equilibria. In *Proc. 16th Annual Symposium on Theoretical Aspects of Computer Science (STACS 1999)*, volume 1563 of *Lecture Notes in Computer Science*, pages 404–413. Springer, 1999. doi:10.1007/3-540-49116-3_38.

34 E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433:312–316, 2005. doi:10.1038/nature03204.

35 Nathan Linial. Locality in Distributed Graph Algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992. doi:10.1137/0221015.

36 Moni Naor and Larry Stockmeyer. What Can be Computed Locally? *SIAM Journal on Computing*, 24(6):1259–1277, 1995. doi:10.1137/S0097539793254571.

37 Luis E Ortiz and Michael Kearns. Nash propagation for loopy graphical games. *Advances in neural information processing systems*, pages 817–824, 2003. doi:10.5555/2968618.2968720.

38 Alessandro Panconesi and Romeo Rizzi. Some simple distributed algorithms for sparse networks. *Distributed Computing*, 14(2):97–100, 2001. doi:10.1007/PL00008932.

39 David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. Society for Industrial and Applied Mathematics, 2000. doi:10.1137/1.9780898719772.

40 Tim Roughgarden. Routing games. *Algorithmic game theory*, 18:459–484, 2007.

41 Grant R Schoenebeck and Salil Vadhan. The computational complexity of nash equilibria in concisely represented games. *ACM Transactions on Computation Theory (TOCT)*, 4(2):1–50, 2012. doi:10.1145/2189778.2189779.

42 Richard Southwell, Yanjiao Chen, Jianwei Huang, and Qian Zhang. Convergence dynamics of graphical congestion games. In Vikram Krishnamurthy, Qing Zhao, Minyi Huang, and Yonggang Wen, editors, *Game Theory for Networks*, pages 31–46. Springer Berlin Heidelberg, 2012. doi:10.1007/978-3-642-35582-0_3.

43 Alexander J Stewart, Mohsen Mosleh, Marina Diakonova, Antonio A Arechar, David G Rand, and Joshua B Plotkin. Information gerrymandering and undemocratic decisions. *Nature*, 573(7772):117–121, 2019. doi:10.1038/S41586-019-1507-6.

44 David Vickrey and Daphne Koller. Multi-agent algorithms for solving graphical games. *AAAI/IAAI*, 2:345–351, 2002. URL: http://www.aaai.org/Library/AAAI/2002/aaai02-053.php.

A Preliminaries

A.1 Defininig LVLs

Formally LVLs are defined as follows.

▶ **Definition 13** (Locally verifiable labelings). *An LVL P is a computational problem parametrized by a constant maximum degree Δ, and consists of an input alphabet Σ, an output alphabet Γ, and a set of configurations $\mathcal{C}$. The input alphabet Σ is finite set of labels, and the output alphabet Γ is possibly an infinite set (this generalises LCLs). Each configuration $C \in \mathcal{C}$ is a graph G centered on some node v with radius at most k for some constant k (the verification radius of P). Each node of C is labelled with some element $\sigma \in \Sigma$ and $\gamma \in \Gamma$.*

An instance of P is a pair (N, x) where $N = (V, E)$ is a network of maximum degree Δ and x is a mapping $x\colon V \to \Sigma$. If P has no input labels (i.e. $|\Sigma| = 1$), any network N is an instance of P. We say that a mapping $f\colon V \to \Gamma$ is a solution to P on (N, x) if and only if each k-neighborhood of N labelled with x and f is a configuration in $\mathcal{C}$.

We note here that in this work, we only consider LVLs with radius $k = 1$, as this matches the definition of graphical games. The definition assumes a constant bound on the maximum degree of the input network. Throughout this work, we will assume that networks have a constant maximum degree Δ. This is important assumption from the perspective of distributed complexity theory, as it is very different in dense networks (large maximum degree).

A.2 Complexity gap of LVLs

There is a gap in the possible complexities of LVL problems: they can either be solved very fast or require logarithmic time.

▶ **Fact 2** (Chang et al. [16]). *On general bounded-degree graphs, i.e., if the maximum degree of the graph N is bounded by a constant Δ, the deterministic distributed complexity of an LVL is either $O(\log^* n)$, or $\Omega(\log n)$. The randomized complexity is either $O(\log^* n)$ or $\Omega(\log \log n)$.*

Here, $\log^* n$ is the *iterated logarithm* (pronounced "log-star"), a function which grows significantly more slowly than the logarithm: e.g. $\log^*$ of the number of atoms in the observable universe is 5. Formally, $\log^* n$ is defined as:

$$\forall x \leq 2 : \log^* x := 1, \quad \forall x > 2 : \log^* x := 1 + \log^*(\log x)$$

The complexity gap on bounded-degree networks is also the best possible: there exists an LVL such that its deterministic complexity is $\Theta(\log n)$ and the randomized complexity is $\Theta(\log \log n)$ [16, 11, 25]. This also proves that the deterministic and randomized complexities of an LVL can be *exponentially separated*.

On other graph families, the complexity gap between easy and hard problems can be even larger. For example, on paths and cycles LVLs have complexity either $O(\log^* n)$ or $\Theta(n)$ [16], and on two-dimensional grids and toruses either $O(\log^* n)$ or $\Omega(\sqrt{n})$ [12].

Throughout this paper, we will call a distributed algorithm with complexity $O(\log^* n)$ *fast*.

A.3 Correspondence of LVLs and Nash equilibria of graphical games

To study the asymptotic hardness of computing equilibria, we define *families* $\mathcal{G}$ of graphical games, parametrized by a constant degree bound Δ and some (possibly infinite) set $\mathcal{N}$ of networks of maximum degree Δ. For example, this could be all networks of maximum degree Δ, or all trees of maximum degree Δ.

▶ **Definition 14** (Family of graphical games). *A family of graphical games then consists of all possible games on $\mathcal{N}$ generated from some base set of rules. The* type *of an agent is a pair (A, u), where A is an action set and u is a utility function. Fix Δ finite collections of types $\mathcal{T} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_\Delta)$, one for each possible degree of an agent. Then the family of games $\mathcal{G} = \mathcal{G}(\mathcal{N}, \Delta, \mathcal{T})$ is defined as the set of all possible games on each $N \in \mathcal{N}$ such that for each degree d, each agent of degree d is assigned an action set and utility function according to some type in $\mathcal{T}_d$.*

Using this definition we can construct a locally verifiable labelling that captures exactly the Nash equilibria of the family of graphical games.

▶ **Theorem 15.** *Let $\mathcal{G}$ be a family of graphical games. There is a locally verifiable labelling $P(\mathcal{G})$ such that for each $G \in \mathcal{G}$ the pure strategy Nash equilibria of G correspond exactly to solutions of P.*

Proof. This follows from the locality of the utility functions of graphical games: whether each agent is stable only depends on the strategies of its neighbours, and a strategy profile is a Nash equilibrium if and only if all agents are stable.

Consider an arbitrary family of graphical games $\mathcal{G}$. To construct the corresponding LVL P, do the following. The set of input labels is the union of the types of $\mathcal{G}$. The set of output labels Γ is the set of all possible pure strategies, i.e. the union of action sets of the types of $\mathcal{G}$. To construct the set of configurations C of P, let $\mathcal{B}$ denote all possible 1-neighbourhoods of $\mathcal{G}$ labelled with types, over all $G \in \mathcal{G}$. Consider an arbitrary such 1-neighbourhood $B \in \mathcal{B}$ centered at some agent i. Each agent $j \in B$ is labelled with a type (A_j, u_j). Consider all possible strategy profiles a drawn from the action sets of the agents. For each a, if a_i is a best response of i with respect to a_{-i}, add B with the input labelling given by the types and the output labelling given by a to C.

By construction, for each specific $G \in \mathcal{G}$ there exists a corresponding instance of P such that the solutions are exactly the Nash equilibria of G. ◀

It should be stressed that the correspondence does not use any properties specific to pure Nash equilibria. For example mixed strategy Nash equilibria of graphical games can also be used to define LVLs analogously: the output labels of the LVL consist of mixed strategies instead of pure strategies, as above.

A.4 Simulating best responses

In this section we show how best-response dynamics can be simulated in the LOCAL model with small overhead.

Let $G = (A, u, N)$ be a graphical game. We construct a corresponding instance in the LOCAL model by taking the network N and labelling each agent with its type. Then, if we consider the deterministic LOCAL model, an adversary assigns $O(\log n)$-bit names to the nodes. In the randomized LOCAL model, each node gets a uniformly random infinite string of bits as input instead.

We define the convergence time of best-response dynamics for a family of graphical games as the worst-case convergence time over all games. We show that if the best responses converge in $T(n)$ rounds, where n is the size of the underlying network, then this can be turned into a distributed algorithm that computes the corresponding Nash equilibrium in $O(\log^* n + T(n))$ rounds.

▶ **Theorem 16** (best responses correspondence)**.** *Fix a function T. Consider a family $\mathcal{G}$ of graphical games.*

1. *If the best-response dynamics converge on each $G \in \mathcal{G}$ in $T(n)$ rounds from a constant initial strategy profile, then there exists a deterministic distributed algorithm in the LOCAL model that solves the LVL $P(\mathcal{G})$ corresponding to the (pure) Nash equilibria of $\mathcal{G}$ on each G in $O(\log^* n + T(n))$ rounds.*
2. *If the best-response dynamics converge with high probability on each $G \in \mathcal{G}$ in $T(n)$ rounds from a random initial strategy profile, then there exists a randomized distributed algorithm in the LOCAL model that solves the LVL $P(\mathcal{G})$ corresponding to the (pure) Nash equilibria of $\mathcal{G}$ on each G in $O(\log^* n + T(n))$ rounds with high probability.*

Before proving the theorem we show a helper lemma that shows how best-response dynamics can be simulated from any specific initial state.

▶ **Lemma 17.** *Assume that $(\mathcal{A}, u, N)$ is a graphical game and a is some strategy profile. A distributed algorithm that is given a as an input can simulate T rounds of best responses, for some ordering of the play, in $O(\log^* n + T)$ rounds.*

Proof. The simulation consists of two phases. In the first phase, the nodes compute a coloring of N^2 (the virtual network obtained by connecting all nodes at distance at most 2 in N) with $k = \Delta^2 + 1$ colors. That is, each node v chooses a label $c(v)$ from $\{1, 2, \ldots, k\}$ such that any two nodes u and v within distance 2 in N have different labels $c(u) \neq c(v)$. This can be computed in $O(\log^* n)$ rounds [7].

In the second phase this coloring is treated as a *schedule*: at round j of the second phase each node with color $i = j \bmod k$ is active, applies the best response to the current strategy profile, and sends its new strategy to its neighbors. The key is that any two nodes updating their strategy at the same time do so *independently*: since they are not neighbors, their choices do not depend on each other. Therefore applying best responses at all nodes of the same color class is equivalent to letting the corresponding agents play in any sequential order: given an initial strategy profile a, all orderings produce the same strategy profile a'. Simulating all color classes one by one therefore corresponds to *some* ordering of sequential play.

Since there are $k = \Delta^2 + 1$ color classes, simulating one fair round of best responses takes k rounds in the LOCAL model. Since we assume Δ is a constant, simulating T rounds of best responses can be done in $O(T)$ rounds. With the initial coloring step we have that the total running time of the simulation is $O(\log^* n + T)$, as required. ◀

It follows that simulating best responses until convergence can be done with an additive $O(\log^* n)$ overhead.

Proof of Theorem 16. First, assume that the best responses start from a constant or worst-case initial strategy profile. Each node can simply choose the same initial value and simulate $T(n)$ rounds of best responses by Lemma 17. Since we assume that best responses converge for any order of play in $T(n)$ rounds, it follows that $T(n)$ rounds of simulation converge as well. Computing the simulation until convergence takes $O(\log^* n + T(n))$ rounds in the deterministic LOCAL model.

Next, assume that the best responses start from a random initial strategy profile. Now it is no longer possible to use deterministic algorithms to run the simulation. Using the random inputs, each node can choose a random initial strategy. Then it can simulate best responses for $T(n)$ rounds by Lemma 17. Since we assume that the best responses converge with high probability and the dynamics are deterministic given the initial configuration, the simulation also converges with high probability in $T(n)$ rounds. The simulation can be computed in $O(\log^* n + T(n))$ rounds in the randomized LOCAL model. ◀

As shown in Theorem 15, a family of graphical games $\mathcal{G}$ defines an LVL $P(\mathcal{G})$. It also defines a family of problem instances of P: each G can be seen as network N with input labelling given by the types of the agents.

▶ **Corollary 18.** *Assume that problem of solving the LVL $P(\mathcal{G})$ over the instances defined by $\mathcal{G}$ has deterministic complexity $\Omega(T(n))$ and randomized complexity $\Omega(T'(n))$, for any $T(n), T'(n) = \Omega(\log^* n)$. Then best-response dynamics for $\mathcal{G}$ require $\Omega(T(n))$ and $\Omega(T'(n))$ fair rounds to converge from a constant and a randomized initial strategy profile, respectively.*

Proof. This follows from Theorem 16: if best-response dynamics converge faster, then this can be turned into a fast distributed algorithm, a contradiction. ◀

The complexity gap of LVLs in the LOCAL model, in the context of our result, implies that if the Nash equilibria of a game are not efficiently computable, then best responses converge significantly slower.

▶ **Corollary 19.** *Let $\mathcal{G}$ be a family of games such that $P(\mathcal{G})$ cannot be solved in time $O(\log^* n)$. Then the best-response dynamics for $\mathcal{G}$ require $\Omega(\log n)$ fair rounds to converge from constant initial strategy profile, and $\Omega(\log \log n)$ fair rounds to converge from a random initial strategy profile.*

We emphasize that the deterministic and randomized convergence time lower bounds hold even if we consider best-response dynamics that start respectively from any constant or a randomized initial state.

B Simulation games

B.1 Simulation games: Overview

Simulation games are based on the existence of fast distributed algorithms: assume there exists an algorithm A with running time $T(n) = O(\log^* n)$ solving some LVL P. Then for any network N we can construct a graphical game G where best-response dynamics converge in one fair round to a Nash equilibrium that encodes the output of A on N. In particular, if A computes Nash equilibria for some family of games, then best-response dynamics in the corresponding simulation games will also converge to the Nash equilibria computed by A.

Our construction has several desirable properties.

1. Convergence time. Best responses converge in one fair round.
2. Equilibrium selection. Since best responses converge to an output of algorithm A, the resulting Nash equilibria inherit the properties of solutions computed by A. In Section 5.2 we show how this can be used to ensure convergence to a desirable subset of Nash equilibria.
3. Locality. It would be trivial to construct a new game with the above two properties: an outside observer could set a utility function structured such that agents have to choose a specific outcome. This, however, would essentially amount to a centralised authority assigning the strategies to the agents. In our construction, each agent only receives the description of the algorithm A and information about other agents up to some constant distance, independent of the size of the underlying network. This means that assuming limited local communication, the agents can set up this simulation game locally without centralised control.

Due to the locality of simulation no similar constructions exist for games that don't have efficiently solvable Nash equilibria: this would also imply efficient algorithms for the corresponding LVL problems.

On a high level, simulation games are constructed as follows: We are given a network N and an efficient algorithm $\mathcal{F}$. The actions of each agent consist of all possible correct local executions of the algorithm $\mathcal{F}$ on N. The utility of an agent is 1 if and only if its strategy is an execution that is compatible with the executions of its neighbors. As a correct execution always exists, best responses converge to such an execution when starting from an empty initial state (i.e. each agent has a special empty action).

The following theorem is the formal version of Theorem 8.

▶ **Theorem 20.** *Let $\mathcal{F}$ be a distributed algorithm that solves LVL P in time $O(\log^* n)$ in networks of maximum degree Δ. Then for each P-instance (N, x) there exists a simulation game $G = (A, u, N')$ with the following properties.*

(P1) *Best responses converge in one fair round from the empty initial strategy profile in G.*

(P2) *Nash equilibria of G correspond to legal outputs of $\mathcal{F}$.*

(P3) *Given (N, x) and $\mathcal{F}$, simulation game G can be constructed in $O(t)$ rounds with a distributed algorithm.*

A particularly interesting use case for the simulation games is when a fast algorithm computes the Nash equilibria of a graphical game.

▶ **Corollary 21.** *If $\mathcal{F}$ solves LVL $P(\mathcal{G})$ corresponding to the Nash equilibria of a family of graphical games $\mathcal{G}$, then the simulation game converges to a Nash equilibrium of $\mathcal{G}$.*

We will next give the technical details for the construction of simulation games.

B.2 Constructing simulation games

To define simulation games, we need to consider algorithms in a specific *normal form*, the existence of which is implied by the speedup result of [16]. Lemma 22 (stated in the following section) states that $O(\log^* n)$-time algorithms can be decomposed into two phases. In the first phase the algorithm computes a distance-$(2t+2)$ coloring for some constant parameter t that depends on the algorithm. Then a t-round algorithm is applied with the coloring taking the role of unique identifiers. We say that this is the t-normal form of the algorithm. In simulation games the best responses construct these colorings and then choose the output of the algorithm on that particular coloring.

Now let $\mathcal{F}$ be a distributed algorithm in t-normal form that solves LVL P. For a P instance (N, x) with maximum degree Δ, define the *simulation game of $\mathcal{F}$ on N* as $G = (A, u, N')$ as follows.

1. The set of agents are the nodes of N. Construct a new network N' by connecting two nodes u and v if and only if their distance in N is at most $4t + 2$ (some constant).
2. The actions A_v of each agent v encode the possible locally correct simulations of $\mathcal{F}$. This is defined in two parts R_v and L. The first set R_v consists of all possible labellings of the t-neighborhood of v in N with distinct colors from $\{1, 2, \ldots, \Delta^{2t+2} + 1\}$. The second part L consists of the possible output labels of P. Include the pair (r, ℓ) in A_v if and only if $\mathcal{F}$ would output ℓ at v given x as the input labelling and r as the input coloring of $B(v, t)$. In addition there is the empty action.
3. The utility functions u_v encode the correct simulations. The utility $u_v(s) = 1$ if and only if the following hold. First, the coloring $r_v \in R_v$ is *compatible* with the colorings $r_u \in R_u$ of each neighbor u with a non-empty strategy. That is, for each $w \in B_N(v, t) \cap B_N(u, t)$, we have that $r_v(w) = r_u(w)$ or r_u is empty. Second, the colorings form a proper $(2t + 2)$-hop coloring of N. That is, if we map all the compatible colorings r_v for all v onto N, then two nodes in N at distance at most $2t + 2$ have distinct colors. Since two agents are connected in N' if they are within distance $4t + 2$ in N, it is possible to encode this in u. The color assigned to each agent at distance t from some agent v must differ from the colors of other agents within distance $2t + 2$ of it. These agents can only be colored by agents within distance $4t + 2$. Note that by construction we only allow actions where ℓ corresponds to the correct output of $\mathcal{F}$ so there is no need to encode this in the utility function.

   For the empty action and for strategy profiles that do not have these properties the utility is 0.

We will show that simulation games converge in one fair round to an equilibrium that is equivalent to an equilibrium of the original game.

We also note that similar constructions do not exist for algorithm that solve hard problems. If they did, by Theorem 16 we would get a faster algorithm for solving the problem, a contradiction.

B.3 Proving the properties of simulation games

Before proving Theorem 20, we need the following technical lemma. It establishes that $O(\log^* n)$-time solvable LVLs can also be solved by algorithms in the required normal form.

▶ **Lemma 22.** *Assume that LVL P can be solved in $O(\log^* n)$ rounds by a deterministic distributed algorithm. Then there exists an algorithm $\mathcal{F}$ in the following normal form: the algorithm runs in $t = O(1)$ rounds (for some t dependent on P), it takes a $(2t+2)$-hop c-coloring, for $c = \Delta^{2t+2} + 1$, as an input, and outputs the solution to P.*

The proof follows from the speedup theorem of [16]. A similar normal form construction has been used by Brandt et al. [12].

We are now ready to prove the properties of simulation games.

Proof of Theorem 20. We show that the strategies of agents who have already played will constitute a correct partial simulation in N'. Initially this is trivially true, as all agents are assumed to start from the empty strategy.

Now assume that a is the strategy profile after some number of best responses such that the non-empty strategies agree on the color of each agent and assume that some agent v is scheduled to play. Since the utility is 0 if the agent chooses an action that is not compatible as a coloring, it must choose a compatible coloring. Since we assumed that a encodes a partial coloring and there are always enough colors to choose from (i.e. there are $\Delta^{2t+2} + 1$ colors), v can always choose an action that gives it utility 1.

After one fair round, each agent v has chosen a strategy such that the colorings r_v agree on all applicable nodes, and this is a Nash equilibrium of G'. The output label σ_v of each action, by definition, corresponds to the output of $\mathcal{F}$ on (N, x) given the input coloring that the agents have chosen.

Finally, it remains to show that G' can be constructed efficiently in the LOCAL model on the network N. This is achieved using a standard approach. First each node v gathers its $(4t+2)$-hop neighborhood and outputs its neighbors in N'. Since the algorithm has access to $\mathcal{F}$, the algorithm in the normal form, it can consider every coloring of $B(v, t)$ and form the action set A'_v. Finally, since the algorithm has access to the $(4t+2)$-neighborhood of v in (N, x), it can compute the value of the utility function u_v for all possible strategies of the neighbors. ◀