Institute of Science and Technology

The Complexity of Evolutionary Games on Graphs

Krishnendu Chatterjee and Rasmus Ibsen-Jensen and Martin A Nowak

Technical Report No. IST-2015-323-v1+1
Deposited at 19 Feb 2015 10:16
http://repository.ist.ac.at/323/1/main.pdf

IST Austria (Institute of Science and Technology Austria)
Am Campus 1
A-3400 Klosterneuburg, Austria

The Complexity of Evolutionary Games on Graphs*

Krishnendu Chatterjee† Rasmus Ibsen-Jensen† Martin A. Nowak‡

† IST Austria

‡ PED, Harvard University

Abstract

Evolution occurs in populations of reproducing individuals. The structure of the population affects the outcome of the evolutionary process. Evolutionary graph theory is a powerful approach to study this phenomenon. There are two graphs. The interaction graph specifies who interacts with whom in the context of evolution. The replacement graph specifies who competes with whom for reproduction. The vertices of the two graphs are the same, and each vertex corresponds to an individual of the population. A key quantity is the fixation probability of a new mutant. It is defined as the probability that a newly introduced mutant (on a single vertex) generates a lineage of offspring which eventually takes over the entire population of resident individuals. The basic computational questions are as follows: (i) the qualitative question asks whether the fixation probability is positive; and (ii) the quantitative approximation question asks for an approximation of the fixation probability. Our main results are: (1) We show that the qualitative question is NP-complete and the quantitative approximation question is #P-hard in the special case when the interaction and the replacement graphs coincide and even with the restriction that the resident individuals do not reproduce (which corresponds to an invading population taking over an empty structure). (2) We show that in general the qualitative question is PSPACE-complete and the quantitative approximation question is PSPACE-hard and can be solved in exponential time.

Keywords: *Evolution; Evolutionary games on graphs; Fixation probability; Computational complexity.*

*The full version is attached as appendix.

1 Introduction

In this work we study the basic computational questions for evolutionary games on graphs, and present complexity results for them. We start with a description of the model of evolution on graphs and its significance. We then state the basic computational questions and present our results.

Evolutionary dynamics with constant selection. Evolutionary dynamics act on populations. The composition of the population changes over time under the influence of mutation and selection. Mutation generates new types and selection changes the relative abundance of different types. A fundamental concept in evolutionary dynamics is the fixation probability of a new mutant [10, 16, 21, 22]: Consider a population of N *resident* individuals, each with a non-negative fitness value, r. A single *mutant* with fitness value 1 is introduced in the population as the initialization step[1]. Then the following step is repeated. At each time step, one individual is chosen proportional to the fitness to reproduce and one individual is chosen uniformly at random for death. The offspring of the reproduced individual replaces the dead individual. This so-called Moran process continues until either all individuals are mutants or all individuals are residents. The *fixation probability* is the probability that the mutants take over the population, which means all individuals are mutants. A standard calculation shows that the fixation probability is given by $(1-r)/(1-r^N)$. The correlation between the relative fitness of the mutant (with respect to resident fitness, i.e., $1/r$) and the fixation probability is a measure of the effect of natural selection in that population structure [26, 19, 31]. A neutral mutant, $r = 1$, has fixation probability $1/N$. The rate of evolution, which is the rate at which subsequent mutations accumulate in the population, is proportional to the fixation probability, the mutation rate, and the population size N. Hence fixation probability is a fundamental concept in evolution.

Evolutionary game dynamics. The fitness values of individual types (resident and mutant) need not be constant, but could themselves depend on the composition of the population. This idea brings us to evolutionary game theory, where the individuals of a population interact with each other to receive a payoff. There could be two strategies, R and M, and a payoff matrix

$$\begin{array}{c} \\ R \\ M \end{array} \begin{array}{c} \begin{array}{cc} R & M \end{array} \\ \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \end{array} \tag{1}$$

The payoff of an individual is the average payoff of the interactions (see [22, Section 7.1]; also Section 2). Standard evolutionary game theory assumes a well-mixed population structure, which means all individuals interact with equal probability. Again a fundamental question is the fixation probability of a mutant [10, 16, 21, 22], which quantifies whether or not a mutant is favored by natural selection.

Evolutionary graph theory: Informal model. The outcome of an evolutionary process is dependent on population structure. Evolutionary graph theory studies this phenomenon. The individuals of the population occupy the vertices of a graph. The links (edges) determine who interacts with whom. Evolutionary graph theory describes evolutionary dynamics in spatially structured population where most interactions and competitions occur mainly among neighbors in physical space [24, 19, 27, 8, 12]. Another application is cultural evolution (spread of ideas and behaviors) in social networks [14]. Finally, the hierarchy of cellular proliferation and differentiation in the human body, which is crucial for physiological function and for reducing cancer initiation, is described by evolutionary graph theory [25]. The evolutionary graph theory considers directed graphs because interactions between individuals need not be symmetric [25, 19]: Examples of population structures and evolutionary processes that resemble directed graphs include somatic evolution of cancer either in epithelial tissue [25, 23] or in the hematopoeitic system [20], the spatial distribution of microbial and other populations along flowing water gradients or social networks, where some people are more influential than others [1].

Evolutionary graph theory: Basic results. For the case of constant fitness (which means residents with relative fitness r and mutants with relative fitness 1) graphs have been identified that maintain the same selection pressure as the well mixed population, that amplify selection, or that reduce selection. For example, a star graph is an amplifier of selection, because the fixation probability of the mutant is given by $\frac{1-r^2}{1-r^{2N}}$; hence the star graph squares the relative fitness [19]. In contrast, 'isothermal graphs' where the in-degree and out-degree of all vertices coincide (such as

[1] In the literature, an alternative notion is to consider that the mutant have fitness r and the residents have fitness 1, we follow the notation that leads to simpler formulas

regular undirected graphs) have the same fixation probability as the Moran process, $\frac{1-r}{1-r^N}$ [19, 4]. There are some graphs and update rules that enhance the evolution of cooperation, which is a particular strategy in evolutionary games, for example in the well-known Prisoner's dilemma [19, 27]. Evolution of cooperation is a major topic in evolutionary biology, because cooperation is seen as a main component for the creative tendency of evolution. A crucial aspect of evolutionary graph theory is the computation of the fixation probability of an invading mutant.

The formal model and computational questions. In the study of evolutionary games on graphs in general there are two graphs (that have the same vertices) [19, 28]. The "interaction graph" specifies who interacts with whom for payoff. The "replacement graph" specifies who competes with whom for reproduction. The initial step is the introduction of a mutant uniformly at random and then at each step a vertex is chosen proportional to the fitness (or payoff). If the fraction of successors in the interaction graph that are of the same type as the chosen vertex is below a threshold (i.e., a density constraint is satisfied), then the individual in the vertex reproduces to a successor uniformly at random among the successors in the replacement graph. The density constraint, which is relevant in many applications of evolution (see books [3, page 470] [29, page 320]), can also be *encoded* in the payoff matrix (see Remark 10). The relevant computational questions for evolution on graphs are as follows: (1) the *qualitative* question asks whether the fixation probability is positive; and (2) the *quantitative approximation* question asks, given $\epsilon > 0$, to compute an approximation of the fixation probability within an additive error of ϵ.

Special cases of the model. While in the general model the interaction and replacement graphs are different (we refer to the model as the I&R model), an important special case is where these two graphs coincide (we refer to the model as the IEQR model) [19, 27]. Another important special case is when the residents cannot reproduce, and this corresponds to the case where a mutant arises in an empty geographic location, and the question is whether the mutant can spread (hence the residents, being non-existing, cannot reproduce). Note that if the residents cannot reproduce in the standard Moran process (without density constraints), i.e., $r = 0$, the fixation probability is 1.

Our contributions. While previous results characterized the fixation probabilities of specific graphs (such as star or regular undirected graphs), the complexity of computing the fixation probability for arbitrary input graphs has been open (explicitly referred to as an important open problem in a survey [30, Open Problem 2.1 and 2.2]). We study the computational complexity of the basic questions for evolution on graphs and our results are as follows:

1. We show that under no resident reproduction, the qualitative decision question is NP-complete both for the I&R and IEQR models.
2. We show that under no resident reproduction, the quantitative approximation problem is #P-hard even for the IEQR model, where $\epsilon > 0$ is part of the input and specified in binary. Our result implies the #P-hardness of the quantitative approximation in all models.
3. We show that with resident reproduction, the qualitative question is PSPACE-complete for the general I&R model. Finally, for the general I&R model with resident reproduction, we show that the quantitative approximation question is PSPACE-hard (for all constants $0 < \epsilon < 1$) and can be solved in polynomial space with double exponentially small error probability (which we refer to as *RPS*), and the exact fixation probability can be computed in exponential time.

Our results are summarized in Table 1 and our main contributions are the lower bounds. Moreover, note that we will present the lower bounds for constant selection with the density constraints which is a special case of evolutionary games on graphs (as argued in Remark 10). We will present the relevant aspects of the lower bounds, and the upper bounds and other technical details are in the full version.

Related complexity result. To the best of our knowledge, previous to our results, there was only one computational complexity result for evolutionary games on graphs. For the precise computation of the fixation probability, NP-hardness for evolutionary games on graphs (named as frequency dependent selection) in the IEQR model was stated in [19]. Our result presents much stronger lower bounds: we show NP-hardness even for the qualitative problem and #P-hardness even for approximation. The problem of computing evolutionary stable strategies without the population structure (the underlying graphs) but for any number of strategy types has been considered in [9], whereas evolutionary games on graphs consider two strategy types (resident and mutants) but the evolutionary dynamics operates on a population structure. The problem of time scale (or speed) of evolutionary proceeses has also been studied in different contexts [5, 33], which are related to mixing time of Markov chains; however, none of these works consider evolutionary games on graphs or complexity results. While the problem of stochastic processes on automata (probabilistic

automata) have been studied in depth [2, 6, 11], the computational study of the stochastic process on graphs induced by evolutionary games has largely been left open [30, Open Problem 2.1 and 2.2].

Technical contributions. The complexity study of evolutionary games on graphs brings together many diverse fields of studies related to logic (namely, game theory, graph theory, evolutionary stochastic processes, and computational complexity): it involves the study of stochastic processes which arise in the context of evolution, and requires the analysis of stochastic processes in combination with graph theory. Our main results are computational complexity results for the analysis of the fundamental evolutionary stochastic processes on graphs, and our main technical contribution is to develop novel gadgets on graphs that in combination with the evolutionary stochastic processes can mimic runs of a polynomial-space Turing machine (for the PSPACE lower bounds), or has the ability to count the number of matchings in bipartite graphs (for the #P-hardness).

	No Resident Reproduction		Resident Reproduction	
	IEQR model	I&R model	IEQR model	I&R model
Qual.	NP-c ((**LB**) Lem. 2)	NP-c ((**UB**) Lem. 1)	NP-h, PSPACE	PSPACE-c ((**LB**) Lem. 7, (**UB**) Lem. 5)
Appr.	#P-hard, RPS ((**LB**) Thm. 4)	#P-hard, RPS	#P-hard, RPS	PSPACE-h, RPS ((**LB**) Lem. 8, (**UB**) Lem. 5)

Table 1: Complexity of evolution on graphs. Qual is short-hand for qualitative and appr for approximation. Our main contributions of lower bounds (LB) and upper bounds (UB) are boldfaced. NP-c (resp., PSPACE-c) means NP-complete (resp., PSPACE-complete). Similarly, NP-h (resp., PSPACE-h) means NP-hard (resp., PSPACE-hard). RPS indicates that the problem can be solved in polynomial space, with randomization and double exponentially small error probability.

2 Models of Evolution on Graphs

In this section we present the basic definitions related to the different models of evolution on graphs and the basic computational questions.

Evolutionary graphs. An *evolutionary graph* $G = (V, E_I, E_R)$ consists of a finite set V of vertices; a set $E_I \subseteq V \times V$ of *interaction* edges; and a set $E_R \subseteq V \times V$ of *replacement* (or reproduction) edges [28]. The sets E_I and E_R consist of directed edges, and the graph $G_I = (V, E_I)$ is called the interaction graph, and $G_R = (V, E_R)$ is called the replacement graph. The graph G_I is responsible for determining the interaction of individuals in the graph (which affects the fitness or payoff), and the graph G_R captures the underlying structure for reproduction and replacement of individuals in the graph. Given an edge (v, u) we say u is a *successor* of v and v is a *predecessor* of u.

Fitness of individuals. Each vertex of the graph will be occupied by one of two types of individuals, namely, the *resident* type and the *mutant* type. In evolutionary games, along with the evolutionary graph there is a payoff matrix as defined in Equation (1) (Section 1), where the entries of the matrix are rational numbers and represent the payoff of an interaction, i.e., a_{11} (resp., a_{12}) is the payoff of a resident type interacting with another resident (resp., mutant) type, and a_{21} (resp., a_{22}) is the payoff of a mutant type interacting with a resident (resp., mutant) type. Given two types, x and y, we denote by $\mathsf{pay}(x, y)$ the payoff of type x versus type y. The fitness of an individual at a vertex v is a non-negative number and determined as follows: Let $E_I(v) = \{u \mid (v, u) \in E_I\}$ denote the set of interaction successors of v, then the fitness of v, denoted as $f(v)$, is the average payoff of the interactions but at least 0, i.e., $f(v) = \max\{\frac{\sum_{u \in E_I(v)} \mathsf{pay}(v,u)}{|E_I(v)|}, 0\}$. A special case of the payoff matrix is the *constant fitness* (aka constant selection) matrix defined as follows:

$$\begin{array}{c} \\ R \\ M \end{array} \begin{array}{c} \begin{array}{cc} R & M \end{array} \\ \begin{pmatrix} r & r \\ 1 & 1 \end{pmatrix} \end{array}$$

i.e., the mutant types always have fitness 1 and the resident types fitness r, where $r \geq 0$. Intuitively, the fitness of an individual represents the reproductive strength.

Threshold for density constraints. Along with the evolutionary graph and the payoff matrix, we have two thresholds, namely, θ_R and θ_M, for the resident type and the mutant type, respectively. Intuitively, the thresholds represent a *density constraint*, and if an individual is surrounded by a lot of individuals of the same type, then its reproductive strength decreases. The density constraint, which is relevant in many applications of evolution (see books [3, page 470] [29, page 320]), can also be encoded in the payoff matrix (see Remark 10).

The evolutionary process. The evolutionary process we consider is the classical *birth-death* process on an evolutionary graph defined as follows:

1. Initially all vertices of the graph are of the resident type and a mutant type is introduced uniformly at random at one of the vertices of the graph and then the following step (referred to as a *generation*) is repeated.

2. In every generation, a vertex is selected proportional to the fitness of the individual at the vertex to reproduce. Let the selected vertex for reproduction be v. Let $\mathsf{Same}(v)$ denote the number of vertices in $E_I(v)$ that are of the same type as v. If v is a mutant type, and $\frac{\mathsf{Same}(v)}{|E_I(v)|} \leq \theta_M$ (resp., if v is a resident type, and $\frac{\mathsf{Same}(v)}{|E_I(v)|} \leq \theta_R$), then the individual gives birth to an individual of the same type. The new born individual replaces one of the replacement successors of v, i.e., it replaces a vertex chosen uniformly at random from the set $E_R(v) = \{u \mid (v,u) \in E_R\}$. Note that the density constraint implies that if the constraint is violated, then the selected individual does not reproduce.

Step 2 (or generations) is repeated until nothing can change (in particular, if all vertices have fitness 0 or have the same type, then nothing can change).

Fixation probability. The most relevant question from an evolutionary perspective is the *fixation probability* which is the probability that the mutant takes over the population, i.e., eventually all vertices become the mutant type.

Computational questions. Given an evolutionary graph, a payoff matrix, and the thresholds for density constraints, we consider the following questions:

1. the *qualitative* decision question asks whether the fixation probability is positive; and

2. the *quantitative approximation* question, given $\epsilon > 0$, asks to compute an approximation of the fixation probability within an additive error of ϵ.

Special cases. There are several special cases of interest that we will explore.

1. **The** I&R **and** IEQR **models.** One important special case is when the interaction and the replacement graphs coincide, i.e., $E_I = E_R$ [19, 27]. We refer to the general model as the I&R *model* (with possibly different interaction and replacement graphs) and the special case where the graphs coincide as the IEQR *model*.

2. **No resident reproduction.** Another special case is when the payoff matrix is the constant payoff matrix with $r = 0$. In this case, the resident types cannot reproduce. This represents the scenario that a mutant invades an empty geographic location.

3 Qualitative Analysis for No Resident Reproduction

In this section we establish two results for the no resident reproduction model: the qualitative analysis problem is (1) in NP for the general I&R model; and (2) is NP-hard even in the special case of IEQR model.

3.1 Upper bound

The upper bound is relatively straightforward. We simply check if there exists an initial choice v_1 for the initial mutant and a sequence $(e_i)_{2 \leq i \leq n}$ of edges of length $n-1$ in the replacement graph for reproductions that ensures that all vertices are mutants. The initial vertex v_1 and the sequence of edges together define a unique sequence of vertices for reproduction; and at every stage we check that for the vertex chosen for reproduction the density constraint is satisfied and it is a mutant. We also need to check that in the end all vertices are mutants. The choice of the initial vertex and

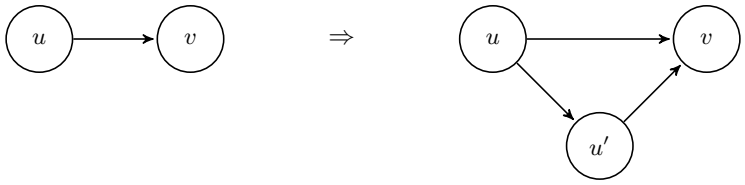

Figure 1: Illustration of a predecessor gadget (u, v).

the sequence of reproductions then happen with positive probability and we are done. Observe that since there is no resident reproduction, if a vertex becomes a mutant, then it remains a mutant. Note that there always exists a sequence of length $n-1$, because if the fixation probability is positive, then we can WLOG assume (till all vertices are mutants) that in each step i there is a vertex v that is a mutant, with a fraction of mutant neighbors in the interaction graph below the threshold θ_M, and an edge (v, v') in the replacement graph such that v' is not a mutant (and becomes a mutant in step i), as otherwise nothing can change. This shows that if the answer to the qualitative decision question is yes in the no resident reproduction model, then there is a polynomial witness and polynomial-time verification procedure.

Lemma 1. *The qualitative decision question for no resident reproduction in the general* I&R *model is in NP.*

3.2 Lower bound

In this section we present an NP lower bound, and we will prove it for the IEQR model with no resident reproduction. Moreover, since there is no resident reproduction, the threshold θ_R does not matter. We will present a reduction from the 3-SAT problem (which is NP-complete [7, 18, 13]) and use threshold θ_M as $\frac{1}{2} - \delta$, for any $0 < \delta \leq \frac{1}{10}$. However it would be easy to modify our construction for any threshold θ_M in $(0, 1)$. The "right" way to think of the threshold is that it is $\frac{1}{2}$ and that the density constraint uses a strict inequality (the exact range of feasibility on δ comes from that each vertex in our lower bound has degree 5 or less). The upper bound is chosen because we will use vertices with degree five or less.

Notation. Let $X = \{x_1, x_2, \ldots, x_n\}$ be a set of n Boolean variables. Consider a 3-CNF formula $\varphi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$, where each C_i is a *clause* of a list of (precisely) three *literals* (where a literal is a variable x or its negation $\overline{x}$, where $x \in X$). Each clause represents a disjunction of the literals that appear in it. An instance of the 3-SAT problem, given a 3-CNF formula φ, asks whether there exists a satisfying assignment. We will now construct an evolutionary graph $G(\varphi)$, given an instance of a 3-SAT problem, with (i) $E_I = E_R$, (ii) no resident reproduction, and (iii) threshold $\theta_M = \frac{1}{2} - \delta$, for $0 < \delta \leq \frac{1}{10}$ such that there is a satisfying assignment iff the answer to the qualitative decision problem is YES. We first present two gadget constructions that will be used in the reduction.

Predecessor gadget. We present a *predecessor* gadget for a vertex pair (u, v) such that v is the only successor of u. The gadget ensures the following property (namely, the *predecessor gadget* property): if all vertices become mutants, then the vertex u must have become a mutant before vertex v. The construction of the gadget is as follows: Add a new *dummy* vertex u'. Let the successors of u be v and u', and the successor of u' be only v. Then the only way for u' to become a mutant is if u is a mutant, since u is the only predecessor of u'. But u' can only become a mutant if u is a mutant and v is not (since otherwise the threshold condition with $\theta_M = \frac{1}{2} - \delta$ is not satisfied for u, for any $0 < \delta \leq \frac{1}{10}$). Hence, if all vertices become mutant, then u must become a mutant before v. There is an illustration of the predecessor gadget for (u, v) in Figure 1. We will denote by $\mathsf{PredEdges}(u, v, u')$ the set $\{(u, v), (u, u'), (u', v)\}$ of edges of the predecessor gadget.

(Extended) Binary tree gadget. Given a vertex rt, and a set L of vertices, we will denote by $\mathsf{BinTr}(\mathsf{rt}, L)$ a binary tree with rt as root and L as leaf vertices[2]. In a binary tree, every non-leaf vertex has out-degree 2. Note that the binary tree gadget adds additional vertices, and has $O(|L|)$ vertices. By an abuse of notation we will use $\mathsf{BinTr}(\mathsf{rt}, L)$ to denote both the set of vertices and the set of edges of the binary tree, and it would either be clear from the context or explicitly mentioned. Given a binary tree T and an *extension* vertex $z \notin T$, an *extended binary tree* (EBT) consists of T and an edge from every non-leaf vertex to z. Given a root vertex rt, a set L of leaf vertices, and an extension vertex

[2]For a fixed L and rt there exists many possible binary trees $\mathsf{BinTr}(\mathsf{rt}, L)$, however every one of them will work for our purpose

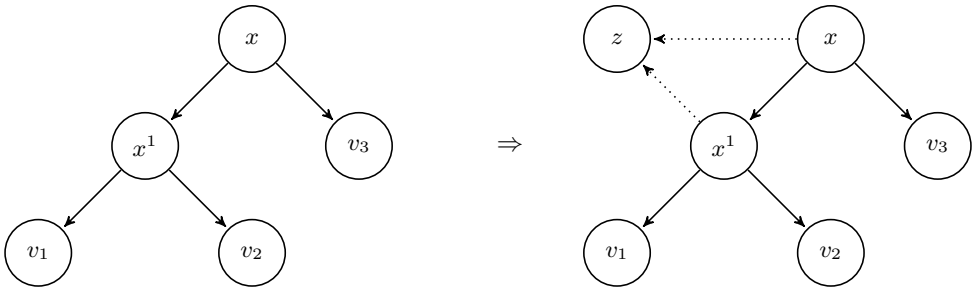

Figure 2: A binary tree $\mathsf{BinTr}(x, \{v_1, v_2, v_3\})$ and the corresponding EBT $\mathsf{ExBinTr}(x, \{v_1, v_2, v_3\}, z)$, where we extend with the vertex z. The edges to z are dotted to make the similarities easier to see.

z, we denote by $\mathsf{ExBinTr}(\mathsf{rt}, L, z)$ the edge set of the extended binary tree that extends the binary tree of rt and L. We will explicitly use the following property for an EBT (namely, *qualitative EBT (QEBT)* property):

- *(QEBT Property).* In an EBT, every non-leaf vertex has out-degree 3, and for density constraint with threshold $\frac{1}{2} - \delta$, for $0 < \delta \leq \frac{1}{10}$ (the construction works even if δ is up to $\frac{1}{6}$), if the root becomes a mutant and z is not a mutant, then the root can be responsible for making every vertex in the tree a mutant. However, note that if z is a mutant, then any vertex in the tree with out-degree 3 cannot make both its children in the underlying tree mutants due to the density constraint.

There is an illustration of a binary tree $\mathsf{BinTr}(x, \{v_1, v_2, v_3\})$ and the corresponding EBT $\mathsf{ExBinTr}(x, \{v_1, v_2, v_3\}, z)$ in Figure 2.

The evolutionary graph $G(\varphi)$. We now present the evolutionary graph $G(\varphi)$, see Figure 3 for an illustration, where we first describe the vertex set and then the edges. Recall that n is the number of variables and m the number of clauses of the 3-SAT instance ϕ.

The vertex set. The set V of vertices is as follows (intuitive descriptions follow):

$$\begin{aligned} \{v_\top, z_\bot, y_\bot, z'_\bot\} \quad &\cup \quad \{c_1, c_2, \ldots, c_m\} \qquad \cup \{c_i^1, c_i^2, c_i^3 \mid 1 \leq i \leq m\} \quad \cup \quad \{\widehat{c}_i^1, \widehat{c}_i^2, \widehat{c}_i^3 \mid 1 \leq i \leq m\} \\ &\cup \quad \{x_i, x_i^t, x_i^f \mid x_i \in X\} \quad \cup \{v_0, v'_0\} \\ &\cup \quad \{u_i^t, u_i^f \mid 1 \leq i \leq n\} \quad \cup \bigcup_{1 \leq i \leq n} (\mathsf{BinTr}(x_i^t, L_i^t) \cup \mathsf{BinTr}(x_i^f, L_i^f)) \end{aligned}$$

The vertex $v_\top$ will be the start vertex; and the vertices $z_\bot$, $y_\bot$, and $z'_\bot$ are end vertices (that will form a predecessor gadget for $(z_\bot, y_\bot)$ with dummy vertex $z'_\bot$). We have a vertex c_i for each clause C_i (named the clause vertices); and one for each literal c_i^1, c_i^2, and c_i^3 in the clause (named the clause-literal vertices). Similarly, we have a vertex x_i for each variable in X (named the variable vertices), and vertices x_i^t and x_i^f (named the variable-value vertices) to represent the truth values to be assigned to x_i. Corresponding to x_i^t and x_i^f we also have vertices u_i^t and u_i^f (named the duplicate vertices). The vertex v_0 forms a predecessor gadget (using the dummy vertex v'_0) to u_1^t. Let $L_i^t = \{\widehat{c}_k^j \mid 1 \leq k \leq m,\ 1 \leq j \leq 3,\ c_k^j = x_i\}$ denote a copy of the clause-literal vertices that correspond to x_i and $L_i^f = \{\widehat{c}_k^j \mid 1 \leq k \leq m,\ 1 \leq j \leq 3,\ c_k^j = \overline{x}_i\}$ denote a copy of the clause-literal vertices that correspond to negation of x_i. The set $\mathsf{BinTr}(x_i^t, L_i^t)$ (resp., $\mathsf{BinTr}(x_i^f, L_i^f)$) represents the vertices of a binary tree with the root vertex x_i^t (resp., x_i^f) and leaf vertices L_i^t (resp., L_i^f).

The edge set. We now describe the edge set:

- There is an edge from the initial vertex $v_\top$ to the first clause vertex c_1; and we have two predecessor gadgets; (i) $(z_\bot, y_\bot)$ with dummy vertex $z'_\bot$; and (ii) (v_0, u_1^t) with dummy vertex v'_0.

- For each clause vertex c_i, there are five edges, three to clause-literal vertices c_i^j (for $j = 1, 2, 3$) of the clause, one to the next clause vertex (for c_m this next vertex is x_1), and to the vertex u_1^t.

- For each variable vertex x_i, there are three edges: to x_i^t and x_i^f, and to the next variable vertex x_{i+1} (for x_n the next vertex is v_0).
- Each duplicate vertex u_i^t has three edges: to u_i^f, to x_i^t, and to $y_\perp$. Similarly, each vertex u_i^f has three edges: to u_{i+1}^t (u_n^f has edge to $z_\perp$ instead), to x_i^f, and to $y_\perp$.
- Finally, we have the EBT with x_i^α (for $\alpha \in \{t, f\}$) as root, L_i^α as leaf vertices and $y_\perp$ as the extension vertex. For each vertex in L_i^α, for $\alpha \in \{t, f\}$, we add edges to the corresponding clause-literal vertex and to u_1^t. This ensures that every internal vertex of the binary tree has degree three, and leaf vertices have degree two.

The formal description is as follows:

$$\{(v_\top, c_1)\} \ \cup \ \mathsf{PredEdges}(z_\perp, y_\perp, z'_\perp) \ \cup \ \mathsf{PredEdges}(v_0, u_1^t, v'_0)$$

$$\{(c_i, c_i^j) \mid 1 \le i \le m, 1 \le j \le 3\} \ \cup \ \{(c_i, u_1^t) \mid 1 \le i \le m\} \ \cup \ \{(c_i, c_{i+1}) \mid 1 \le i \le m-1\} \cup \{(c_m, x_1)\} \ \cup$$

$$\{(x_i, x_i^t), (x_i, x_i^f) \mid 1 \le i \le n\} \ \cup \ \{(x_i, x_{i+1}) \mid 1 \le i \le n-1\} \cup \{(x_n, v_0)\} \ \cup$$

$$\{(u_i^t, u_i^f) \mid 1 \le i \le n\} \ \cup \ \{(u_i^f, u_{i+1}^t) \mid 1 \le i \le n-1\} \ \cup \ \{(u_n^f, z_\perp)\} \ \cup \ \{(u_i^\alpha, x_i^\alpha), (u_i^\alpha, y_\perp) \mid 1 \le i \le n, \alpha \in \{t, f\}\} \cup$$

$$(\bigcup_{1 \le i \le n} (\mathsf{ExBinTr}(x_i^t, L_i^t, y_\perp) \cup \mathsf{ExBinTr}(x_i^f, L_i^f, y_\perp)) \ \cup \ \{(\widehat{c}_k^j, c_k^j), (\widehat{c}_k^j, u_1^t) \mid \widehat{c}_k^j \in L_i^\alpha, 1 \le i \le n, \alpha \in \{t, f\}\}$$

Example. We will now give an example of the graph $G(\varphi)$ for $\varphi = (\bar{x} \vee y \vee x) \wedge (z \vee x \vee \bar{x})$. See Figure 3. The edges to u_1^t are dashed and the edge from u_i^α for all $1 \le i \le 3$ and $\alpha \in \{t, f\}$ are dotted, for readability. Also, the vertex u_1^t is included twice to make it clearer that it is in a predecessor gadget.

Basic facts. We first mention some basic facts about the evolutionary graph obtained.

1. First, observe that the predecessor gadget property implies that for fixation the vertex v_0 must become a mutant before vertex u_1^t; and vertex $z_\perp$ before vertex $y_\perp$.
2. Second, for a vertex with degree ℓ, it can reproduce a mutant as long as at most $\ell \cdot (\frac{1}{2} - \delta)$ successors are mutants. In particular, for vertices with five (resp., three) successors, like the clause (resp., variable) vertices, it can reproduce a mutant until at most three (resp., two) successors are mutants, because of the bounds on θ_M. If a vertex has out-degree two (or one), then it can reproduce a mutant until at most one successor is a mutant, because of the bounds on θ_M. The conditions follow from the density constraint with threshold $\frac{1}{2} - \delta$.

Two phases for fixation. For mutants to attain fixation (i.e., all vertices become mutants), certain conditions must be fulfilled. The first basic fact above implies that for the evolutionary process to attain fixation, it must make vertex x_n a mutant (then vertex v_0 a mutant) before vertex u_1^t. We thus split the process of fixation in two phases: in the first phase u_1^t is not a mutant, and in the second phase u_1^t will be a mutant. We further split the first phase into two sub-phases, the first sub-phase is related to clause vertices becoming mutants, and the next sub-phase is related to the variable vertices becoming mutants. The description of the phases for fixation are as follows:

1. *(Phase 1:Part A).* The mutant must be initialized at the start vertex $v_\top$ (since $v_\top$ has no predecessor). After $v_\top$, the clause vertex c_1 becomes a mutant. Since at most half (three) successors can become mutant from c_1 (recall that c_1 has five successors), and one of them must be c_2 (as the only incoming edge for c_2 is from c_1), it follows that c_2 and at most two clause-literal vertices for clause C_1 becomes mutant from c_1. This process is then repeated for all the clause vertices c_i till x_1 becomes a mutant.
2. *(Phase 1:Part B).* Each of the vertices x_i has three successors, and hence can make two of them mutants. One of them must be x_{i+1} (as x_{i+1} has only x_i as the predecessor), and the other one is at most one of x_i^t or x_i^f. This continues till we reach v_0. Note that once x_i^t becomes a mutant, then the entire EBT under x_i^t, including the corresponding clause-literal vertices, but not $y_\perp$ and u_1^t, can become mutants, as long as $y_\perp$ and u_1^t are not mutants. The reasoning is as follows: the leaf vertices has two out-going edges, and since u_1^t is not a mutant, it can reproduce a mutant to the corresponding clause-literal vertices, and the rest follows from the QEBT property.

The phase 1 ends with the predecessor gadget of (v_0, u_1^t) becoming mutants. Note that this phase corresponds to a partial assignment of truth values to the variables as follows: for a variable x_i, if x_i^t was chosen (made mutant), it corresponds to assigning true to x_i; if x_i^f was chosen, it corresponds to assigning false to x_i; otherwise, if neither was chosen, then it corresponds to no assignment to x_i (if fixation is reached without having made an assignment to some set U of variables, then any possible assignment of values to the variables of U will make the partial assignment a satisfying assignment).

3. *(Phase 2).* This phase starts after u_1^t is a mutant. We establish a key property of this phase that will be used in the proof. Consider the EBT under some variable-value vertex. Each leaf vertex of the tree has out-degree two: one of the successors is u_1^t and the other is a clause-literal vertex. It follows that once u_1^t has become a mutant, then the leaf vertices cannot reproduce any more. Thus the key property of Phase 2 is as follows: leaf vertices of EBTs cannot reproduce mutants to clause-literal vertices after Phase 2 starts.

The graph $G(\varphi)$ has positive fixation probability iff φ is satisfiable. We present two directions of the proof.

1. *Satisfiablity implies positive fixation.* Consider a satisfying assignment to φ, and intuitively the assignment chooses at least one literal in each clause. The sequence of mutants reproduced in the two phases for fixation is as follows:

   - (Phase 1). The sequence in Phase 1 is the following: (1) initial vertex $v_\top$ becomes a mutant which then reproduces a mutant to c_1; (2) in vertex c_i, it reproduces upto three mutants, one to c_{i+1} (to x_1 for $i = m$) and upto two mutants for vertices c_i^j of the clauses which are not chosen by the satisfying assignment (this corresponds to Phase 1:Part A); (3) for a vertex x_i it reproduces two mutants, one to x_{i+1} (to v_0 for $i = n$), and the other to x_i^t (resp., x_i^f) if the assignment chooses x_i to be true (resp., false); and moreover, the entire EBT under x_i^t (resp., x_i^f) including the clause-literal vertices become mutants (other than u_t^1 and $y_\perp$); and (4) then v_0' becomes a mutant and then u_1^t becomes a mutant from v_0, and proceed to Phase 2.
   - (Phase 2). The sequence in Phase 2 is the following: (1) In every vertex u_i^α (for $\alpha = t$ or f) it makes x_i^α mutant (if it is not already a mutant) and then it makes the next vertex in line a mutant (if $i = n$ and $\alpha = f$, then the next vertex is $z_\perp$, otherwise, the next vertex is u_i^f if $\alpha = t$ and u_{i+1}^t if $\alpha = f$); moreover, once x_i^α becomes a mutant, so does the entire binary tree (other than $y_\perp$) under it (but not the clause-literal vertices since u_1^t is a mutant); and (2) finally the $(z_\perp, y_\perp)$ predecessor gadget becomes mutants.

   The claim follows.

2. *No satisfying assignment implies no fixation.* Note that for fixation we need the two phases. In every clause c_i at least one of the clause-literal vertices c_i^j was not made a mutant by c_i in Phase 1:Part A (or even after that). This implies that if Phase 2 has started and not all clause-literal vertices c_i^j of a clause c_i have become mutants, then at least one of these vertices cannot become a mutant, by the key property of Phase 2. For each (partial) assignment that is not satisfying, there exists at least one clause, in which no literals are chosen. Recall that the reproduction of mutants in Phase 1:Part B gives a partial assignment of truth values to variables. Hence, in the process of reproducing mutants in Phase 1:Part B, there must remain a clause where at most two clause-literal vertices are mutants. Therefore it implies that if there is no satisfying assignment, then fixation is not possible.

We obtain the following result. Lemma 1 and Lemma 2 give Theorem 3.

Lemma 2. *The qualitative decision question for no resident reproduction in the* IEQR *model is NP-hard.*

Theorem 3. *The qualitative decision question for no resident reproduction in both the general* I&R *model and the* IEQR *model is NP-complete.*

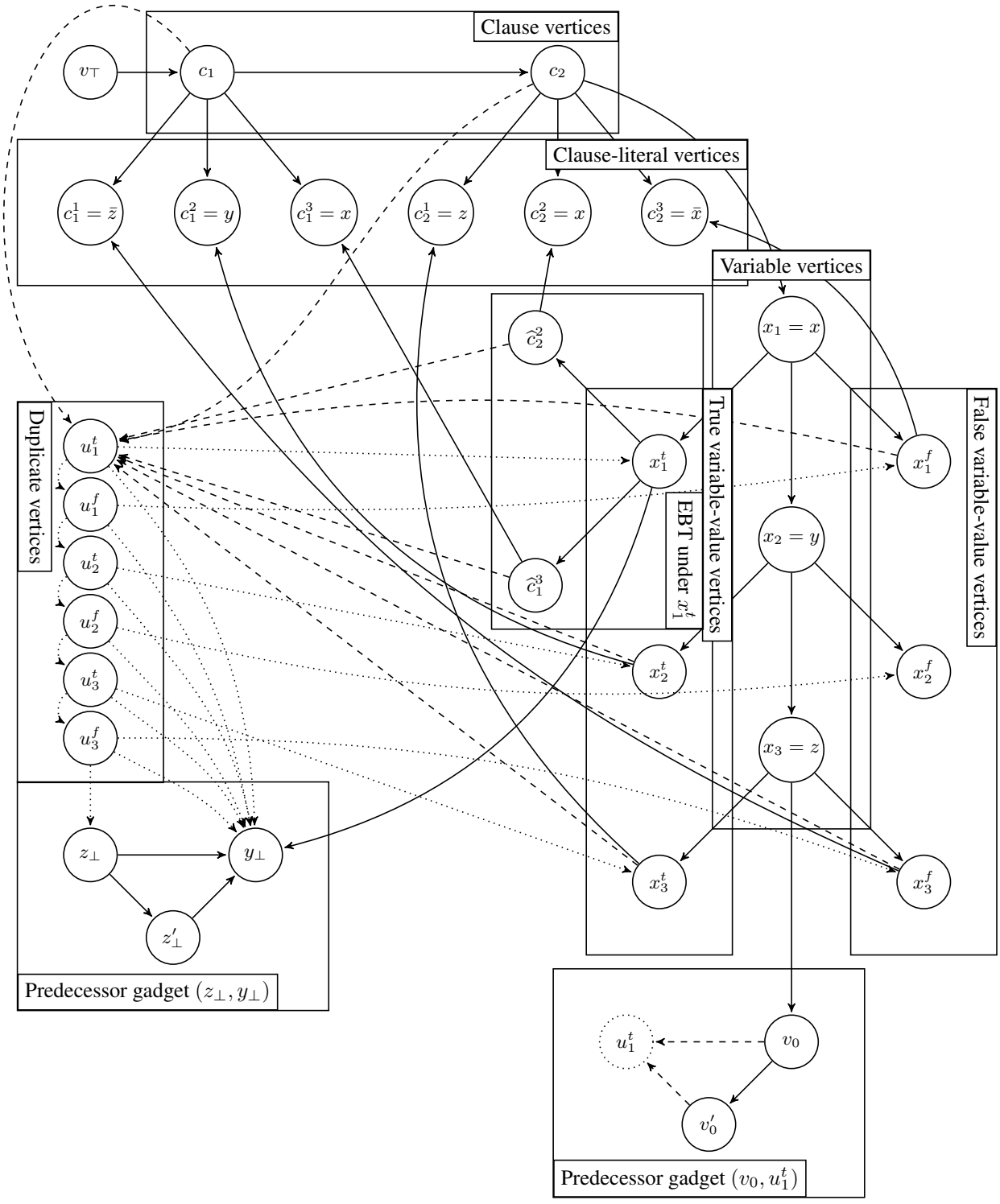

Figure 3: The graph $G(\varphi)$ for $\varphi = (\bar{z} \vee y \vee x) \wedge (z \vee x \vee \bar{x})$. Edges to u_1^t are dashed and edges from u_i^α are dotted for readability. The vertex u_1^t is included twice to make it clearer that it is in a predecessor gadget. The notation $c_1^2 = y$ indicates that the second variable of the first clause is variable y. The notation $x_1 = x$ indicates that the first variable is variable x.

4 Approximation in the IEQR Model with No Resident Reproduction

In this section we show that for $\epsilon > 0$ the problem of approximating the fixation probability within $\pm\epsilon$ is #P-hard, in the IEQR model with no resident reproduction. Again the threshold θ_M will be $\frac{1}{2} - \delta$, for any $0 < \delta \leq \frac{1}{10}$ (because the degree is again bounded by 5).

Perfect matching in bipartite graphs. We present a reduction from the computation of the number of perfect matchings in a bipartite graph $G = (V, E)$. In a bipartite graph G, the vertex set V is partitioned into vertices V_ℓ (left vertices) and V_r (right vertices) and all edges go from a vertex in V_ℓ to a vertex in V_r (i.e., $E \subseteq V_\ell \times V_r$). We also have $|V_\ell| = |V_r| = n$. A *perfect matching* PM is a set $\{e_1, e_2, \ldots, e_n\}$ of n edges from E such that for every vertex $v_\ell \in V_\ell$ (resp., $v_r \in V_r$) there exists an edge $e_\ell = (v_\ell, v'_r)$ (resp., $e_r = (v'_\ell, v_r)$) in PM. Given a bipartite graph, the problem of computing the number of distinct perfect matchings was shown by Valiant [32] to be #P-complete.

Uniform degree property. First, we will show that we only need to consider bipartite graphs for which there exists an integer k such that all vertices in V_ℓ have either degree 2^k or 1. We refer to the property as the *uniform degree* property.

Reduction to uniform-degree graphs. We present a reduction from counting the number of perfect matchings in a general bipartite graph $G = (V, E)$ (with $|V_\ell| = |V_r| = n$) to counting the number of perfect matchings in a bipartite graph $G' = (V', E')$ with at most $6n$ vertices and which has the uniform degree property. Let $k = \lceil \log d_{\max} \rceil$, where $d_{\max}$ is the maximum degree of any vertex in G. The graph G' will have precisely as many perfect matchings as G. Observe that $2^k < 2n$. We construct G' by adding 2^k new pairs of vertices, one on each side, and for each new pair (v, v'), we add an edge from v to v'. Then, for vertex $v \in V_\ell$, we add edges from v to some newly added vertex in V'_r until v has degree 2^k. It is clear that any perfect matching in G corresponds to a perfect matching in G' using the same edges, and the edges between newly added pairs. Conversely, we also see that in each perfect matching in G', for each newly added pair (v, v'), the matching must use the edge between v and v', since the vertex in $(V'_\ell \setminus V_\ell)$ has degree 1. Thus every perfect matching in G' corresponds to one in G.

Perfect binary trees. We will consider perfect binary trees as gadgets.

- A *perfect binary tree* (PBT) is a balanced binary tree (every internal vertex has exactly two children) with all leaves at the same level (i.e. with 2^k leaf vertices, for some non-negative integer k). For a PBT we will use the following property, which we refer to as the *probabilistic PBT (PPBT)* property: if the root becomes a mutant, then eventually all vertices in a path from the root to some leaf will become mutants, where such a path is chosen uniformly at random. Since every non-leaf vertex has out-degree two, due to the density constraint, each internal vertex can make one of its children (chosen uniformly) a mutant and hence the PPBT property follows.

The graph Red(G)**.** Given a bipartite graph G with the uniform degree property, let the vertex sets be V_ℓ and V_r, respectively. Let $N(v) = \{u \mid (v, u) \in E\}$ denote the successors of a vertex $v \in V_\ell$. Let $V_\ell^k = \{v \in V_\ell \mid |N(v)| = 2^k\}$ be the set of vertices with degree 2^k; and $V_\ell^1 = V_\ell \setminus V_\ell^k$ be the set of vertices in V_ℓ with degree 1. Our reduction, denoted Red(G), will construct an evolutionary graph (with $E_I = E_R$ and hence we only specify one set of edges), which consists of three parts: part 1 sub-graph, then edges related to V_r, and a copy of part 1 with some additional edges. We first describe the part 1 sub-graph and then its copy.

- *(Part 1).* We have a start vertex v_s, a final vertex $y_\perp$, and we create an EBT B_s as follows: $\mathsf{ExBinTr}(v_s, V_\ell, y_\perp)$, i.e., the start vertex is the root, V_ℓ is the set of leaf vertices, and $y_\perp$ is the extension vertex. For every vertex $v \in V_\ell^k$, let $N(v) = \{u^1, u^2, \ldots, u^j\}$, and we consider the set $L_v^k = \{u_v^1, u_v^2, \ldots, u_v^j\}$ of $j = 2^k$ vertices and construct a PBT $P_v = \mathsf{BinTr}(v, L_v^k)$. Note that B_s is an EBT, but the underlying binary tree is not necessarily perfect.

- *(Edges related to V_r).* From every vertex $v \in V_\ell^k$, and every u_v^i in L_v^k, we add two edges: one to $u^i \in N(v)$ and one to $y_\perp$. From every vertex $v \in V_\ell^1$ (with degree 1), we add two edges: to the unique $u \in N(v)$ and to $y_\perp$. Every vertex in V_r has an edge to $y_\perp$.

- *(Copy 1 of Part 1 with additional edges).* First, we create a copy of the part of the graph described in part 1, along with one additional vertex $z_\perp$. For every vertex v of part 1, let the corresponding vertex in the copy be

called $\overline{v}$, and the copy of the extension vertex is $\overline{y}_\perp$. We describe the difference in the copy as compared to the graph of part 1: (i) first there is an edge from $y_\perp$ to the copy $\overline{v}_s$ of the start vertex; (ii) for every vertex $\overline{z}$ which is a copy of a non-leaf vertex z in P_v, for some $v \in V_\ell^k$, (i.e., $z \notin L_v^k$), there are three additional edges from $\overline{z}$: (a) to z (i.e., from the copy to the original vertex), (b) to $\overline{y}_\perp$, and (c) to $z_\perp$; and (iii) for every vertex $\overline{z}$ which is a copy of a leaf vertex z in P_v, for some $v \in V_\ell^k$, (i.e., $z \in L_v^k$), there is only one edge which goes to z (i.e., there is no edge to V_r or $y_\perp$, but an edge from the copy to the original vertex). Hence in the copy of P_v, for any v, internal vertices have degree five, and leaf vertices have degree 1.

- Finally, we have the following edges: $\{(y_\perp, \overline{y}_\perp), (y_\perp, z_\perp), (\overline{y}_\perp, z_\perp)\}$.

We denote by $\widehat{n}$ the number of vertices in Red(G), and note that $\widehat{n} = O(m)$, where m is the number of edges in G.

Example. We consider the graph G with six vertices, where $V_\ell = \{v_1, v_2, v_3\}$ and $V_r = \{v_4, v_5, v_6\}$, such that v_1 and v_2 each have edges to v_4 and v_5 and v_3 has an edge to v_6. See Figure 4 for an illustration. Observe that G satisfies the uniform degree property. In Figure 5 we have part 1 of the graph Red(G) along with V_r. In Figure 6 we have the remainder of Red(G). Consider some fixed perfect matching PM in G, i.e. $v_1 \to v_4$ and $v_2 \to v_5$ and $v_3 \to v_6$. The graph Red$(G)^{\mathsf{PM}}$ is then the same graph as in Figure 5 and Figure 6, except that in Figure 5 it does not contain the edges from v_2^1 or v_1^2.

The process of fixation in Red(G)**.** The process of fixation in Red(G) can be decomposed in two phases. The first phase (Phase 1) is over when $y_\perp$ becomes a mutant; and the second phase (Phase 2) is over with the fixation. A key property of Phase 2 is as follows: vertices in V_r cannot become a mutant after $y_\perp$ has become a mutant: This is because for each vertex u in V_r, every predecessor v of u has exactly two successors, and one them is $y_\perp$ (and hence the density constraint with threshold $\frac{1}{2} - \delta$ ensures that if $y_\perp$ is a mutant, then vertices in V_r cannot become mutants after that).

- *Phase 1.* In Phase 1, the vertex v_s must be the first vertex to become a mutant (since it has no predecessor). After v_s, all vertices in B_s turn into mutants (by the QEBT property). Once a vertex $v \in V_\ell^k$ becomes a mutant, then a path in the PBT P_v under v is chosen uniformly at random to become mutants (by the PPBT property), and then the leaf of the path can make the corresponding vertex in V_r a mutant. Once a vertex v in V_ℓ^1 with degree 1 becomes a mutant, then it can reproduce a mutant to the unique neighbor in V_r. In the end, some vertex in V_r reproduces a mutant to $y_\perp$ and Phase 1 ends.

- In Phase 2, first the copy $\overline{v}_s$ becomes a mutant from $y_\perp$. After $\overline{v}_s$, all vertices which are copy of vertices in B_s become mutants (again by the QEBT property). Once copies of vertices in V_ℓ^k become mutant, then the tree underneath them in the copy become mutants. Consider a vertex $\overline{u}$ which is a copy of a vertex $u \in P_v$, for some $v \in V_\ell^k$, and there are two cases: (i) if u is a non-leaf vertex, then $\overline{u}$ has degree five, and can reproduce mutants until the two children in the tree and the original vertex u are mutants (note if $\overline{y}_\perp$ or $\overline{z}_\perp$ is a mutant, then both the children and the original copy cannot all become mutants due to the density constraint); (ii) if u is a leaf-vertex, then $\overline{u}$ has degree one, and can reproduce mutant for u. Finally, $y_\perp$ makes $\overline{y}_\perp$ a mutant, which then makes $z_\perp$ a mutant.

Fixation and a perfect matching. Observe that fixation implies that all vertices in V_r have become mutant, and no vertex in V_r can become a mutant in the second phase. Each vertex in V_ℓ is responsible for making at most one neighbor in V_r a mutant (for vertices with degree 1 it is the unique successor in V_r, and for vertices with degree 2^k, it corresponds to the leaf of the path in the perfect binary tree chosen uniformly at random by the PPBT property). This defines a perfect matching. Conversely, given a perfect matching, Phase 1 and Phase 2 of fixation can be described using the pairs of the matching (to chose paths uniformly at random in the perfect binary trees). Thus given fixation, it defines a perfect matching, and we say that fixation has *used* the perfect matching.

Exact fixation probability. Consider some perfect matching PM. Observe that if there are $s > 0$ perfect matchings, then the exact fixation probability is $s \cdot x_{\mathsf{PM}}$, where x_{PM} is the probability that we have fixation and used PM. This is because each perfect matching has the same probability to be the chosen matching in Phase 1 by the PPBT property. In Phase 2, any vertex v which is either a vertex in V_ℓ^1 or a leaf in P_v, for $v \in V_\ell^k$, cannot reproduce by the key property of Phase 2 (and thus can be viewed as having no out-going edges). Thus in Phase 2, by symmetry, the probability x_{PM} of fixation for a perfect matching PM is independent of PM.

Bounds on x and s. We show that the probability x for fixation of a fixed matching is at least $\eta = \widehat{n}^{-2\widehat{n}}$, where $\widehat{n}$ is the number of vertices in Red(G). Each possible way that all vertices can become mutants happens with probability at least $\widehat{n}^{-2\widehat{n}}$, because there are at most $\widehat{n}$ reproductions (effective reproductions which produce a new mutant) and each specific reproduction chooses two vertices v and v' at random from some set of vertices and thus, a specific choice happens with probability at least $\widehat{n}^{-2}$. Thus the lower bound η on x follows. Finally, observe that the number s of perfect matchings can be at most $n!$ (i.e., upper bound on s is $n!$).

The graph Red$(G)^{\mathsf{PM}}$**.** Given a perfect matching PM, we can find x as the fixation probability for the graph Red$(G)^{\mathsf{PM}}$, which is similar to Red(G), except that each leaf vertex u_v^i in P_v, for $v \in V_\ell^k$, if (v, u^i) is not in the matching, then we remove all out-edges from u_v^i, and otherwise u_v^i has the same edges as in Red(G). It is clear that the fixation probability in Red$(G)^{\mathsf{PM}}$ is x.

Approximating the fixation probability is $\#\mathsf{P}$-hard. Our reduction is as follows: Given a graph G with the uniform degree property, we want to find the number of perfect matchings s in it. First, (i) we find an arbitrary perfect matching PM in polynomial time using the algorithm of [15] (if there exists none, we are done); (ii) construct Red(G) and Red$(G)^{\mathsf{PM}}$ in polynomial time; and (iii) compute the approximation y' of the fixation probability y^* in Red(G) for $\epsilon = \frac{\eta}{16}$, and the approximation x' of the fixation probability x in Red$(G)^{\mathsf{PM}}$ for $\epsilon_{\mathsf{PM}} = \frac{\eta}{n! \cdot 16} = \frac{\epsilon}{n!}$. We now show how to obtain s from y' and x'. We have that y' is such that

$$y' \leq x \cdot s + \epsilon \leq (x' + \epsilon_{\mathsf{PM}}) \cdot s + \epsilon = x' \cdot s + \frac{\eta}{n! \cdot 16} \cdot s + \frac{\eta}{16} \leq x' \cdot s + \frac{\eta}{8} \ ,$$

and similarly $y' \geq x' \cdot s - \frac{\eta}{8}$. This shows that

$$s - \frac{\eta}{8x'} \leq \frac{y'}{x'} \leq s + \frac{\eta}{8x'}.$$

Since we also have $x' \geq x - \epsilon = \eta - \frac{\eta}{n! \cdot 16} \geq \frac{15 \cdot \eta}{16}$ we see that $\frac{\eta}{8x'} < 1/3$ and thus s is the integer closest to $\frac{y'}{x'}$.

Theorem 4. *The quantitative approximation problem for $0 < \epsilon < 1$, with ϵ given in binary, for no resident reproduction in both the general* I&R *model and the* IEQR *model is $\#\mathsf{P}$-hard.*

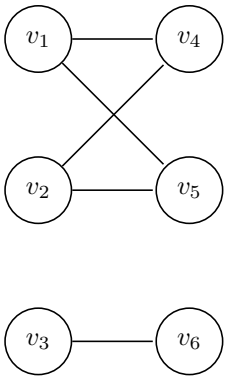

Figure 4: The graph G.

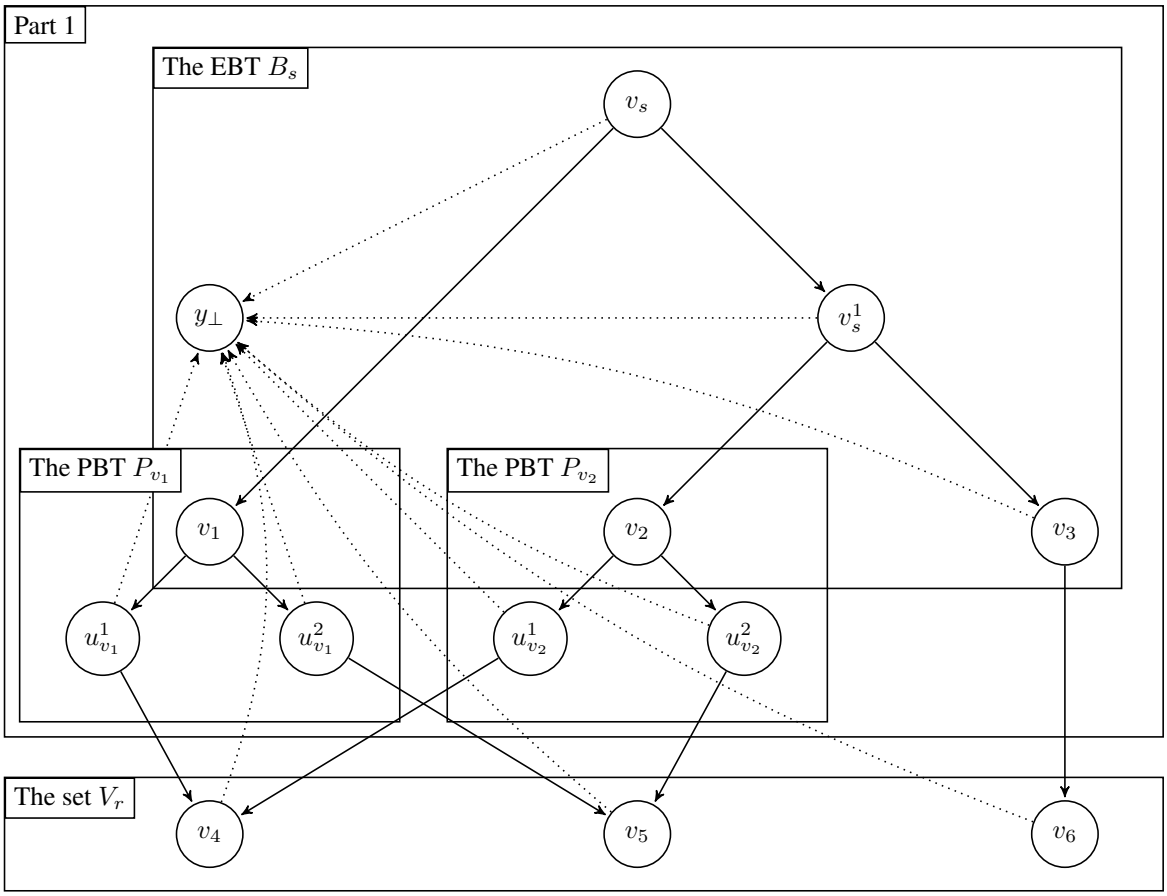

Figure 5: Part 1 and the edges related to V_r of the graph Red(G). The edges to $y_\perp$ are dotted for readability.

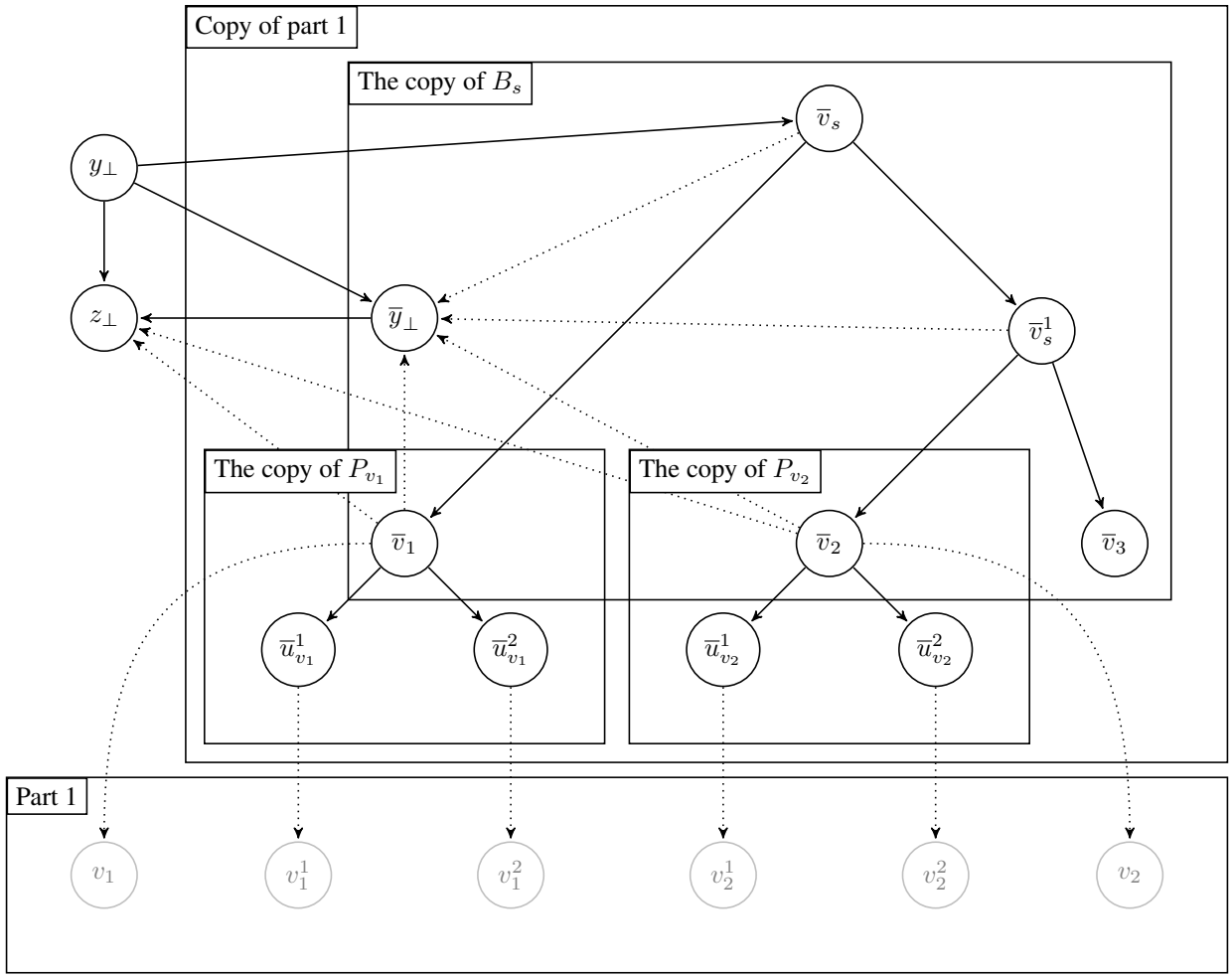

Figure 6: The copy of part 1 of the graph Red(G). Most edges to $\overline{y}_\perp$ and to $z_\perp$ are dotted for readability.

5 PSPACE-Completeness for the I&R Model with Resident Reproduction

In this section we will establish the polynomial space upper bound and lower bound in the I&R model with resident reproduction.

5.1 Upper bound

In regards to the approximation problem, we only provide a randomized algorithm with double exponentially small error probability. We first describe what is a Markov chain and Markov chains associated with an evolutionary graph.

Markov chain. A Markov chain $M = (S, \Delta)$ consists of a finite set S of states, and a probabilistic transition function Δ that assigns transition probabilities $\Delta(s, s')$ for all $s, s' \in S$, i.e., $0 \leq \Delta(s, s') \leq 1$ for all $s, s' \in S$ and for all $s \in S$ we have $\sum_{s' \in S} \Delta(s, s') = 1$. Given a Markov chain, its graph (S, E) consists of the set S as the set of vertices, and $E = \{(s, s') \mid \Delta(s, s') > 0\}$ positive transition probabilities as the set of edges.

Exponential Markov chain. Given an evolutionary graph $G = (V, E_I, E_R)$, with a payoff matrix, and the density

constraints, an exponential Markov chain $M_E = (S, \Delta)$ is constructed as follows: (1) S consists of subsets of V which denotes the set of vertices of V which are currently mutants; (2) for $s \in S$ and $s' \in S$ there is positive transition probability if the cardinality of s and s' differ by 1 and the transition probability $\Delta(s, s')$ is computed depending on the payoff matrix, E_I, E_R, and the density constraints. Observe that for the Markov chain M_E, the transition probabilities of a state in the Markov chain can be constructed in polynomial space, and hence the Markov chain can be constructed in polynomial (working) space.

Qualitative analysis and approximation of Markov chains. We sketch the arguments for the upper bounds.

- The qualitative analysis is achieved by simply checking if in the graph of the Markov chain the state $s_f = V$ is reachable from some state $s = \{v\}$ for $v \in V$. It follows that the qualitative question is in PSPACE.
- For the approximation problem we simulate the Markov chain as follows. We start at an initial state uniformly at random among those where there is exactly one mutant. Consider a *trial run* of the Markov chain as follows. Given the current state, we first check if (i) the current state is V; else we check (ii) if there is a path from the current state to $s_f = V$. If (i) is true we have a success; and if (ii) is false we have a failure. If we neither succeed or fail, we use the transition probability of the Markov chain to obtain the next state till we succeed or fail. Note that each trial run succeeds or fails eventually with probability 1. We can view the outcome of each trial run as the outcome of a Bernoulli distributed random variable with success probability equal to the fixation probability. Hence repeating the trial runs independently an exponential number of times, we can approximate the fixation probability using Chernoff bounds, within any given $\epsilon > 0$, with double-exponential small error probability.

Lemma 5. *The qualitative decision problem in the general* I&R *model is in PSPACE. The quantitative approximation problem can be solved for the general* I&R *model in polynomial space with double exponentially small error probability.*

Remark 6. *Observe that since precise probabilities to reach a state in a Markov chain can be computed in polynomial time in the size of the Markov chain [17], it follows that the precise fixation probabilities can be computed in exponential time.*

5.2 Lower bound

We show two lower bounds: (i) the qualitative decision question is PSPACE-hard; and (ii) the question that given an evolutionary graph with the promise that the fixation probability is either 0 or close to 1, deciding which is the case is PSPACE-hard. We will present a reduction from the membership problem of a deterministic polynomial space Turing machine (which is PSPACE-hard by definition) to an evolutionary graph (with separate E_I and E_R) and a constant fitness matrix (but $r > 0$, and hence residents can reproduce). Given an instance of a Turing machine A with binary input I of length n, whether A accepts I using space at most $P(n)$, where P is a polynomial, we present the reduction in two stages. First we will present a reduction such that if a specific vertex is the first to become a mutant, then the fixation probability is precisely 1 if A accepts input I using at most $P(n)$ space, otherwise it is 0. Thus since all vertices are chosen uniformly at random for the initial mutant, there is a fixation probability of at least $\frac{1}{N}$ if the machine accepts (where N is the number of vertices in the evolutionary graph). This already shows the hardness for the qualitative problem. Later we show how to amplify the fixation probability to show the hardness for approximation.

Density constraint. Our construction will be for $\theta_R = \theta_M = 0$, but a similar construction will work for any choice of $\theta_R, \theta_M \in [0, 1)$. The thresholds $\theta_R = \theta_M = 0$ indicates that a vertex v can reproduce precisely as long as all its successors in E_I are of the opposite type of v, because of the density constraint.

Ideas behind the reduction. We will use the following ideas in our construction:

1. *Changing Turing machine:* We will first reduce the problem to a similar Turing machine A', which has states $A'(S) = \{0, 1, 2\} \times A(S)$, where $A(S)$ is the set of states of A. For each transition t from state s to state s' in A there are three corresponding transitions in A', one for each $i \in \{0, 1, 2\}$, which updates the tape content and the position of the tape-head in the same way as t, but goes from state $(i \mod 3, s)$ to state $(i + 1 \mod 3, s')$ in A'. This reduction given two successive configurations of the Turing machine allows to detect which is the former and which the later.

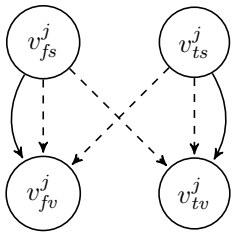

Figure 7: Boolean-value gadget: Dashed edges are in E_I and non-dashed are in E_R.

2. *States which are nearly always a mutant/resident:* Similar to the previous lower bounds, we have a vertex v_s without any predecessor in E_R. Thus, if v_s is not made a mutant at the start, then it cannot become a mutant. Hence we will only consider the case when v_s is a mutant in the beginning and stays a mutant forever. We will also have a vertex $\widehat{v}_s$, and our construction will ensure that it stays a resident until all other vertices are mutants and then (after a few more steps) all vertices become mutants, and we get fixation. We will use the vertices v_s and $\widehat{v}_s$ to ensure that a given vertex has a desired type, and otherwise the vertex cannot reproduce. Our construction will ensure (using the density constraint) the following properties:

   - A vertex v with $\widehat{v}_s$ as a successor under E_I can only reproduce if it is a mutant (using the density constraint and $\widehat{v}_s$ is a resident). Similarly, a vertex v with v_s as a successor under E_I can only reproduce if it is a resident.

3. *Boolean-value gadgets:* The vertices of the graph will encode values of tape cells of the Turing machine, along with the states and tape head positions of the Turing machine. We describe how to implement a *boolean-value* gadget in the evolutionary graph, which can be checked and set to a boolean value using a single external resident predecessor. In effect the value setting is done at random, but the resident predecessor will keep on setting the value of the boolean-value gadget either to true or false and eventually, with probability 1 it will be set to the right value. We first describe the construction of the gadget, then its requirement, and finally present the principle with which the gadget works (given the requirement is fulfilled).

   - *(Construction):* Each boolean-value gadget j consist of four vertices: Two *value vertices*, namely, (i) v^j_{tv} (the *true-value-vertex*) which is a mutant if the boolean value is true; and (ii) v^j_{fv} (the *false-value-vertex*) which is a mutant if the boolean value is false. If neither of the value vertices are mutants, we interpret that the gadget has *no value*. Besides the value vertices, there are two vertices (called *the setter vertices*) which will be required to be mutants: (i) v^j_{ts} (the *true-setter-vertex*) and (ii) v^j_{fs} (the *false-setter-vertex*). The edge set is as follows: (i) both v^j_{ts} and v^j_{fs} have $\widehat{v}_s, v^j_{tv}, v^j_{fv}$ as successors under E_I; (ii) v^j_{ts} (resp., v^j_{fs}) has only v^j_{tv} (resp., v^j_{fv}) as a successor under E_R (see Figure 7). The purpose of the edges in E_I are as follows: the edge to $\widehat{v}_s$ enforces that the setter vertex is a mutant before reproduction; and the other two edges enforce that only if the gadget has no value (i.e., both value vertices are resident), then the setter vertex can reproduce a mutant (by the density constraint and that $\theta_R = \theta_M = 0$).
   - *(Requirement):* The only requirement for the gadget to work is that both the setter vertices are mutants and the setter vertices have no other successors (except for the ones specified above).
   - *(Principle):* The main principle of the gadget is as follows: the two setter vertices become mutants (in an initial phase and remain mutants, to ensure the requirement). In a *consistent phase*, exactly one of the two value vertices will be a mutant and the other cannot become a mutant due to density constraints. The consistent phase represents a boolean value. To change the boolean value the procedure is as follows: an external resident vertex can check using E_I that the Turing machine should change the value of the gadget, say from false to true. Then the external resident vertex reproduces a resident to the false-value-vertex v^j_{fv}. At this point, the boolean gadget has no value (i.e., not in a consistent phase), and both the setter vertices can reproduce mutants (especially, there is a positive probability to reproduce a mutant from v^j_{ts} to v^j_{tv}). If a mutant is reproduced to v^j_{fv}, then the external vertex can keep reproducing residents to v^j_{fv}, which ensures

that eventually with probability 1 the vertex v_{tv}^j is a mutant and the boolean gadget restores to a consistent phase (with a true value). The process of setting a true value to false is similar by reproducing residents to v_{tv}^j.

The construction of the graph $G(A, I, P)$. We now start the description of the construction of the evolutionary graph given an instance of the membership problem of a polynomial-space Turing machine. We start with the boolean-value gadgets for the Turing machine.

Turing machine boolean-value gadgets. For each i in $\{-1, 0, 1, \ldots, P(n)+1\}$ there are $(2 \cdot |A'(S)|+1)$ boolean-value gadgets. One boolean-value gadget i corresponds to the content of the tape at position i. For each state s in $A'(S)$, each position i, and each content c (where $c = 0$ or 1), there is a boolean-value gadget (i, s, c) for the tape-head being in position i, in state s, and the tape content at position i being c. We say that the Turing machine is in a *super-position* if more than one boolean-gadget (i, s, c) is true. The value vertices in most of these boolean-value gadgets have no successors in either E_I or E_R. We now describe the exceptions. For each accepting state s in $A'(S)$ and for each i in $\{0, 1, \ldots, P(n)\}$, consider the value vertices $v_{tv}^{(i,s,1)}$ and $v_{tv}^{(i,s,0)}$ of the boolean-value gadgets $(i, s, 1)$ and $(i, s, 0)$. The value vertices have one successor $\widehat{v}_s$ in E_I and a special vertex $v_\top$ in E_R. The key idea is as follows: If the machine accepts, then one of these value vertices become a mutant, and then they will make $v_\top$ mutant. The edge to $\widehat{v}_s$ under E_I ensures that the vertex reproduces only if it is a mutant. Our construction will ensure that once $v_\top$ is a mutant, then fixation follows given v_s is already a mutant.

The three stages. We will split the construction with the following stages in mind. (1) *The initialization stage* consists of two parts. First, for each Turing machine boolean-value gadget, the initial values are set in two steps. A gadget with initial value false (resp., true) is *partly initialized* if the false-setter-vertex (resp., true-setter-vertex) becomes a mutant which then reproduces a mutant to the false-value-vertex (resp., true-value-vertex) of the gadget. In the first stage of the initialization, all Turing machine boolean-value gadgets are partly initialized (using another boolean-value gadget named b_1); and then checked that the partial initialization is achieved (using a check vertex called c_1). In the second stage, all the remaining setter vertices in the Turing machine boolean-value gadgets become mutants, and each gadget gets to a consistent phase. The second stage of initialization and checking are achieved (similar to the first stage) with boolean-value gadget b_2 and check vertex c_2, respectively. The boolean-value gadget b_1 is only set to true after the boolean-value gadgets of the Turing machines are partly initialized. Similarly, the boolean-value gadget b_2 is only set to true after all the boolean-value gadgets for the Turing machine have finished initialization (i.e., are in a consistent state). (2) *The execution stage* which corresponds to the execution of the Turing machine on the input. (3) *The post-acceptance stage* which corresponds to the steps after acceptance of the Turing machine to ensure fixation.

1. *Preprocessing step:* Before the initialization phase we describe two vertices and their roles.

   (a) *Start vertex:* The vertex v_s has no predecessor in E_R (hence must be the first vertex to become a mutant) and no successors in E_I. The vertex has two successors in E_R, namely, vertex v_1 and the false-setter-vertex $v_{fs}^{b_1}$ for b_1. This ensures that after v_s both $v_{fs}^{b_1}$ (to partly initialize b_1) and v_1 can become mutants (note that since v_s has no successor in E_I it can always reproduce mutants).

   (b) *Last resident:* The vertex v_1 has no successors in either set (i.e., no out-going edges in E_I or E_R). If there is fixation, then this vertex will become a mutant in the beginning (from v_s), then will become a resident close to the end of fixation (only after acceptance); and then finally become a mutant again as the last vertex.

2. *Initialization state, part 1:* We first partly initialize the boolean-value gadgets of the Turing machine and it is achieved as follows: first the partial initialization is done (by $v_{fv}^{b_1}$), and then it is checked (by an additional vertex c_1).

   (a) *First part of initialization:* The false-value-vertex $v_{fv}^{b_1}$ for b_1 has $\widehat{v}_s$ as the only successor in E_I (to enforce that the vertex is a mutant before reproduction). The successors under E_R are as follows:

   - The true-setter-vertex $v_{ts}^{b_1}$. This allows $v_{fv}^{b_1}$ to finish the initialization of b_1.

- For each boolean-value gadget of the Turing machine, $v_{fv}^{b_1}$ has either the true-setter-vertex or the false-setter-vertex as successor depending on the initial value for the gadget being true or false, respectively. This allows $v_{fv}^{b_1}$ to partly initialize the boolean-value gadgets of the Turing machine.

(b) *Check if first part of initialization is done:* We have a check vertex c_1. The check vertex c_1 has out-going edges in E_I consisting of the following:

- Edges to $v_s, v_1, v_{fv}^{b_1}, v_{fs}^{b_1}, v_{ts}^{b_1}$. The purpose of the edge to v_s is to ensure that c_1 is itself a resident before reproduction. The rest of the edges enforce that all these vertices are mutants before c_1 reproduces residents, to ensure that v_1 is a mutant, and the boolean-value gadget b_1 has value false.
- For $z \in \{t, f\}$, an edge to each setter-vertex v_{zs}^{b} and the corresponding value vertex v_{zv}^{b} in the Turing machine boolean-value gadgets where the setter vertex is a successor of $v_{fv}^{b_1}$ under E_R. These edges enforce that these gadgets are partly initialized before c_1 reproduces residents.

The successor of c_1 in E_R is $v_{fv}^{b_1}$ to set the boolean-value gadget b_1 to true. Thus c_1 is an external resident vertex to set the boolean-value gadget b_1 to true. Note that b_1 is only set to true after all the boolean-value gadgets of the Turing machine are partly initialized.

(c) *Go to part two of initialization:* The true-value-vertex $v_{tv}^{b_1}$ for b_1 has the false-setter-vertex $v_{fs}^{b_2}$ for b_2 as a successor in E_R, and $\widehat{v}_s$ as successor in E_I. The edge in E_I enforces that $v_{tv}^{b_1}$ is itself a mutant, if it can reproduce. The edge in E_R ensures that the boolean-value gadget b_2 can become partly initialized. Also, we will later use $v_{tv}^{b_1}$ to check that the first part of the initialization is over (by checking that it is a mutant).

3. *Initialization stage, part 2:* The second phase of initialization begins when the check vertex c_1 has set the boolean-value gadget b_1 to true. In this phase, the initialization of the Turing machine (which was partly done in the first part) is completed and checked. The procedure is similar to the first part and the details are as follows.

(a) *Second part of initialization:* The false-value-vertex $v_{fv}^{b_2}$ for b_2 has $\widehat{v}_s$ as the only successor in E_I (to enforce that the vertex is a mutant before reproduction). The successors in E_R consists of the following:

- The true-setter-vertex $v_{ts}^{b_2}$ for b_2. This allows $v_{fv}^{b_2}$ to finish the initialization of b_2.
- Each setter-vertex in a boolean-value gadget in the Turing machine which is not a successor for $v_{fv}^{b_1}$ under E_R. This allows $v_{fv}^{b_2}$ to finish the initialization of the boolean value gadgets in the Turing machine.

(b) *Check if second part of the initialization is done:* The successors of check vertex c_2 in E_I are

- The vertices $v_s, v_{tv}^{b_1} v_{fv}^{b_2}, v_{fs}^{b_2}$, and $v_{fs}^{b_2}$. The purpose of the edge to v_s is to enforce that if c_2 can reproduce, then it is itself a resident. The edge to $v_{tv}^{b_1}$ enforces that the first part of initialization is over, and the remaining edges enforce that b_2 has been initialized to false, before c_2 reproduces residents.
- The successors of $v_{fv}^{b_2}$ under E_R. The purpose of these edges is to ensure that the boolean-value gadgets of the Turing machine have been initialized before c_2 reproduces residents.

The only successor of c_2 under E_R is $v_{fv}^{b_2}$. Thus c_2 is an external resident vertex to b_2, to set the value of b_2 to true. Again note that the value of b_2 is set to true, only after the boolean-value gadgets of the Turing machine have been initialized. Another important point is that after the initialization, since the setter vertices of the boolean-value gadgets are all mutants, the requirement for all such gadgets are fulfilled.

(c) *Initialization is done:* The true-value-vertex $v_{tv}^{b_2}$ for b_2 has no successors in E_I or E_R and is used to check that the second part of initialization is done (by checking that it is a mutant).

4. *The execution stage:* For each $i \in \{0, 1, \ldots, P(n)\}$, each state s of $A'(S)$, and each possible content $c \in \{0, 1\}$ of the tape at position i there are five check vertices, namely, $c_1^{(i,s,c)}$, $c_2^{(i,s,c,0)}$, $c_2^{(i,s,c,1)}$, $c_3^{(i,s,c,0)}$, and $c_3^{(i,s,c,1)}$. Let the content of the tape at position i just after having been in state s be b (and the content before was c) and the complement value of b be $\overline{b}$. Let the next position of the tape head be i' and the state s', given that the Turing machine is in state s, the tape head is at position i, and the tape-content is c (this is defined by description of the Turing machine).

Intuitively, we will split each step of the execution into three parts. First, (1) we update the content of the tape at position i (if needed); (2) then we set the next configuration (i.e. the boolean-value gadget (i', s', c'), where c' is the content of the tape at position i') of the Turing machine to true; and (3) then at the end we set the current configuration to false (i.e. the boolean-value gadget (i, s, c)). Each check vertex associated with part j has subscript j, for $j \in \{1, 2, 3\}$. We have one check vertex for the first part and two for each of the others, because we do not know a priori the content c' of the tape at position i'. Note that we enter a super-position after part 2, but by construction of A' we can still distinguish (i, s, c) from (i', s', c'). Note that for $i = -1$ and $P(n) + 1$ we do not have check vertices, ensuring that if the Turing machine head enters one of these positions, then the machine does not accept, and in $G(A, I, P)$ the evolutionary process stops without fixation.

(a) *Updating the tape:* First we will describe the successors of the check vertex $c_1^{(i,s,c)}$. The set of successors for $c_1^{(i,s,c)}$ in E_I consists of

- The vertices $v_s, v_{tv}^{b_1}, v_{tv}^{b_2}$. Theses edges enforce that the vertex $c_1^{(i,s,c)}$ is a resident, and the initialization is over, before $c_1^{(i,s,c)}$ reproduces.
- The true-value-vertex $v_{tv}^{(i,s,c)}$ for the tape-head being in position i, in state s, with the tape content at position i being c. In other words, this enforces that the Turing machine is in that position/state/has that content, before $c_1^{(i,s,c)}$ reproduces residents.
- For each i'', s'', c'' such that $i \neq i''$ or $s \neq s''$ or $c \neq c''$, the false-value-vertex $v_{fv}^{(i'',s'',c'')}$ for the tape-head being in position i'' and in state s'' of $A'(S)$ while the content of the tape below is c''. This enforces that the Turing machine is not in a super-position before $c_1^{(i,s,c)}$ reproduces residents.
- The $\overline{b}$-value-vertex $v_{\overline{b}v}^{i}$ for position i of the tape. This enforces that the content of the tape at position i should be updated, before $c_1^{(i,s,c)}$ reproduces residents.

The set E_R is then the $\overline{b}$-value-vertex $v_{\overline{b}v}^{i}$ for position i of the tape. Thus $c_1^{(i,s,c)}$ is an external resident vertex that changes the value of the tape to b.

(b) *Moving the tape head, part 1:* Next we will describe the successors of the check vertex $c_2^{(i,s,c,c')}$, for $c' \in \{0, 1\}$. The set of successors for $c_2^{(i,s,c,c')}$ in E_I, is similar to the vertex $c_1^{(i,s,c)}$, (except that $c_2^{(i,s,c,c')}$ has one more, and the one checking the tape has changed, and the first three items are exactly similar) and consists of

- The vertices $v_s, v_{tv}^{b_1}, v_{tv}^{b_2}$.
- The true-value-vertex $v_{tv}^{(i,s,c)}$ for the tape-head being in position i, in state s, and with content at i being c.
- For each i'', s'', c'' such that $i \neq i''$ or $s \neq s''$ or $c \neq c''$, the false-value-vertex $v_{fv}^{(i'',s'',c'')}$ for the tape-head being in position i'' and in state s'' of $A'(S)$ while the content of the tape below is c''.
- The b-value-vertex v_{bv}^{i} for position i of the tape. This enforces that the content of the tape at position i has the right value, before $c_2^{(i,s,c,c')}$ reproduces residents.
- The c'-value vertex $v_{c'v}^{i'}$ for the content of the tape at position i'. This enforces that the content of the tape at position i' (the place the head is moving to) is c', before $c_2^{(i,s,c,c')}$ reproduces residents. Observe that the check vertex $c_2^{(i,s,c,c'')}$ for $c' \neq c''$ checks for the opposite value.

The set E_R is then the false-value-vertex $v_{fv}^{(i',s',c')}$ for the tape-head being in position i', in state s', and the content of the tape being c' (the vertex $c_2^{(i,s,c,c')}$ is the external resident vertex). This puts the Turing machine in a super-position.

(c) *Moving the tape head, part 2:* Last we will describe the successors of the check vertex $c_3^{(i,s,c,c')}$, for $c' \in \{0, 1\}$. The first two items are exactly similar as the previous two cases. The set of successors for $c_3^{(i,s,c,c')}$ in E_I consists of

- The vertices $v_s, v_{tv}^{b_1}, v_{tv}^{b_2}$.
- The true-value-vertex $v_{tv}^{(i,s,c)}$ for the tape-head being in position i, in state s, and with content at i being c.
- The true-value-vertex $v_{tv}^{(i',s',c')}$ for the tape-head being in position i', in state s', and with content at i being c'. This enforces that the Turing machine is in the super-position introduced in the last step, before $c_3^{(i,s,c,c')}$ reproduces residents.
- For each i'', s'', c'' such that $(i'', s'', c'') \notin \{(i,s,c),(i',s',c')\}$ the false-value-vertex $v_{fv}^{(i'',s'',c'')}$ for the tape-head being in position i'', in state s'' of A', with the tape content c'' under it. This enforces that the Turing machine is not in any further super-position, before $c_3^{(i,s,c,c')}$ reproduces residents.

The set E_R is then the true-value-vertex $v_{tv}^{(i,s,c)}$ for the tape-head being in position i, in state s with tape content c. Thus the vertex $c_3^{(i,s,c,c')}$ is an external resident vertex that resolves the super-position by setting the boolean (i,s,c) to false. Afterwards we are not in a super-position and the Turing-machine is in position i', in state s', with content of the tape at position i' being c'. Note that the construction of A' ensures that check vertex $c_3^{(i',s',c',0)}$ and check vertex $c_3^{(i',s',c',1)}$ cannot reproduce, since we cannot get back to state s in the next step from s' in the Turing machine A' (i.e., we cannot resolve the super-position backwards).

We remark that in the execution stage, at any point there is exactly one boolean-gadget that is *active* in the sense that reproduction can change the value of the boolean-value gadget, and nothing else can change. Moreover, the active boolean-value gadget is set to the right value by reproduction in finitely many steps with probability 1.

5. *The post-acceptance stage:* We will now describe the vertices that makes fixation happen after acceptance.

   (a) *After accept:* The vertex $v_\top$ has vertex $\widehat{v}_s$ as successor in E_I and all vertices besides v_s and $\widehat{v}_s$ as successors in E_R. Once $v_\top$ is a mutant and $\widehat{v}_s$ is a resident, it ensures that eventually all vertices other than $\widehat{v}_s$ become mutants. This is because, nothing changes any of the check vertices (i.e. the vertices c_1 and c_2 and the vertices $c_1^{(i,s,c)}$, $c_2^{(i,s,c,0)}$, $c_2^{(i,s,c,1)}$, $c_3^{(i,s,c,0)}$, and $c_3^{(i,s,c,1)}$, for any i,s,c) back to residents after they have become mutants and thus eventually all those vertices become mutants. At that point no vertex can change any vertex in any boolean-value gadget in the Turing machine to residents and thus, eventually they also become mutants.

   (b) *The vertex which is nearly always a resident:* The vertex $\widehat{v}_s$ has all other vertices as successors in E_I and vertex v_1 in E_R. In other words, after the vertex $v_\top$ has made all other vertices into mutants, $\widehat{v}_s$ makes v_1 a resident.

   (c) *Changing vertex $\widehat{v}_s$ to a mutant:* The vertex $y_\top$ has $\widehat{v}_s$ and v_1 as successors in E_I and vertex $\widehat{v}_s$ as successor under E_R. The vertex $y_\top$ changes $\widehat{v}_s$ to a mutant. Note that the only predecessor of $y_\top$ in E_R is $v_\top$ and especially, it cannot become a mutant before after v_1 has become a mutant (which happens in the first part of the initialization). Thus, it can first reproduce once $\widehat{v}_s$ has made v_1 back into a resident, which first happens once all other vertices are mutants. After $\widehat{v}_s$ has become a mutant, then v_s makes v_1 a mutant. Note that v_s or $v_\top$ might make v_1 a mutant before $y_\top$ has made $\widehat{v}_s$ a mutant, but in that case, $\widehat{v}_s$ will just try again by making v_1 a resident, and eventually, $y_\top$ then makes $\widehat{v}_s$ into a mutant. Hence fixation happens with probability 1.

Illustrations. There is an illustration of the construction of $G(A,I,P)$ in Figure 8, not explicitly including the Turing machine (it is shown as just a gray box) and not including the edges (1) in E_I to and from $\widehat{v}_s$ (each of the 8 vertices in the boolean-value-gadgets, the place where "Execution" is written and the vertices $v_\perp$ and $v_\perp^2$ has one to $\widehat{v}_s$ and $\widehat{v}_s$ has one to each other vertex); and (2) in E_R from $v_\perp$ (there is one to each other vertex, besides $\widehat{v}_s$ and v_s). Also, the gray edges are used for partial initialization. The location where "Part init." is written is for partial initialization. The location where "Finish init." is written is for the remaining part of initializing the booleans in the Turing machine. The location where "Execution" is written is the active part of the Turing machine.

In Figure 9 there is an illustration of the operation of the Turing machine. Each vertex which is black with white text is a mutant and each other vertex is a resident. The Turing machine is such that it if it is in state 1 and reads a 0, the Turing machine writes 1 to the tape, moves right and goes to state 2. The illustration only contains the small part of the Turing machine needed for the move from being in position 1 in state 1 with tape content 0, when the tape content at position 2 is 0 (i.e. v^1_{fv} is a mutant). The Turing machine is in state 1 at position 1 and the content of the tape is 0 at position 1 (as seen by $v^{(1,1,0)}_{tv}$ and v^1_{fv} being mutants). This causes $c^{(1,1,0)}_1$ to reproduce (it is the only vertex that can) and makes v^1_{fv} into a resident and then both v^1_{ts} and v^1_{fs} can reproduce. If v^1_{fs} reproduces we repeat (i.e. $c^{(1,1,0)}_1$ reproduces). Eventually v^1_{ts} reproduces and afterwards, the vertex $c^{(1,1,0,0)}_2$ can reproduce and makes $v^{(2,2,0)}_{fv}$ into a resident (this lets $v^{(2,2,0)}_{fs}$ and $v^{(2,2,0)}_{ts}$ reproduce like before - again, if $v^{(2,2,0)}_{fs}$ reproduces, then so does $c^{(1,1,0,0)}_2$ repeatedly). Eventually, $v^{(2,2,0)}_{ts}$ reproduces, which lets $c^{(1,1,0,0)}_3$ reproduce and changes $v^{(1,1,0)}_{tv}$ to resident. That lets $v^{(1,1,0)}_{ts}$ and $v^{(1,1,0)}_{fs}$ reproduce. If $v^{(1,1,0)}_{ts}$ does, then $c^{(1,1,0,0)}_3$ does as well, repeatedly. Eventually, $v^{(1,1,0)}_{fs}$ reproduces and the Turing machine is in state 2 at position 2, with 0 on the tape at position 1 (as seen by $v^{(2,2,0)}_{tv}$ and v^2_{fv} being mutants).

The graph $G(A, I, P)$ has the wanted properties. It is straightforward, following the description in the construction, to see that if v_s becomes mutant at first, then fixation is ensured with probability 1 if A accepts input I using at most $P(|I|)$ space. If A does not accept I using at most $P(|I|)$ space, then $v_\perp$ cannot become a mutant. It follows that given v_s becomes a mutant at first, then the fixation probability is 1 if A accepts I with space at most $P(|I|)$, otherwise the fixation probability is 0. Note that initialization, each step of the execution, and the fixation stage might take long, but we have that each ends with probability 1 after a finite number of steps. Note that the PSPACE-hardness for qualitative question follows.

Lemma 7. *The qualitative decision question for the general* I&R *model is PSPACE-hard.*

Amplifying the probability: The graph $G'(A, I, P, p)$. We now describe how to, given a polynomial p, increase the fixation probability of the graph $G(A, I, P)$, if the Turing machine accepts with polynomial space from $\frac{1}{N}$ to $1 - \frac{1}{p(n)}$.

The graph $G'(A, I, P, p)$ from $G(A, I, P)$ and p. We create a new graph $G'(A, I, P, p)$ as follows: We add $k = N \cdot (p(n) - 1)$ new vertices $v^1, v^2, \ldots, v^k$, such that, for all $i \neq k$, the vertex v^i has $\widehat{v}_s$ as the only successor in E_I and v^{i+1} in E_R. The vertex v^k has $\widehat{v}_s$ as the only successor in E_I and v_s and v^1 as the successors in E_R. The check vertex c_1 has, besides the successors in E_I defined in the construction of $G(A, I, P)$ also the vertices v^i for all i as successors in E_I.

The graph $G'(A, I, P, p)$ has the desired properties. Observe that if some vertex v^i has become a mutant at the start, then each vertex v^j can become mutants (one after the other) and eventually also v_s. Note also that the vertex v^i can keep reproducing mutants till $\widehat{v}_s$ has become a mutant. At the time when $\widehat{v}_s$ has become a mutant, the vertex c_1 must have changed b_1 to true (given that v^i was the first mutant for some i). But in that case all vertices v^j have become mutants and remain mutants. Hence, using a argument like the above, we see that if we pick some vertex v^i to be the initial mutant, then the fixation probability is 1 if the Turing machine accepts and 0 otherwise. This shows that the fixation probability is either $1 - \frac{N}{N \cdot (p(n)-1)+N} = 1 - \frac{1}{p(n)}$, or 0, as desired.

Lemma 8. *Given an evolutionary graph $G = (V, E_I, E_R)$ in the general* I&R *model, a polynomial p, with the promise that the fixation probability in G is either (i) 0 or (ii) $1 - \frac{1}{p(|V|)}$, deciding between (i) and (ii) is PSPACE-hard.*

Hence we have the following result.

Theorem 9. *The following assertions hold for evolutionary graphs G in the general* I&R *model: (1) The qualitative decision question is PSPACE-complete. (2) For $0 < \epsilon < 1$ (specified in unary), with the promise that the fixation probability in G is either (i) 0 or (ii) $1 - \epsilon$, deciding between (i) and (ii) is PSPACE-hard; and the approximation of the fixation probability with double exponentially small error probability can be achieved in polynomial space. (3) The fixation probability can be computed in exponential time.*

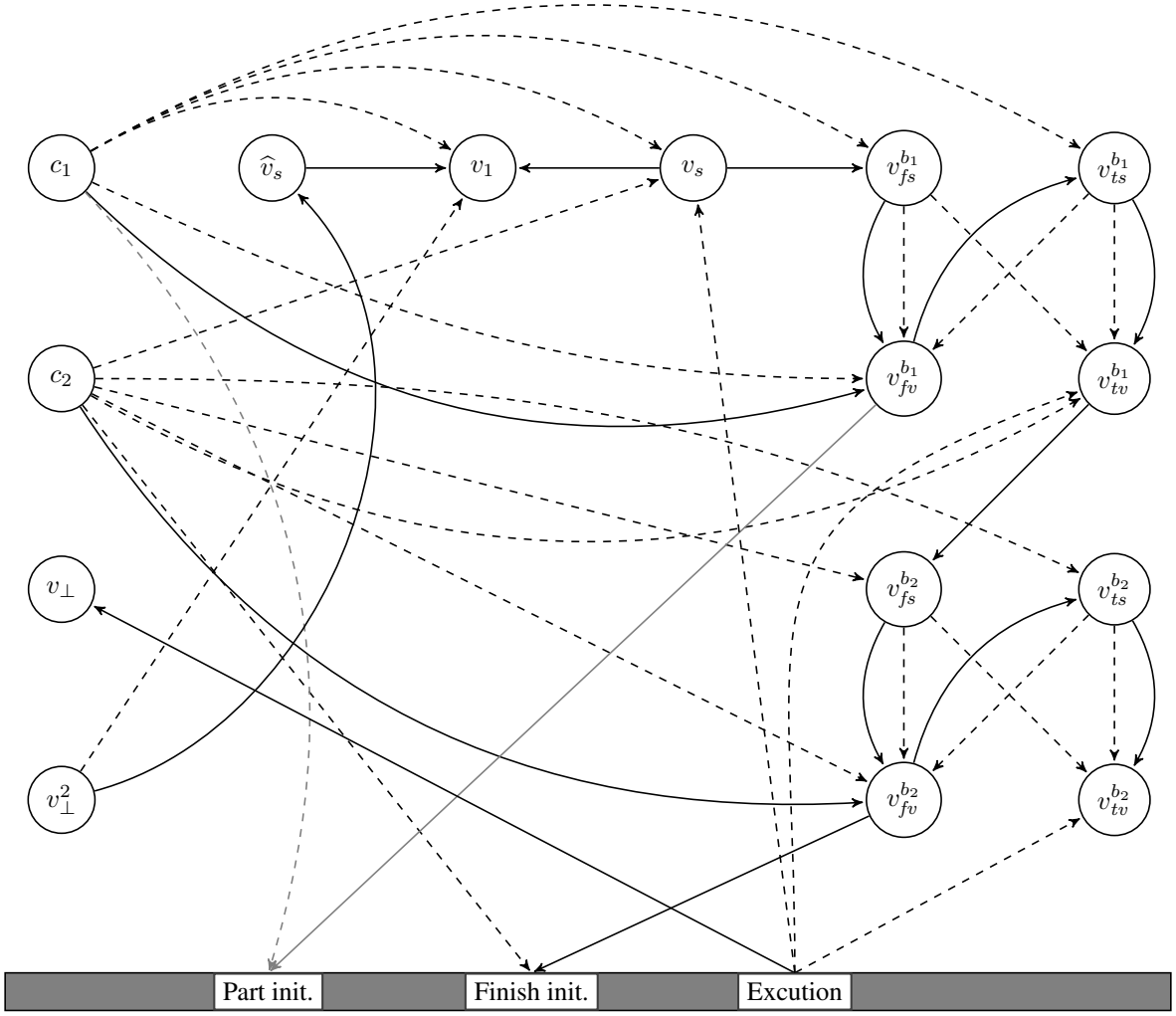

Figure 8: The part of the graph $G(A, I, P)$ not including the Turing machine (the Turing machine is in the gray box). Some edges are not included to make the graph more readable.

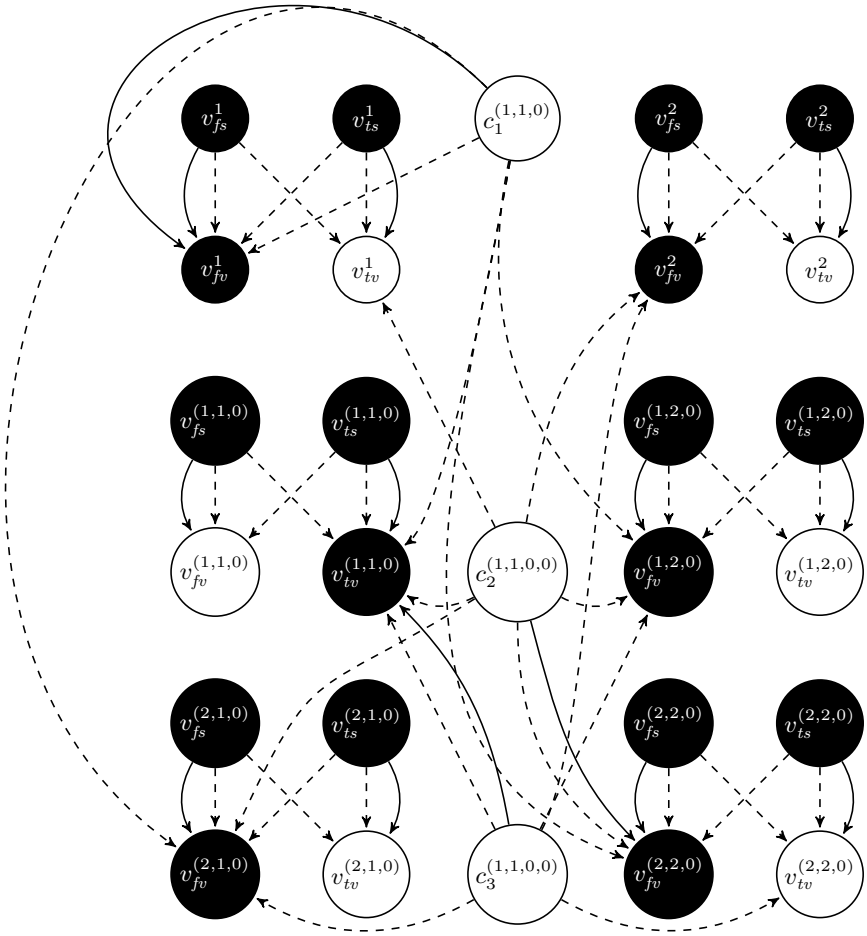

Figure 9: Part of the operation of the Turing machine. Black vertices with white text are mutants.

Remark 10 (Matrix encoding of density constraints)**.** *Note that in our results for lower bounds we consider density constraints of $\frac{1}{2} - \delta$, for $0 < \delta < 1/10$ (in Section 3 and Section 4) and 0 in this section. In all the lower bounds, the payoff matrix is constant, and for the first two lower bounds $r = 0$. The density constraints can be encoded as a payoff matrix (that is not constant) as follows:*

$$\begin{array}{cc} & \begin{array}{cc} R & M \end{array} \\ \begin{array}{c} R \\ M \end{array} & \begin{pmatrix} 0 & 0 \\ 1 & -1 \end{pmatrix} ; \end{array} \qquad \begin{array}{cc} & \begin{array}{cc} R & M \end{array} \\ \begin{array}{c} R \\ M \end{array} & \begin{pmatrix} -N & 1 \\ 1 & -N \end{pmatrix} ; \end{array}$$

the first payoff matrix encodes that a vertex that is a mutant can reproduce only if strictly less than half of the successors in E_I are mutants; and the second matrix (for vertex set of size N) encodes that a vertex can reproduce only if all the successors in E_I are of the opposite type. Note that with the matrix encoding the PPBT property still holds for $\#\mathsf{P}$-hard lower bounds, and hence the lower bound proof argument remains unchanged.

Concluding remarks. In this work we studied the complexity of basic computation questions for evolution on graphs. We established many lower and upper bounds. An interesting open question is the exact complexity of the quantitative approximation question for the general I&R model. Our paper widens the reach of complexity investigations to the computation of fixation probability in evolutionary graph theory, a fundamental problem in evolution. While we establish several important complexity results (in many cases precise complexity bounds), further investigations are necessary to establish precise complexity bounds for some of the problems.

Acknowledgements. We thank Erez Lieberman for insightful discussions and sharing his thoughts on the problem.

References

[1] B. Allen, C. Sample, Y. A. Dementieva, R. C. Medeiros, C. Paoletti, and M. A. Nowak. The molecular clock of neutral evolution can be accelerated or slowed by asymmetric spatial structure. arxiv:1409.5459, 2014.

[2] C. Baier, M. Größer, and N. Bertrand. Probabilistic ω-automata. *J. ACM*, 59(1):1, 2012.

[3] N. Barton, D. E. Briggs, J. A. Eisen, D. B. Goldstein, and N. H. Patel. *Evolution*. Cold Spring Harbor Laboratory Press, 2007.

[4] M. Broom and J. Rychtár. An analysis of the fixation probability of a mutant on special classes of non-directed graphs. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Science*, 464(2098):2609–2627, 2008.

[5] K. Chatterjee, A. Pavlogiannis, B. Adlam, and M. A. Nowak. The time scale of evolutionary innovation. *PLoS Computational Biology*, 10(9), 2014.

[6] K. Chatterjee and M. Tracol. Decidable problems for probabilistic automata on infinite words. In *LICS 2012*, pages 185–194, 2012.

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[8] F. Débarre, C. Hauert, and M. Doebeli. Social evolution in structured populations. *Nature Communications*, 2014.

[9] K. Etessami and A. Lochbihler. The computational complexity of evolutionarily stable strategies. *Int. J. Game Theory*, 37(1):93–113, 2008.

[10] W. Ewens. *Mathematical Population Genetics 1: I. Theoretical Introduction*. Interdisciplinary Applied Mathematics. Springer, 2004.

[11] N. Fijalkow, H. Gimbert, and Y. Oualhadj. Deciding the value 1 problem for probabilistic leaktight automata. In *LICS 2012*, pages 295–304, 2012.

[12] M. Frean, P. B. Rainey, and A. Traulsen. The effect of population structure on the rate of evolution. *Proceedings of the Royal Society B: Biological Sciences*, 280(1762), 2013.

[13] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[14] A. L. Hill, D. G. Rand, M. A. Nowak, and N. A. Christakis. Emotions as infectious diseases in a large social network: the SISa model. *Proceedings of the Royal Society B: Biological Sciences*, 277:3827–3835, 2010.

[15] J. E. Hopcroft and R. M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (Swat 1971)*, SWAT '71, pages 122–125, Washington, DC, USA, 1971. IEEE Computer Society.

[16] S. Karlin and H. M. Taylor. *A First Course in Stochastic Processes, Second Edition*. Academic Press, 2 edition, Apr. 1975.

[17] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[18] L. A. Levin. Universal sequential search problems. *Problems of Information Transmission*, 9(3):265–266, 1973.

[19] E. Lieberman, C. Hauert, and M. A. Nowak. Evolutionary dynamics on graphs. *Nature*, 433(7023):312–316, Jan. 2005.

[20] F. Michor, T. P. Hughes, Y. Iwasa, S. Branford, N. P. Shah, C. L. Sawyers, and M. A. Nowak. Dynamics of chronic myeloid leukaemia. *Nature*, 435(7046):1267–1270, 06 2005.

[21] P. A. P. Moran. *The Statistical Processes of Evolutionary Theory*. Oxford University Press, Oxford, 1962.

[22] M. A. Nowak. *Evolutionary Dynamics: Exploring the Equations of Life*. Harvard University Press, 2006.

[23] M. A. Nowak, N. L. Komarova, A. Sengupta, P. V. Jallepalli, I.-M. Shih, B. Vogelstein, and C. Lengauer. The role of chromosomal instability in tumor initiation. *Proceedings of the National Academy of Sciences*, 99(25):16226–16231, 2002.

[24] M. A. Nowak and R. M. May. Evolutionary games and spatial chaos. *Nature*, 359:826, 1992.

[25] M. A. Nowak, F. Michor, and Y. Iwasa. The linear process of somatic evolution. *Proceedings of the National Academy of Sciences*, 100(25):14966–14969, 2003.

[26] M. A. Nowak, A. Sasaki, C. Taylor, and D. Fudenberg. Emergence of cooperation and evolutionary stability in finite populations. *Nature*, 428(6983):646–650, 2004.

[27] H. Ohtsuki, C. Hauert, E. Lieberman, and M. A. Nowak. A simple rule for the evolution of cooperation on graphs and social networks. *Nature*, 441:502–505, 2006.

[28] H. Ohtsuki, J. M. Pacheco, and M. A. Nowak. Evolutionary graph theory: Breaking the symmetry between interaction and replacement. *Journal of Theoretical Biology*, 246(4):681 – 694, 2007.

[29] S. Otto and T. Day. *A Biologist's Guide to Mathematical Modeling in Ecology and Evolution*. Princeton University Press, 2011.

[30] P. Shakarian, P. Roos, and A. Johnson. A review of evolutionary graph theory with applications to game theory. *Biosystems*, 107(2):66 – 80, 2012.

[31] C. E. Tarnita, H. Ohtsuki, T. Antal, F. Fu, and M. A. Nowak. Strategy selection in structured populations. *Journal of Theoretical Biology*, 259(3):570–81, 2009.

[32] L. G. Valiant. The complexity of computing the permanent. *Theor. Comput. Sci.*, 8:189–201, 1979.

[33] N. Vishnoi. The speed of evolution. In *SODA 2015*, 2015.