Simulating Quadratic Dynamical Systems is PSPACE-complete (preliminary version)

Sanjeev Arora* UC Berkeley

Yuval Rabani† MIT

Umesh Vazirani‡ UC Berkeley

Abstract

Quadratic Dynamical Systems (QDS), whose definition extends that of Markov chains, are used to model phenomena in a variety of fields like statistical physics and natural evolution. Such systems also play a role in genetic algorithms, a widely-used class of heuristics that are notoriously hard to analyze. Recently Rabinovich et al. took an important step in the study of QDS's by showing, under some technical assumptions, that such systems converge to a stationary distribution (similar theorems for Markov Chains are well-known). We show, however, that the following sampling problem for QDS's is PSPACE-hard: Given an initial distribution, produce a random sample from the t'th generation. The hardness result continues to hold for very restricted classes of QDS's with very simple initial distributions, thus suggesting that QDS's are intrinsically more complicated than Markov chains.

*Supported by an IBM Graduate Fellowship and partly under NSF grant CCR-9310214. Email: arora@cs.berkeley.edu.

†Work done while at ICSI, Berkeley, and supported in part by a Rothschild postdoctoral fellowship. Email: rabani@theory.lcs.mit.edu.

‡Supported by NSF grant CCR-9310214. Email: vazirani@cs.berkeley.edu.

1 Introduction

Quadratic dynamical systems (QDS) have recently attracted some interest in the computer science community, particularly as a model for genetic algorithms. They were classically used to model various natural phenomena in physics and biology. Like a Markov chain (a linear system), a QDS maps a probability distribution $\mathcal{D}$ on a state space S into another probability distribution $\mathcal{D}'$ (on S). The map is quadratic in that two random samples are picked independently from $\mathcal{D}$ to produce a random sample from $\mathcal{D}'$ according to some fixed mating rule β.

A classical use of the QDS model in physics is Boltzmann's treatment of the kinetic gas model of Maxwell (see [Rei]). The result of collisions between pairs of molecules can be described by a mating rule derived from the laws of Newtonian mechanics. By assuming that the velocities of the molecules of gas in a chamber are independent (a problematic assumption, which leads to the *entropy* paradox), Boltzmann gave a QDS description of the dynamics of the gas, and derived the famous Maxwell-Boltzmann distribution for the steady state.

The QDS model also forms the basis of the Hardy-Weinberg approach for describing the evolution of a population (see [CB]). The state space is the set of individuals (represented by their *genotype*), and the mating rule is based upon the laws of genetics. (It might also incorporate the survival probability in the given environment, the mutation rate, etc.) The crucial assumption which allows the use of the QDS formulation is that mating pairs are chosen randomly and independently. Another application in this vein is Volterra's model of a predator-prey ecological system (see [HS]).

It has been proposed that the QDS formulation captures the essential features of genetic algorithms, a class of heuristics to solve optimization problems. Our results clarify this connection, and we discuss these implications in the next section.

Classical research on QDS's often focusses on the study of *fixed points* i.e., distributions in equilibrium. However, computational applications require upper bounds on the rate of convergence to equilibrium. For instance, applications of Markov chains in computer science [AKS, JS, DFK] were made possible by new techniques for analyzing convergence rates.

By contrast, QDS's do not necessarily converge to a stationary distribution. However, many QDS's arising in applications use a *symmetric* mating rule (see the next subsection for definitions). For such systems, Rabinovich, Sinclair and Wigderson [RSW] showed – using ideas from statistical physics – convergence to a stationary distribution. (Some technical assumptions need to be made about the initial distribution, and the stationary distribution may depend on the initial distribution.) Furthermore, spurred by the use of Markov chains in applications like approximating the number of matchings in a bipartite-graph, or 0/1 permanent [JS], the authors suggested similar applications of QDS's. They analyzed the equilibrium distributions and the rate of convergence of a QDS related to the problem of counting matchings in trees.

Another such result is Rabinovich and Wigderson's analysis of a QDS (bitwise independent mating) that optimizes linear functions over the domain of n-bit strings [RW]. They analyze the rate of convergence of the QDS, which is used in [Rab] to obtain a polynomial time algorithm for the problem. This polynomial time algorithm does not work by faithfully simulating the quadratic operator, as one would expect, but by simulating some key properties of it.

As we will see, this is but one example of a more general problem with QDS's: we will show that given a symmetric mating rule that is polynomial time computable, and a simple initial distribution $\mathcal{D}_0$ (say, one that concentrates all probability in one state) it is PSPACE-hard to produce a random sample according to $\mathcal{D}_t$, the distribution after t generations (the t used in our reductions is small, so $\mathcal{D}_t$ is not the equilibrium distribution).

1.1 Definitions and Results

A *Quadratic Dynamical System* (QDS) consists of a state space S, and a mating operator $\beta : S^4 \rightarrow \Re^+ \cup \{0\}$. Following convention, we denote $\beta(i,j,k,l)$ by β_{ijkl}. We assume β has been normalized, so that $\sum_{k,l} \beta_{ijkl} = 1$. In addition, β is said to be *symmetric* if $\forall i,j \in S, \beta_{ijkl} = \beta_{jikl}$, and $\forall i,j,k,l \in S, \beta_{ijkl} = \beta_{klij}$; in other words, β is locally reversible.

A *population* is a probability distribution on S. The mating operator describes the evolution of the population (in discrete time steps) according to the following algorithm.

Evolution Rule The following is the algorithm to go from the distribution at time t, $\mathcal{D}_t$, to the distribution at time $t+1$, $\mathcal{D}_{t+1}$.

Take two independent samples i and j from $\mathcal{D}_t$.
Construct two samples $k, l \in S$ according to the following probability distribution.
$\Pr[(k,l)] = \beta_{ijkl}$.
Output k.
STOP.

The sequence $\mathcal{D}_0, \mathcal{D}_1, \ldots, \mathcal{D}_t, \ldots$ of points in the $|S|$-dimensional simplex is called a *trajectory.*

The following is an explicit formula for $\mathcal{D}_{t+1}$ in terms of $\mathcal{D}_t$.

$$\mathcal{D}_{t+1}(k) = \sum_{i,j,l} \mathcal{D}_t(i)\mathcal{D}_t(j)\beta_{ijkl}.$$

Quadratic Dynamical Systems get their name from the quadratic nature of the above expression. Note that Markov chains are *linear* dynamical systems, in contrast.

An algorithm is said to *sample* from a distribution $\mathcal{D}$ if it outputs a random element x according to the distribution $\mathcal{D}$ (i.e. x is output with probability $\mathcal{D}(x)$). We consider sampling algorithms to be efficient if their expected running time is polynomial in the length of the samples. Sometimes we allow the sampling algorithm to output samples from a distribution $\mathcal{D}'$ whose ℓ_1-*distance* from the desired distribution $\mathcal{D}$ is at most ϵ (for a given error parameter ϵ), that is:

$$|\mathcal{D} - \mathcal{D}'|_1 \quad = \quad \sum_x |\mathcal{D}(x) - \mathcal{D}'(x)| \quad \leq \epsilon.$$

Any notion of efficient sampling from a QDS makes sense only for QDS's (and initial distributions) that are themselves computable in $poly(n)$ time, where $n = \log |S|$. We say that a QDS β is *succinctly-defined* if there is a probabilistic polynomial time machine T_β which, on input (i, j), can produce a sample (k, l) according to the distribution $Pr[(k, l)] = \beta_{ijkl}$.

We assume for now that the initial distribution $\mathcal{D}_0$ is polynomial time samplable. It will be seen that the actual $\mathcal{D}_0$ used in our reduction is just the *point distribution*: it assigns a probability 1 to some specific element of S and probability 0 to everything else.

Definition. *Sampling Problem:*
INPUT: $|S|$ (in binary notation); a succinctly-defined QDS β, given as a Turing machine T_β; an initial distribution $\mathcal{D}_0$, given as a Turing Machine; 1^t (string of t 1's); and an error parameter ϵ.
OUTPUT: A sample $x \in S$ according to a distribution $\mathcal{D}'$, such that $|\mathcal{D}' - \mathcal{D}_t|_1 \leq \epsilon$.

A machine is said to *simulate* the QDS if it solves the corresponding sampling problem. A *reduction* from sampling problem A to sampling problem B is a polynomial time transformation from A to B, such that simulating B (with the new distribution) allows a simulation of A. Our result that simulating symmetric QDS's is PSPACE-hard follows easily (in Section 4) from the following more general reduction.

Theorem 1 *Sampling an arbitrary QDS (i.e., even a non-symmetric one) with a poly-time samplable initial distribution can be reduced to sampling some symmetric QDS with an initial distribution that is a point distribution.*

This theorem, which will be proved in Section 3, is somewhat surprising since removing the symmetry condition from β is known to result in systems with extremely complex behavior [HS]. For instance, the convergence theorem in [RSW] is known to not hold then.

An interesting corollary of Theorem 1 is that simulating a general k-adic dynamical system (i.e., one with k-ary interactions instead of binary ones) also reduces to the simulation of a symmetric QDS. Such systems have found some applications in computer science. For instance, Valiant [Val] uses a nonlinear system with 4-way interactions to generate, and thus show the existence of, small sized monotone circuits for majority. Clarkson et al. [CEMST] give algorithms for approximating the center point of a set of points in $\mathbb{R}^d$. They use, in effect, a simulation of a $(d + 2)$-adic dynamical system that converges to the desired result.

2 Why Simulating a QDS is Hard: Intuition and Examples

This section explains the basic intuition why simulating a QDS is a non-trivial problem, whereas simulating Markov chains is not. The casual reader might read this section to get a flavor of the reductions in the remaining sections. The discussion in this section also ties in to some research on Genetic Algorithms, a field in which QDS's (or similar systems) fit in naturally. We will restrict attention to QDS's and Markov chains that are succinctly-defined by polynomial time Turing machines. Also, for the purposes of this section, the initial distribution $\mathcal{D}_0$ is required to be merely polynomial-time samplable.

To sample from $\mathcal{D}_t$ for a Markov chain, we can just generate one sample from $\mathcal{D}_{t-1}$, and apply one step of the chain to it. Unrolling the recursion implicit in the previous statement, the following simple algorithm is obtained: Generate a sample from $\mathcal{D}_0$ and take it through t steps of the chain.

For a QDS, producing a sample from $\mathcal{D}_t$ seems to require *two* independent samples from $\mathcal{D}_{t-1}$. Unrolling the implicit recursion yields the following algorithm: Generate 2^t samples — or 2^{t-1} pairs of samples — from $\mathcal{D}_0$. Apply the mating operator to each pair to obtain 2^{t-1} independent samples from $\mathcal{D}_1$. Go on like this for $t - 1$ steps until a single sample from $\mathcal{D}_t$ remains.

It is easy to see that one can covert the above algorithm to run in polynomial space. The following procedure does that.

$L_0, L_1, L_2, \ldots, L_t := \emptyset$.
While $L_t = \emptyset$ do
  Generate a sample from $\mathcal{D}_0$ and add it to L_0.
  For $i := 0$ to $t-1$ do
    If $|L_i| = 2$ then
      Mate the two samples in L_i and add the child to L_{i+1}.
      $L_i := \emptyset$.
Output L_t.
STOP.

Actually, this simulation can be done for a wider class of QDS's. We only need that mating can be done by a polynomial space bounded Turing machine. If mating is done in linear space, the simulation runs in linear space.

Does one inherently need 2^t samples from $\mathcal{D}_0$ to produce a sample from $\mathcal{D}_t$? It is easy to verify that for a $\mathcal{D}_0$ given as a "black box" (which generates samples from $\mathcal{D}_0$ for us) this bound is tight. The following example (a weaker form of our results) shows that the sampling problem is NP-hard even when $\mathcal{D}_0$ is polynomial-time samplable.

An NP-hard Sampling Problem. Let φ be a SAT instance with n variables. We define the following symmetric QDS.
$S = \{0,1\}$. $\beta_{0000} = 1$, $\beta_{0101} = 1/3$, $\beta_{0111} = 1/3$ (other values of β can be determined using symmetry). $\mathcal{D}_0$ is defined as $Pr[1] = 2^{-n} \cdot |\{x : x \text{ is a satisfying assignment to } \varphi\}|$, and $Pr[0] = 1 - Pr[1]$.

$\mathcal{D}_0$ is polynomial time samplable, since a machine could just generate a random assignment, and output 1 iff it satisfies φ. The operator β can be described in words as follows: *When a* 0 *and a* 1 *mate, they produce a* $(1,1)$ *with probability* $1/3$. *When a* 1 *and a* 1 *mate, they produce a* 0 *and a* 1 *with probability* $1/3$. *In every other case, there is no change.* Clearly, this system tends to equalize the probability of 1's and 0's. More specifically, if p_t is the probability of 1 in $\mathcal{D}_t$, then

$$p_{t+1} = \frac{2}{3}p_t^2 + \frac{4}{3}p_t(1-p_t) \quad = \frac{2}{3}p_t(2-p_t).$$

If φ is not satisfiable, $p_0 = 0$, hence $p_t = 0 \; \forall \; t$. If φ is satisfiable, then $p_0 \geq 2^{-n}$, and $\forall t$, if $p_t \leq \frac{1}{4}$, then $p_{t+1} \geq \frac{7}{6}p_t$. Therefore p_t increases geometrically, and for $t \geq 5n$, $p_t \geq 1/4$. Thus a random sample from $\mathcal{D}_{5n}$ allows us to decide the satisfiability of φ with probability $1/4$. ■

2.1 Implications for Genetic Algorithms

The number of samples from $\mathcal{D}_0$ required to produce a sample in $\mathcal{D}_t$ is related to the notion of *population size* encountered in the context of genetic algorithms (GAs). Such algorithms try to find better and better solutions to an optimization problem (say, Traveling Salesman Problem) by "mating" solutions that have already been found. The algorithm maintains a set of of possible solutions (a given solution may occur with a certain multiplicity) and at every iteration, repeatedly mates two randomly-chosen solutions to create the next generation of solutions. In an effort to imitate evolution in nature, the mating operator is augmented using rules that ensure the "survival of the fittest" and "mutations."

The maximum number of solutions kept around by the GA is called its population size. When the population size is infinite the action of the mating operator on the population defines exactly a QDS, and the distribution of solutions after the tth iteration is $\mathcal{D}_t$. But infinite (or even very large) population sizes are infeasible in real genetic algorithms. GAs with finite population sizes are not true quadratic systems, but Markov chains, as pointed out in [NV]. However, in an attempt to link real GAs (i.e. those with finite population sizes) with quadratic systems it was shown [Vos] that with a "large enough" population size, a real GA behaves like the corresponding QDS for a "long" time.

We must point out however (and this is a simple exercise), that to correctly simulate the QDS used in GAs for t steps in general requires a population size of $2^{O(t)}$. This suggests a need for a fresh look at whether the QDS model is appropriate for real GA's. (In fact we can show that simulating the fitness operator in genetic algorithms is NP-hard, using ideas similar to the above toy example.) Even in our above example, note that when $p_0 = 2^{-n}$, we need an initial population of size 2^n — i.e. this many samples from $\mathcal{D}_0$ – to have a good probability of including 1's in the population. A much smaller starting population will never exhibit any 1's, and will therefore diverge from the true $\mathcal{D}_t$ exponentially fast. Later we will see examples of systems where the initial population can be small (even of size 1) but a faithful simulation of later generations requires large population sizes unless BPP = PSPACE.

3 Simulating General Quadratic Systems by Symmetric Systems

Let β be a succinctly-defined QDS. Given any time interval t, and error parameter ϵ, we show how to transform β into another succinctly-defined QDS $\hat{\beta}$, such that $\hat{\beta}$ is symmetric, and for any initial distribution $\mathcal{D}_0$, the distributions after t steps under β and $\hat{\beta}$, $\mathcal{D}_t$ and $\hat{\mathcal{D}}_t$ respectively, are within l_1-distance at most ϵ.

Let S be the state space of β. Let $m \geq |S|$. The state space of $\hat{\beta}$ is $Z = \cup_{j=0}^{t} S \times [m]^j$, where $[m] = \{1, \ldots, m\}$. If a state $\alpha \in Z$ is an element of $S \times [m]^j$, we shall say that $level(\alpha) = j$, and write $\alpha = (\alpha_0, \ldots, \alpha_j)$.

The transition probabilities of the new QDS are defined as follows: If $level(\alpha) \neq level(\alpha')$, then $\hat{\beta}_{\alpha,\alpha',\alpha,\alpha'} = 1/2$, and $\hat{\beta}_{\alpha,\alpha',\alpha',\alpha} = 1/2$.

If $level(\alpha) = level(\alpha') = k$, then let $\alpha_0 = s$, and $\alpha'_0 = s'$. Let states $u, u' \in S$ be such that $\beta_{s,s',u,u'} = p$.

The forward transitions of $\hat{\beta}$ are defined as follows: Let $\mu, \mu' \in Z$ satisfy $level(\mu) = level(\mu') = k+1$, and $\mu_0 = u$, and $\mu_i = \alpha_i$ for $1 \leq i \leq k$, $\mu'_0 = u'$, and $\mu'_i = \alpha'_i$ for $1 \leq i \leq k$. Then $\hat{\beta}_{\alpha,\alpha',\mu,\mu'} = p(1-(|S|/m)^2)/m^2$.

If $k = 0$, then the transitions are completed by adding a self-loop of probability $(|S|/m)^2$ for each pair. i.e. $\hat{\beta}_{\alpha,\alpha',\alpha,\alpha'} = (|S|/m)^2$.

If $k \geq 1$, there are both reverse transitions and self-loops: The probability of a reverse transition, from a pair of level k+1 states to a pair of level k states, is defined to be exactly equal to the probability of the corresponding forward transition (from the pair of level k states to the pair of level k+1 states). The self-loop for a pair of states has probability $(|S|/m)^2$ minus the sum of the probabilities of the reverse transitions from that pair.

It is clear from the definition that $\hat{\beta}$ is symmetric.

Lemma 2 *$\hat{\beta}$ is succinctly-defined.*

Proof. (*sketch*) Let α and α' be level k states for $k \geq 1$. Let s, s' be as defined above. Then $T_{\hat{\beta}}[\alpha, \alpha']$ can be computed as follows:

- With probability $1-(|S|/m)^2$: let $T_\beta[s,s'] = u, u'$. Define a pair of level K+1 states μ, μ' as follows: $\mu_0 = u$, and $\mu_i = \alpha_i$ for $1 \leq i \leq k$, and μ_{k+1} is chosen uniformly at random in $[m]$. Similarly, $\mu'_0 = u'$, and $\mu'_i = \alpha'_i$ for $1 \leq i \leq k$, μ'_{k+1} is chosen uniformly at random in $[m]$. Output μ, μ'.
- Otherwise, with probability $1-(|S|/m)^2$: choose a pair of states $q, q' \in S$ uniformly at random. If $T_\beta[q,q'] = s, s'$ then (reverse transition) define a pair of level $k-1$ states ν, ν', with $\nu_0 = q$, $\nu_i = \alpha_i$ for $1 \leq i \leq k-1$, and $\nu'_0 = q'$, $\nu'_i = \alpha'_i$ for $1 \leq i \leq k-1$. Output ν, ν'. Else if $T_\beta[q,q'] \neq s, s'$ (self-loop), output α, α'.
- Otherwise (self-loop) output α, α'.

It is easy to complete the definition of procedure $T_{\hat{\beta}}$ for the remaining values of α and α'. Also, it is readily verified that if T_β is a polynomial time procedure, then so is $T_{\hat{\beta}}$.

■

Let $\mathcal{D}_0$ denote the initial distribution from which we wish to simulate the quadratic system β for t steps. We now analyze the behavior of the new quadratic system $\hat{\beta}$ on the same initial distribution (note: $S \subseteq Z$). Let $\hat{\mathcal{D}}_k$ denote the distribution after k steps under $\hat{\beta}$.

Lemma 3 *Under the distribution $\hat{\mathcal{D}}_k$:*

- *The probability of any state whose level is larger than k is* 0.
- *The total probability of level k states is*

$$(1-(|S|/m)^2)^{2^k}$$

.

- *The conditional distribution of the first coordinate of the level k states is exactly $\mathcal{D}_k$.*

Proof. (*sketch*) The first statement is obvious. The second statement follows by induction on k, and the observation that the total probability of forward transitions from any pair of states is

$(1-(|S|/m)^2)$. The third statement follows by induction on k and the fact that when restricted to the first coordinate, the forward transitions from any pair of states under $\hat{\beta}$ exactly mimic the transition probabilities according to $\hat{\beta}$. ■

Proof of Theorem 1. Let β be an arbitrary QDS and $\mathcal{D}_0$ a polynomial time samplable distribution. By adding a new state, whose sole interaction is a (irreversible) transformation to the other states according to $\mathcal{D}_0$, we can assume w.l.o.g. that the initial distribution is concentrated on only one state. Using the above notation, we wish to sample from $\mathcal{D}_t$.

As in the above reduction we define $\hat{\beta}$ with $m = |S|2^t$. The size of the state space for $\hat{\beta}$ is at most $\max\{|S|\cdot(|S|^2+1)^t, |S|\cdot(2^t+1)^t\}$, so a state can be coded using $(2t^2+1)\lceil\log|S|\rceil$ bits at most. Since the probability of level t states under the distribution $\hat{\mathcal{D}}_t$ is at least $\left(1-\frac{1}{2^{2t}}\right)^{2^t-1} \geq e^{-1}$, by taking an expected $O(1)$ samples according to $\hat{\beta}$ at the t'th generation we will obtain a sample from the t'th generation of β. ■

Corollary 4 *Let $p(x)$ be a polynomial. Consider the family of k-adic dynamical systems, where $k \leq p(\log|S|)$. The simulation of this family polynomially reduces to the simulation of symmetric quadratic dynamical systems.*

Proof. We reduce the simulation of k-adic systems to the simulation of quadratic systems. By Theorem 1 this is sufficient to prove the claim.

For every k-adic system considered, we construct the following quadratic system. The state space consists of all 2^i multisets of states in S, where $0 \leq i \leq \lceil\log k\rceil - 1$. So each state can be coded using $\log|S|\cdot p(\log|S|)$ bits at most. There are two types of transitions. The first kind of transitions mate, for every i, $0 \leq i < \lceil\log k\rceil - 1$, two 2^i multisets that produce their union — a 2^{i+1} multiset. The second kind of transitions mate two $2^{\lceil\log k\rceil-1}$ multisets, in which case a subset of size k of their union is chosen uniformly at random and the outcome of the mating of these k states under the original dynamical system is produced. So, each step of the original system is simulated by $\lceil\log k\rceil - 1$ steps of the quadratic system. ■

4 PSPACE-Hardness

In this section we show that simulating general QDS's (i.e. those with no constraints on β) is PSPACE-hard. Combining this result with the one in the previous section about symmetric QDS's being able to simulate general QDS's, we conclude that simulating symmetric QDS's is PSPACE-hard. The PSPACE-complete problem we use in this section is the language of True Quantified Boolean Formulae (TQBF) [SM].

$$TQBF = \{\psi : \psi = \exists x_1 \forall x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n),\ \phi \text{ is a 3SAT formula}\}.$$

Let $\psi = \exists x_1 \forall x_2 \ldots Q_n x_n \phi(x_1, x_2, \ldots, x_n)$ be an instance of $TQBF$. By "unrolling" ψ we can define a circuit with the following properties, in the obvious way.

1. There are $n+1$ levels numbered 0 to n. Level i has 2^i gates, each labeled by a unique i-bit binary string. Each gate has fanin 2 and fanout 1 (except for gates at level n, whose values we set, and the top gate, which produces the output).

2. Gates at level n are set to 0 or 1 as follows: gate $b_1 b_2 \ldots b_n$ ($b_i \in \{0,1\}$), is 1 iff $\phi(b_1, b_2, \ldots, b_n) = 1$.

3. All gates at level i are AND's if the i'th quantifier in ψ, Q_i, is $\forall$, and are OR's otherwise. The inputs to the gate $b_1 \ldots b_i$ at level i ($i < n$) are from gates $b_1 \ldots b_i 0$ and $b_1 \ldots b_i 1$ of level $i+1$.

4. The top gate evaluates to 1 iff ψ is true. (This follows from 1 through 3.)

.

Let G be the set of gates in the above circuit. The QDS we define will emulate the operation of the circuit.

$$S = G \times \{0,1\}.$$

The initial distribution, $\mathcal{D}_0$, is the uniform distribution on the following set of size 2^n

$\{(g,b):$ g is a gate at level n and b is its assigned value$\}$.

The mating operator β is such that $\forall i,j,k,l \in S^4$, $\beta_{i,j,k,l}$ is either 1 or 0. We use the notation $(i,j) \rightarrow (k,l)$ to denote $\beta_{i,j,k,l} = 1$.

Rules for Mating.

1. $((g_1,b_1),(g_2,b_2)) \rightarrow ((g_3,b_3),(g_3,b_3))$ if gates g_1, g_2 are inputs to gate g_3, and b_3 is $g_3(b_1,b_2)$ (note: g_3 is either AND or OR).
2. $((g_1,b_1),(g_2,b_2)) \rightarrow ((g_1,b_1),(g_1,b_1))$ if gate g_1 is at a higher level in the circuit than g_2. Likewise $((g_1,b_1),(g_2,b_2)) \rightarrow ((g_2,b_2),(g_2,b_2))$ if g_2 is at a higher level.
3. $((g_1,b_1),(g_2,b_2)) \rightarrow ((g_1,b_1),(g_2,b_2))$ if neither of the previous conditions is true for g_1 and g_2.

■

The basic idea of how the QDS simulates the circuit should be clear: $\mathcal{D}_0$ corresponds to the leaves of the circuit getting assigned correctly. Pairwise interaction of states according to Rule 1 performs computation at an inner gate g_3; this is how the correct values filter up the circuit. Interaction according to Rule 2 boosts the probability of the gate that's closer to the output; this makes the probabilities go up the circuit very fast. Interaction according to Rule 3 leaves the states unchanged. The following lemma is immediate.

Lemma 5 *If at any time t, $\Pr[(g,b)] > 0$, then b is the correct value of gate g.*

■

Therefore $\Pr[(g,b)]$, if it is nonzero, may unambiguously be called $\Pr[g]$.

Lemma 6 *If g_1, g_2 are gates at the same level, then for all times $t \geq 0$, $\Pr[g_1] = \Pr[g_2]$.*

Proof. The claim is true at $t = 0$. Suppose it is true $\forall t \leq k$. We will show that it is then true for $t = k+1$, whence the lemma follows by induction. But the interactions in Rules 1 through 3 treat all gates at the same level identically. So if probabilities do not vary within a level at time k, rules 1 through 3 will not change probabilities within any level at time $k+1$. ■

Let $p_{i,t}$ denote the probability of a level-i gate at time t. Then Rule 1 implies

$$p_{i,t+1} \geq 2p_{i+1,t}^2 \tag{1}$$

The probability of all gates at or below level i at time t, denoted as $P_{\geq i,t}$, is $2^n p_{n,t} + 2^{n-1}p_{n-1,t} + \cdots + 2^i p_{i,t}$. Since both Rules 1 and 2 move probability upward but never downward, $P_{\geq i,t}$ is non-increasing in t. More specifically since a state at a level $\geq i$ is produced only by mating two states from level $\geq i$, we have the following expression.

$$P_{\geq i,t+1} \leq P_{\geq i,t}^2. \tag{2}$$

Lemma 7 *For $t = 3ni+1$, $i \leq n$,*

$$P_{\geq n-i+1,t} \leq 2^{-n}.$$

Proof. We first prove the claim true for $i = 1$. Recall from the description of $\mathcal{D}_0$ that $P_{\geq n,0} = 1$. Then Equation 1 implies that at $t = 1$, some of the probability moves up, so that $P_{\geq n,1} \leq 1 - \frac{1}{2^{2n}}$. Now equation 2 implies that

$$P_{\geq n,t} \leq (1 - \frac{1}{2^{2n}})^{2^{t-1}},$$

and so $P_{\geq n,3n+1} \leq e^{-n}$.

Now assume the claim is true for $i \leq n+1-k$, and let $j = n+1-k$. Then

$$P_{\geq k,3jn+1} \leq 2^{-n}.$$

Since there are 2^{k-1} gates at level $k-1$, Lemma 6 implies that $p_{k-1,t} \leq \frac{1}{2^{k-1}}$ for any time t, and in particular for $t = 3jn+1$. Then Equation 1 implies that in one step, some of this probability moves up to level $k-2$, and $p_{k-1,3jn+2} \leq (1-\frac{1}{2^{k-1}})p_{k-1,3jn+1}$. Therefore

$$P_{\geq k-1,3jn+2} \leq (1 - \frac{1}{2^{k-1}} + \frac{1}{2^n}).$$

Again, Equation 2 shows that

$$P_{\geq k-1,t} \leq (P_{\geq k-1,t-3jn+2})^{2^{t-3jn-2}},$$

which for $t = 3(n+2-k)n+1$ gives the required conclusion. ■

This means that after time $t = 3n^2+1$, the output gate has probability ≈ 1, and a random sample from $\mathcal{D}_{3n^2+1}$ enables us to determine whether or not the circuit accepts.

5 Conclusions

We have exhibited symmetric QDS's whose simulation is PSPACE-hard. An interesting question is: What is the class of QDS's that *can* be simulated in polynomial time? Our reductions use systems that display chaotic behavior, i.e., two very close (in ℓ_1-norm) initial distributions diverge rapidly as they evolve. We initially suspected this chaotic behaviour was the reason why QDS's are hard to simulate. However, Moni Naor has recently noted that our NP-hard system of section 2 can be modified into a non-chaotic one [Nao]. We do not know whether the same is true for the PSPACE-hard system given later.

In some independent work, Pudlak [Pud] has also shown the PSPACE-hardness of simulating non-symmetric QDS's (this corresponds to our Section 4). He does not extend the result to symmetric QDS's, however.

An interesting question (which Pudlak also investigates) is whether the QDS's used in genetic algorithms are hard to simulate. One component of their mating operator, the crossover operator, can be easily simulated in polynomial time (this follows from the Hardy-Weinberg law of genetics). However, other components like *fitness* are non-symmetric, and we can show that simulating them is at least NP-hard. In general, we suspect that hard-to-simulate QDS's are more the norm than the exception. For instance, the fitness operator used in [RSW] in the QDS on tree-matchings is also NP-hard to simulate (with initial distribution that are polynomial time samplable). Proofs of these observations will appear in the final draft.

acknowledgments

Thanks to Steven Omohundro and Alistair Sinclair for helpful discussions and to Yuri Rabinovich for providing [Rab]. We thank the STOC '94 program committee members for bringing to our attention [Val] and [CEMST].

References

[AKS] M. Ajtai, J. Komlós, and E. Szemerédi. Deterministic simulation in LOGSPACE. In *Proc. of the 19th Ann. ACM Symp. on Theory of Computing*, page 132, 1987.

[CB] L. L. Cavalli-Sforza and W. F. Bodmer *The Genetics of Human Populations* W. H. Freeman, San Francisco, 1971.

[CEMST] K.L. Clarkson, D. Eppstein, G.L. Miller, C. Sturtivant, and S.H. Teng. Approximating center points with iterated Radon points. In *Proc. of the 9th Ann. ACM Symp. on Computational Geometry*, pages 91–98, 1993.

[DFK] M. Dyer, A. Frieze, and R. Kannan. A random polynomial time algorithm for approximating the volume of convex bodies. In *Proc. of the 21st Ann. ACM Symp. on Theory of Computing*, pages 375–381, 1989.

[HS] M. Hirsch and S. Smale. Differential Equations, Dynamical Systems, and Linear Algebra. Academic Press, New York, 1974.

[JS] M. Jerrum and A. Sinclair. Approximating the permanent. *SIAM Journal on Computing* 18:1149–1178, 1989.

[Nao] M. Naor. Private communication.

[NV] A.E. Nix and M.D. Vose. Modelling genetic algorithms with Markov chains. In *Annals of Mathematics and Artificial Intelligence,* vol. 5, pages 79-88, 1992.

[Pud] P. Pudlak. Private communication.

[Rab] Y. Rabinovich. Draft of Ph.D. dissertation.

[RSW] Y. RABINOVICH, A. SINCLAIR, AND A. WIGDERSON. Quadratic dynamical systems. In *Proc. of the 33rd Ann. IEEE Symp. on Foudations of Computer Science*, pages 304–313, 1992.

[RW] Y. RABINOVICH AND A. WIGDERSON. Analysis of a simple genetic algorithm. In *Proc. of the 4th International Conference on Genetic Algorithms*, pages 215–221, 1991.

[Rei] L.E. REICHL. *A Modern Course in Statistical Physics.* University of Texas Press, Austin, 1980,

[SM] L.J. STOCKMEYER AND A.R. MEYER. Word problems requiring exponential time. In *Proc. of the 5th Ann. ACM Symp. on Theory of Computing*, pages 1–9, New York, 1973.

[Val] L.G. VALIENT. Short monotone formulae for the majority function. *Journal of Algorithms* 5:363–366, 1984.

[Vos] M.D. VOSE. Modeling simple genetic algorithms. In *Foundations of Genetic Algorithms 2*, ed. Whitley, Morgan Kaufmann, pages 63–73, 1993.