Fast stochastic algorithm for simulating evolutionary population dynamics

William H. Mather1,2,3, Jeff Hasty1,2,3,4, Lev S. Tsimring2,3*

1Department of Bioengineering, University of California, San Diego, CA, USA
2BioCircuits Institute, University of California, San Diego, CA, USA
3San Diego Center for Systems Biology, San Diego, CA, USA
4Molecular Biology Section, Division of Biological Sciences, University of California, San Diego, CA, USA

Associate Editor: Dr. Jeffrey Barrett

ABSTRACT

Motivation: Many important aspects of evolutionary dynamics can only be addressed through simulations. However, accurate simulations of realistically large populations over long periods of time needed for evolution to proceed are computationally expensive. Mutant can be present in very small numbers and yet (if they are more fit than others) be the key part of the evolutionary process. This leads to significant stochasticity that needs to be accounted for. Different evolutionary events occur at very different time scales: mutations are typically much rarer than reproduction and deaths.

Results: We introduce a new exact algorithm for fast fully stochastic simulations of evolutionary dynamics that include birth, death, and mutation events. It produces a significant speedup compared to the direct SSA algorithm for small sub-populations. This method allows us to use large time steps during which multiple birth and death events may have occurred. However it slows down dramatically after a new mutant has been produced, since the algorithm resorts to numerical simulations in order to gain quantitative understanding of underlying dynamics.

On the most basic level, an evolutionary process is a Markov chain of discrete reactions of birth, deaths, and mutations within a population of individuals. A direct and exact way of computing individual evolutionary "trajectories" is to use the Stochastic Simulation Algorithm (SSA) (Gillespie, 1977) or its variants (Gillespie, 1976; Gibson and Bruck, 2000; La et al., 2004), in which birth, death, and mutation events are treated as Markovian "reactions." Unfortunately, for realistically large population sizes, direct stochastic simulation of even simple models becomes prohibitively expensive. Hence, there is an acute need for developing accelerated methods of stochastic simulations of evolutionary processes. Such methods usually involve approximations to the exact stochastic process based on certain small or large parameters that characterize the problem (for example, population size or mutation rates). Several approximate methods have been developed in recent years in the context of stochastic biochemical kinetics (Gillespie, 2001; Rathinam et al., 2003; Cao et al., 2005; Rathinam and El Samad, 2007; Jahnke and Altintan, 2010). Recently, Zhu et al. (Zhu et al., 2011) proposed an approximate hybrid algorithm suitable for simulation of evolutionary dynamics by combining the τ-leap algorithm (Gillespie, 2001) appropriate for abundant sub-populations that do not change their sizes much between individual events, and the direct SSA algorithm for small sub-populations. This method allows one to use large time steps during which multiple birth and death reactions may have occurred. However it slows down dramatically after a new mutant has been produced, since the algorithm resorts to numerical simulations in order to gain quantitative understanding of underlying dynamics.

1 INTRODUCTION

Natural evolution is an inherently stochastic process of population dynamics driven by mutations and selection, and the details of such evolutionary dynamics are increasingly becoming accessible via experimental investigation (Finkel and Kolter, 1999; Ruiz-Jarabo et al., 2003; Barrick et al., 2009; Pena et al., 2010; Chou et al., 2011). The importance of stochasticity comes from the fact that populations are always finite, mutations are random and rare, and at least initially, new mutations are present in small numbers. This realization prompted intensive studies of stochastic effects in evolutionary dynamics (Gillespie, 1984; Baake and Gabriel, 2000; Jain and Krug, 2007; Desai et al., 2007; Brunet et al., 2008; Hallatschek, 2011). Most of the models in these studies consider a reproducing population of individuals which are endowed with genomes that can mutate and thus affect either reproduction or death rate, as with the classical Wright-Fisher (Fisher, 1930; Wright, 1931) and Moran models (Moran, 1958) which describe a fixed population of replicating individuals. Specific models vary in the details of fitness calculation and mutation rules, but recent theoretical studies of even relatively simple models lead to non-trivial predictions on the rate of evolution as a function of the population size and the details of the fitness landscape (Tsimring et al., 1996; Kessler et al., 1997; Rouzine et al., 2003; Desai et al., 2007; Brunet et al., 2008; Hallatschek, 2011). However, the complexity of more realistic evolutionary models makes them analytically intractable and requires researchers to resort to direct numerical simulations in order to gain quantitative understanding of underlying dynamics.

* to whom correspondence should be addressed
the direct SSA for all events in which the new mutants are involved until the population of the new mutant class reaches a pre-defined threshold.

Here we develop a novel exact algorithm for simulation of evolutionary dynamics of a multi-species population undergoing asexual reproduction, death, and mutation. Unlike the direct SSA, it only samples the evolutionary process at the times of mutations. Stochastic contributions from mutation, birth, and death are included exactly, which is especially important for new species that initially contain a population size of one. We call this algorithm BNB (“Binomial-Negative Binomial”), since as the name indicates, a population update requires sampling binomial and negative binomial pseudorandom variables with specific weights. This can be done efficiently using techniques similar to those used in the next reaction method (Gibson and Bruck, 2000).

If the mutations are rare compared with other (birth and death) events, this algorithm offers a significant speed advantage with respect to the SSA. Indeed, in most organisms, the mutation rate is much smaller than the birth and death rates, e.g. the probability of mutation per division for the genome in bacteria is \( \mu_g \sim 10^{-3} \) (Drake et al., 1998). Thus, only a small (compared to the population size) number of new mutants appear in each generation. Even in viruses that generally are characterized by a high mutation rate \( \mu_g \sim 1 \), most mutations are neutral and thus do not strongly influence the population dynamics.

In the following, we begin with a general approach to the stochastic simulation of a system of reactions that are arbitrarily divided into “fast” and “slow” reactions. We then specialize to the evolutionary model in which the mutation rate is assumed to be much smaller than the birth and death rates. We present examples that illustrate the accuracy and power of the proposed algorithm for models describing evolution of a population regulated by serial dilution. Then we discuss a modification of the algorithm that allows for its use in more complex situations when the exact algorithm is not applicable. Finally, we illustrate the approximate method by a simple example of co-evolving species competing for a common nutrient source.

2 ALGORITHM

The BNB algorithm is a stochastic updating rule for the state of an evolving set of species, which are defined by their internal state (“genotype”) that in turn determines the birth, death, and mutation rates for each species. This algorithm is exact when the birth, death, and mutation rates (not the propensities!) remain constant between consecutive mutations. A single iteration of the BNB algorithm updates the state of the system to the time just prior to the next mutation.

The core of the BNB algorithm is based on an exact solution for a stochastic model of dividing, dying, and mutating discrete populations of cells. A single iteration of the BNB algorithm uses this solution to rapidly perform the following steps: (1) determine from which species and at what time a new mutant cell is generated, (2) update the population of all species to the time just prior to this mutation, (3) generate a new mutant cell that either establishes a new species in the simulation or is added to a species already contained in the simulation, and (4) update the time of the simulation to the time of this mutation.

This section contains the derivation and the detailed description of the BNB algorithm.

Stochastic simulation of a two-scale stochastic process

We consider the general case of a continuous time and discrete state stochastic system that is subject to a set of reactions among which some are “fast” and some are “slow.” We designate them as fast and slow operationally, for a given state of the system (e.g., abundances of each species) at a given time. Typically, the mean time interval between two consecutive fast reactions will be much smaller than the mean time interval between two consecutive slow reactions. The same reaction may be fast for some states of the system, but slow for other ones. Our goal is to jump directly from one slow reaction event to the next and exactly sample the state of the system at the time of slow reaction.

Let us lump all slow reactions into one that we call “mutation” and consider the dynamics of the system between two consecutive mutations. For simplicity, we assume that the propensities for each possible mutation are proportional to each other for a given system state, such that we can select the type of mutation independently of when a mutation occurs. If the probability of mutations were zero, the probability \( \rho(t) \) for being at state \( i \) at time \( t \) satisfies the master equation that only includes fast reactions

\[
\frac{d\rho_i}{dt} = \sum_j R_{ij} p_j, \quad \text{with} \quad R_{ij} = -\sum_{j \neq i} R_{ij} \quad \text{and} \quad R_{ij} \geq 0. \quad (1)
\]

Now, suppose that mutations occur with rate \( \mu_i \) at state \( i \). We can introduce the probability \( P(t) \) that the system is at state \( i \) at time \( t \) and a mutation has not yet occurred. It is easy to see that \( P(t) \) satisfies the “leaky” master equation

\[
\frac{dP_i}{dt} = \sum_j R_{ij} p_j - \mu_i P_i. \quad (2)
\]

The probability \( Y_i(t) \) that at least one mutation has occurred while the system is at state \( i \) before time \( t \) satisfies the following equation

\[
\frac{dY_i}{dt} = \mu_i P_i. \quad (3)
\]

Note that \( Y_i(t) = 0 \) at initial time \( t = 0 \). Define the probability \( P(t) \) for no mutation to have occurred by time \( t \) and \( Y(t) = \sum_i Y_i(t) \) for some mutation to have occurred at least once at any state by the time \( t \). By construction, \( Y(t) + P(t) = 1 \), and therefore

\[
\frac{dP}{dt} = -\frac{dY}{dt} = -\sum_i \mu_i P_i. \quad (4)
\]

Thus, \( P(t) \) is strictly non-increasing in time, as expected. Knowledge of \( P(t) \) allows us to sample time to the next mutation \( t_m \). We also need to know which state of the system is mutated. It is easy to show that the probability \( \rho_i(t) \) that the system is at state \( i \) at
the time of a mutation is
\[ \rho_i(t_m) = \frac{\mu_i}{\sum_i \mu_i} P_i(t_m). \]  
(5)

Thus, assuming we can solve for \( P_i(t) \), we can formulate the following algorithm for updating the stochastic system at mutation times:

**Algorithm 1**

1. Define the initial state of the system \( i_0 \), i.e. define \( P_i(0) = \delta_{i_0} \) (where \( \delta_{ij} \) is the Kronecker symbol).
2. Solve for \( P_i(t) \), which provides functions \( P(t) \) and \( \rho_i(t) \) (Eqs. 4–5).
3. Sample the next mutation time according to the cumulative probability \( P(t) \). This can be done via the inversion method, such that the next time \( t_m = P^{-1}(r) \), where \( r \) is a uniform random variable between 0 and 1.
4. Add \( t_m \) to the current time.
5. Sample the distribution \( \rho_i(t_m) \) to generate the new state \( i_m \) just before the mutation (slow reaction).
6. Choose the specific mutation according to their relative propensities and update the state of the system after the state update in Step 5.
7. Return to Step 1 until finished.

It is possible in Step 3 that the population may go extinct. In this step, \( P(0) = 1 \) but \( P(\infty) = P_0(\infty) > 0 \) when there exists a finite extinction probability, so the solution of \( P(t_m) = r \) does not exist for \( r < P_0(\infty) \). If \( r \) falls outside the range of invertibility for \( P(t) \), the population goes extinct. In this case, we can find the duration of time to extinction \( t_\infty \).

Of course, to complete this algorithm, we should be able to solve for or otherwise compute the dynamics of the probabilities \( P_i \) according to Eq. 2. While this may be difficult in general to do analytically, it may still be much simpler that solving the full system. In particular, as we discuss in the following Section, the problem can be solved exactly when the fast reactions include only birth and death while the slow reactions include only mutations.

**Generating function solution for a single-species birth/death/mutation model**

There exists a vast literature on the analysis of statistical properties of the so-called linear birth-death processes. The analytical treatments usually involve solving the corresponding master equation via the generating function method (see Bartlett (1956); Cox and Miller (1965)). Exact solutions have been found for several models including pure birth-death systems as well as systems with immigration and emigration (see, e.g., Karlin and McGregor (1958); Ismail et al. (1988); Novozhilov et al. (2006); Crawford and Suchard (2011)). Here we will follow the same general approach, but since we are interested in the statistics of mutating species, we will add the mutation “reaction” in the model which manifests itself through leakage of probability. We begin with the case of a single class of species. The number of individuals \( n \) can fluctuate due to statistically independent birth, death and mutation reactions. Birth has propensity \( g \), death has propensity \( \gamma \), and mutation has propensity \( \mu \) (as it should be if mutations occur only during replication). As before, we are only interested in the interval of time between two subsequent mutations, so the resultant state of the mutated individual is irrelevant. Thus the mutation is simply defined as the creation and subsequent departure of a single individual from the class.

Define \( P_n(t) \) to be the probability that the system is at state \( n \) at time \( t \) and that a mutation has not yet occurred. The generating function \( G(s, t) = \sum_{n=0}^{\infty} P_n(t) e^{sn} \) can be computed for an initial population \( n_0 \) at time \( t = 0 \) by (see SI for details)

\[ G(s, t) = \left[ (p_M(t) - p_E(t)) e^s G_1(s, t) + p_E(t) \right]^{n_0} \]  
(6)

with

\[ G_1(s, t) = \frac{1 - p_B(t)}{1 - p_B(t)e^s}, \]  
(7)

\[ p_M(t) = \frac{RC(t) + 2\gamma S(t) - WS(t)}{RC(t) - 2\gamma S(t) + WS(t)}, \]  
(8)

\[ p_E(t) = \frac{\gamma (1 - p_M(t))}{W - \gamma - g p_M(t)}, \]  
(9)

\[ p_B(t) = \frac{g p_E(t)}{\gamma}. \]  
(10)

\[ R \equiv \sqrt{(g - \gamma)^2 + (2g + 2\gamma + \mu)^2} \]  
(11)

\[ W = g + \gamma + \mu. \]  
(12)

When Eq. 12 is not satisfied, this indicates that the population will go extinct before a mutation occurs if the population is unperturbed for infinite time.

**Binomial – Negative Binomial expansion**

After computing the time to the next mutation, we need to generate a sample number of individuals at the time of mutation. The number of individuals conditional on no mutation at time \( t \) is distributed according to the generating function \( G(s, t) \) given by Eq. 6. Here we show that this seemingly complicated distribution can be exactly sampled by drawing two random numbers - one binomial, and one negative binomial. Many popular software packages, e.g. (Press et al., 2007), contain fast algorithms for generating these random numbers (note that negative binomials can be generated by Poisson random variates with a Gamma-distributed parameter).

**Equation 6** can be recast via a binomial expansion

\[ G(s, t) = p_M(t)^{n_0} \sum_{m=0}^{n_0} \frac{n_0!}{m! (n_0 - m)!} G_1(s, t)^m e^{ms} \]

\[ \cdot \left( 1 - \frac{p_E(t)}{p_M(t)} \right)^m \frac{p_E(t)}{p_M(t)}^{n_0-m}. \]  
(13)
be interpreted as a generating function of a process in which the system either has mutated by time \( t \) with probability \( 1 - p_M(t) \)^{\infty n}, or if the system hasn’t yet mutated, then it is in a state \( \tilde{n} \) whose distribution is a binomial superposition of \( n_0 \) negative binomial distributions. While Eq. 13 does not directly provide the probability to be in a particular state at the time of a mutation, it provides the probability \( P_a(t) \) at an arbitrary time \( t \) conditional on no mutation. We can then generate a sample of the population \( \tilde{n} \) conditional on no mutation at time \( t \) by the following procedure.

**Algorithm 2**

1. Generate a binomial random number \( \tilde{n} \), with success probability \( 1 - (p_E(t)/p_M(t)) \) and \( n_0 \) terms.
2. If \( \tilde{n} = 0 \), then the system at time \( t \) is in the extinct state \( \tilde{n} = 0 \).
3. Otherwise, generate the new state variable \( \tilde{n} \): \( \tilde{n} = \tilde{n} + NB(\tilde{n}, p_E(t)) \), where \( NB(\tilde{n}, p_E(t)) \) is a negative binomial number of order \( \tilde{n} \) and probability of success \( p_E(t) \).

We are also interested in the probability \( \rho_m(t) \) for a system to be in the state \( n \) at the mutation time. It is easy to see that \( \rho_m(t) \propto \mu \cdot \rho_N(t) \propto n \cdot P_a(t) \) (see Eq. 5). To compute these probabilities, we introduce the corresponding generating function \( G_\rho(s, t) = \sum_0^\infty \rho_n(t) e^{sn} \). After straightforward algebra, we obtain from Eq. 6

\[
G_\rho(s, t) = \left( \frac{p_M(t) - p_E(t)}{p_M(t)} G_1(s, t) + p_E(t) \right)^{n_0 - 1} \cdot e^{s G_1(s, t) + 2}. \tag{14}
\]

which has the binomial expansion

\[
G_\rho(s, t) = \sum_{m=0}^{n_0 - 1} \frac{n_0!}{m!(n_0 - m)!} G_1(s, t)^{m+2} \left( 1 - \frac{p_E(t)}{p_M(t)} \right)^m \cdot e^{(m+1)s} \left( \frac{p_E(t)}{p_M(t)} \right)^{n_0 - 1 - m}. \tag{15}
\]

Equation 15 has the same form as Eq. 13, and thus, \( \rho_m \) can be also sampled. Specifically, the algorithm for computing the state of the system just before the next mutation (at time \( t_m \)) for the single species reads as follows.

**Algorithm 3**

1. Generate a binomial random number \( \tilde{n} \), with success probability \( 1 - (p_E(t_m)/p_M(t_m)) \) and \( n_0 - 1 \) terms.
2. Generate the updated state \( \tilde{n} \) at the mutation time: \( \tilde{n} = \tilde{n} + 1 + NB(\tilde{n} + 2, p_M(t_m)) \), where \( NB(\tilde{n}, p_M(t)) \) is a negative binomial number of order \( \tilde{n} \) and probability of success \( p_M(t) \).

Note that the system will never be in the extinct state, which reflects that an extinct population cannot mutate.

**Simulating multiple co-evolving species: first mutation method**

In this section we return to the original problem of an evolving population of multiple species. We enumerate species by index \( i \), with \( n_i(t) \) individuals in each species. We are interested in sampling the set \{ \( n_i(t_m) \) \} at mutation times \( t_m \). We assume that the system parameters (birth, death, and mutation rates) do not change between mutations unless the algorithm is ended early between two mutations. At the time of mutation, one individual is created from mutating class \( i_m \) and, depending on the type of mutation, is either added to one of the other existing classes (if such a class already exists) or becomes the founding member of a new class.

The algorithm for generating a sample stochastic evolution trajectory, which we call First Mutation BNB, is as follows.

**Algorithm 4**

1. Initialize the system with \( N \) classes of species at time \( t = 0 \). Specify populations of all classes \( n_i, i = 1, ..., N \). Each class has its own set of birth, death, and mutation rates \( g_i, \gamma_i, \mu_i \).
2. For each class, generate \( N \) random numbers \( r_i \), uniformly distributed between 0 and 1. For each \( i = 1, ..., N \), generate a time \( t_i \) to the next mutation by Eq. 11. When Eq. 12 is not satisfied, set \( t_i = \infty \).
3. Find the minimum time \( t_m = \min(t_i) \) and the corresponding class \( i_m \). Update the time \( t \to t + t_m \).
4. Update the population for the mutated class \( i_m \) using two random numbers (one binomial and another negative binomial) according to the Algorithm 3.
5. Update the populations of all other classes according to their distributions \( P_i(t_m) \) using Algorithm 2.
6. Select the specific mutation that occurs. If the mutation generates a member of a nonexistent class, create a new class \( N + 1 \) with \( n_{N+1} = 1 \) and its own set of parameters \( g_{N+1}, \gamma_{N+1}, \mu_{N+1} \). Otherwise, add 1 to the corresponding existing class.
7. One or several of the non-mutated classes may have zero population and are thus extinct. Remove extinct classes from the list and reduce the number \( N \) of classes accordingly.
8. Return to Step 2 until the algorithm has completed.

To end the algorithm at a specific time rather than at a mutation event, all populations can be updated according to Algorithm 2 with the time duration \( t' - t \), where \( t \) is the current time, and \( t' \) is the prescribed end time. This update would be done just after Step 2 if \( t' < \min(t_i) \) first occurs. Ending at a specific time is useful for a number of purposes, such as if the population is reported or modified at fixed time intervals, or if rates are adjusted at fixed time intervals.

The Algorithm 4 is analogous to the First Reaction Method used for stochastic simulation of reaction networks (Gillespie, 1976), in that the simulation of a system with \( N \) classes of co-evolving species generates \( 3N \) random numbers in order to step to the next mutation. This algorithm can thus become inefficient as the number of classes becomes large. To remedy this shortcoming, an optimized and only slightly more complex version of this algorithm is presented in the next section.
Simulating multiple co-evolving species: next mutation method

In fact, the number of random variables generated for each mutation in Algorithm 4 is excessive. Different species evolve independently between mutations, and even at the mutation time, only two classes are coupled, due to the mutating population generating and then contributing a single member to another species class. If this mutational coupling did not exist, the dynamics of species would be statistically independent at all times, and we could simulate all species independently using only 3 random numbers per mutation event.

This line of reasoning leads to a similar but optimized algorithm (see SI for the algorithm and further justification), where the populations and next mutation times of species are resampled only for the two species that are coupled via a mutation event, while population sizes and next mutation times of all other classes are not resampled. Validity of the algorithm hinges on the statistical independence of species that are uncoupled by a mutation. The method is analogous to the Next Reaction Method (Gibson and Bruck, 2000), so we label the algorithm Next Mutation BNB.

The optimized scheme reduces the typical number of new random variables required per mutation to only 6 after the first iteration, independently of the total number of classes $N$. Only initialization and finalization of the algorithm have a computational cost of order $N$, so efficiency of the algorithm primarily depends on how frequently the algorithm is restarted, as is the case whenever the whole population is sampled for observation.

The Next Mutation BNB algorithm is always as fast or faster than the First Mutation BNB. We thus use Next Mutation BNB (or just BNB) exclusively for the simulation examples of this paper.

Approximate simulation method using BNB

One major benefit of the BNB algorithm is that binomials and negative binomials rapidly generate an update for the evolving system with linear propensities for birth, death, and mutation in a non-interacting population. While this situation is typically the case for cells kept in log-phase growth, the cases when species are interacting or when propensities deviate from a linear law are also of interest. Because of this, we outline how the BNB algorithm can be adapted to approximately, but accurately, simulate more complicated systems.

The basis of the BNB algorithm is the generating function solution Eq. 6, and it is straightforward to show from the short time form of this generating function that the BNB algorithm applied for sufficiently short time increments, during which birth, death, and mutation rates are considered constant, can simulate systems with population-dependent rates. Between BNB updates, all of these rates can be updated in a state-dependent manner. This approach is similar to the $\tau$-leap approximation to stochastic systems, which is often used to accelerate simulations of chemical reaction networks (Gillespie, 2001). The basis of $\tau$-leap is that the propensities for reactions can be considered approximately constant during some time interval, such that the update scheme for $\tau$-leap assumes each reaction occurs a Poisson-distributed number of times. Simulation error magnitude in $\tau$-leap is closely associated with how well propensities are kept constant during a given time interval, and based on this connection, a few prescriptions for the step size have been suggested (Gillespie and Petzold, 2003; Cao et al., 2006, 2007). In contrast, BNB as an approximate updating scheme assumes that the propensities are approximately linear with respect to population, i.e. having constant rates. Deviation from the linear law is the primary factor influencing simulation error in BNB updating.

An important aspect of an approximate BNB updating method is that large and small species populations are treated uniformly, such that the same updating scheme applies to both situations with equal speed and relative accuracy. This may be contrasted to $\tau$-leap methods, which due to large relative fluctuations of the propensity for small populations are no longer valid except for very short time steps. Zhu et al. (2011) introduced a hybrid $\tau$-leap method which simulates species lower than a given population (the “cutoff”) using direct Gillespie algorithm. The tradeoff for the increased accuracy is a much-increased workload, since Gillespie algorithm simulates each reaction event individually. New species, which start as single cells, or species that naturally exist in low abundances are especially susceptible to an increase in workload for finite cutoff.

3 RESULTS

Exact simulations

In this section we will apply the BNB algorithm to examples that can be exactly simulated using BNB. These examples deal with modeling the evolution of heterogeneous cell populations in a hypothetical bioreactor designed to maintain exponentially growing cultures. We illustrate several phenomena that have been explored previously in analogous situations, e.g. for populations of fixed size, though we pursue these phenomena in the regime where large fluctuations in total population size (10-fold in most of our simulations) are routine.

The following models assume that cells are kept sufficiently dilute in culture such that limiting nutrients and other cell-cell interactions are not a factor. These cells thus grow and divide freely. The bioreactor prevents cell cultures from growing too dense by measuring the population size periodically (after every time duration $\Delta t$) and diluting the culture by binomial sampling to the mean population size $n_{\text{min}}$ once the population has exceeded the population size $n_{\text{max}}$. In the simulations, we advance time directly from one mutation to the next or until the system has evolved longer than the maximal time duration $\Delta t$, at which point cells may be diluted if the population has exceeded $n_{\text{max}}$. It is also straightforward to simulate a bioreactor that continuously dilutes cultures to stem population growth, where the rate of media turnover and, correspondingly, cell "death" is controlled, but we do not consider such an case here. An analysis in the SI demonstrates that performance of BNB for these situations can far exceed that for direct Gillespie and $\tau$-leap methods.

Abrupt dilution events can greatly enhance the effect of stochasticity, since there is a corresponding reduction in genetic diversity associated with each subsampling of the population. The smaller population after a dilution event will be heavily influenced by the particular individuals retained, leading to a form of the founder effect (Templeton, 1980). Even in light of this fact, we show that many phenomena found for fixed population sizes, e.g. wave behavior for population fitness, also occur using a dilution protocol that might occur experimentally.
Fig. 1. Simulations of the linear fitness model with \( \Delta t = 0.1, \epsilon = 10^{-3}, \eta = 10^{-3}, n_{\text{min}} = n_{\text{max}}/10 \). The instantaneous distributions of the populations over the species index normalized by \( n_{\text{max}} \) as a function of time are shown for \( n_{\text{max}} = 10^4 \) (A) and \( n_{\text{max}} = 10^5 \) (B). Wave-like behavior is evident in both cases, though the smaller population leads to a noisier and slower wave. Panels C and D show the corresponding probabilities averaged over 800 realizations. The wave velocity, by a least squares linear fit to the ensemble mean fitness, is \( 0.95 \times 10^{-3} \) and \( 2.1 \times 10^{-3} \) indices per unit time for (C) and (D), respectively.

Fig. 2. (A) The wave velocity (indices per unit time) of the linear fitness system has a slow (logarithmic) dependence on the population size set by \( n_{\text{max}} \), in agreement with theoretical results (Tsimring et al., 1996; Kessler et al., 1997; Rouzine et al., 2003; Desai et al., 2007; Brunet et al., 2008; Hallatschek, 2011) (parameters are the same as in Fig. 1). Blue dots represent individual velocity measurements based on least squares fitting of a line to the last half of the mean index trajectory. Red line shows the least squares fit of the velocity as a linear function of \( \log n_{\text{max}} \) over the range \( n_{\text{max}} \geq 10^4 \). The velocities from Fig. 1C and Fig. 1D are plotted as green squares and diamonds, respectively. (B) Same as (A) for \( \epsilon = 10^{-4} \) and \( \eta = 10^{-2} \). The weaker fitness gradient leads to a noisier distribution of velocities.

Fig. 3. Ruggedness of the fitness landscape impacts speed of evolution in a linear fitness model. Shown are apparent wave velocities (blue dots) derived by least-square fitting of the mean index \( \langle m \rangle \) across species as a function of time. The model with deterministic alternating fitness and \( n_{\text{max}} = 10^4 \) (A) or \( n_{\text{max}} = 10^5 \) (B) leads to a smooth decay of wave velocity with respect to the perturbation amplitude \( \nu \). In contrast, a model with quenched disorder in fitness and \( n_{\text{max}} = 10^4 \) (C) or \( n_{\text{max}} = 10^5 \) (D) exhibits an abrupt decrease in wave velocity suggesting a phase transition. In all cases, \( n_{\text{min}} = n_{\text{max}}/10, \eta = 10^{-3}, \epsilon = 10^{-2}, \gamma_m = 0.1, \) and \( \Delta t = 0.02 \). The red curve indicates trend lines generated by a Savitzky-Golay filter.

**Linear fitness model** Suppose that species are characterized by a positive integer index \( m \) that is a measure of fitness. Birth rate \( g_m \) is a linear function of \( m \), \( g_m = \gamma (m - 1) \). Death rate \( \gamma_m \) is constant across species. Mutation rate is proportional to growth rate (faster growing species also mutate faster), \( \mu_m = \eta g_m \). During a mutation of species with index \( m \), a new member of species with index \( m - 1 \) or \( m + 1 \) is created, as chosen uniformly at random. If a species with index \( m = 1 \) mutates, a new member of the species with index 2 is always created.

It has been demonstrated for \( \epsilon > 0 \) in the case of a constant total population that evolution on a linear fitness landscape leads to traveling population waves (Tsimring et al., 1996; Kessler et al., 1997; Rouzine et al., 2003; Desai et al., 2007; Brunet et al., 2008; Hallatschek, 2011), such that the mean fitness of the population linearly grows in time. However, the finite-size stochastic system can only be treated heuristically (Tsimring et al., 1996; Kessler et al., 1997), asymptotically (Rouzine et al., 2003; Desai et al., 2007; Brunet et al., 2008), or under certain specific modeling assumptions (Hallatschek, 2011). Thus, exact numerical simulations of large evolving populations in linear fitness landscapes are useful for testing the existing theories. Simulations indeed produce wave-like behavior (see Fig. 1). The wave velocity scales linearly with the logarithm of the population size, as predicted (see Fig. 2).

We used similar simulations to study the effects of quenched fitness fluctuations on the propagation of traveling evolution waves. This problem is qualitatively analogous to the models of transport in systems with quenched disorder that are known to exhibit phase transitions (Bouchaud et al., 1990; Monthus and Bouchaud, 1996), and we expect similar behavior for evolution in a linear model with quenched disorder in the growth rate law. We assumed that the fitness as a function of the species index \( m \) has a fluctuating piece in addition to the linear dependence. Specifically, we consider growth rates that vary as \( g_m^{(\nu)} = 1 + \epsilon (m - 1 + \nu R_m) \), where \( \nu \geq 0 \) provides the scaling of noise, and \( R_m \in [-0.5, 0.5] \) are
an exponentially large time. This case can be contrasted against a fitness, and these barriers when they exist can trap the system for possible to form rare but wide barriers due to fluctuations in the

The case with \( \nu > 1 + m \) leads to an increase in growth rate, and wave propagation should proceed but with moderately reduced velocity. The case with \( \nu > 1 \) is qualitatively different, since an increase in \( m \) need not imply an increase in fitness. In this regime, it is possible to form rare but wide barriers due to fluctuations in the fitness, and these barriers when they exist can trap the system for an exponentially large time. This case can be contrasted against a potential with similar but deterministic barriers, which for \( \nu > 1 \) has fitness barriers only a single species wide. Figure 3 shows that quenched disorder exhibits substantially different behavior than the case when fitness contains regular variation. The system with quenched disorder in particular exhibits a sharp decrease in wave velocity as disorder is increased to \( \nu > 1 \), akin to a phase transition.

Fig. 4. Wave behavior for the evolution in a model with competition, simulated using BNB as an approximate algorithm with time step \( \tau = 1 \). (A) A single realization of the species distribution as a function of time for initial population 100. (B) The mean population distribution for an ensemble of 800 simulations.

**NK model simulations** Due to the general way the BNB algorithm treats mutations, it can be applied to more complicated evolutionary models. We used a variant of the NK model (Kauffman and Levin, 1987) to simulate evolution on fitness landscapes with various degrees of ruggedness. Despite large fluctuations in population, we could reproduce classical results for NK models, including state-dependent wave speed for smooth fitness landscapes, and punctuated evolution for rugged landscapes. Results and analysis of this model are found in the SI.

**BNB as an approximate algorithm: evolution in nutrient-limited environments**

BNB can also be applied as an approximate algorithm for systems with state-dependent growth rates. Propensities may deviate from the linear law assumed in the BNB algorithm, but the BNB algorithm may still approximate a system with non-constant birth, death, and mutation rates by evolving the system with a BNB step restricted to a short duration \( \ell \). Rates are then updated using the new populations before integrating the system with another BNB step, and so on. Validity of this process depends on self-consistency of the assumptions in the BNB algorithm, especially that propensities for reactions are independent of other species and proportional to population. See SI for details.

We checked performance of this approximate algorithm for a system in which several species compete for a common nutrient that is supplied at a constant rate. Different species can consume this nutrient with different effectively, which provides selective pressure. Specifically, we suppose a linear fitness model for species, \( g_m = a_m (1 + \sum_r a_{r} n_{r} / K_{0})^{-1} \), with species index \( m \), \( a_{r} = g_m \), \( a_{m} = 1 + \epsilon (m - 1) \), with species index \( m \), \( \eta = 10^{-3} \), and a scaling factor \( \epsilon = 1 \). In contrast to the other simulations in this text, birth rates are coupled in such a way that the total population in the system autonomously relaxes on average to a fixed value \( \bar{n} \approx 10 K_{0} \) without the need of dilution events triggered by the population. The evolution of the system is linked to the ratio of growth rates \( g_{m1}/g_{m2} = a_{m1}/a_{m2} \), which indicates that species with a higher index \( m \) tend to grow faster than those with lower index. Due to this effect, the system exhibits wave-like behavior (see Fig. 4).

The recurrent creation and subsequent growth of new species in the competition model suggests that BNB could maintain better accuracy than \( \tau \)-leaping schemes, since BNB faithfully simulates arbitrarily small populations and also exponential growth. We tested this for short-time simulations, and we found that in this context that BNB can provide consistently increased accuracy when compared to a hybrid \( \tau \)-leap algorithm (see Fig. 5).

Fig. 5. Simulation accuracy for the model with competition. Using BNB (red), hybrid \( \tau \)-leap (dashed blue), or direct SSA (light green), the model with \( K_{0} = 1000 \) and initial population \( 100 \) was simulated over \( 10^{5} \) realizations. As a measure of error, statistics of the population of the first mutant (index=2) were examined at \( T = 50 \). (A) The histogram (bin width= 250) of this population for simulations using step size \( \tau = 5 \). BNB matches direct simulation closely, while hybrid \( \tau \)-leap with cutoff 10 suffers from major inaccuracies. (B) \( L^{2} \) error between the histogram of direct SSA simulation and that of either BNB or the hybrid \( \tau \)-leap normalized by the minimal expected statistical deviation, see SI for details. (C) Same as (B), but as a function of the cutoff value for the hybrid \( \tau \)-leap algorithm with \( \tau = 5 \). (D) Mean workload of the hybrid \( \tau \)-leap and the approximate BNB algorithms, normalized by the workload for the BNB algorithm, as a function of the cutoff value.
4 DISCUSSION

In this paper, we have proposed an algorithm, which can be used to sample exactly co-evolving species that do not interact between mutations, and faithfully approximate the evolution of weakly-interacting species. BNB algorithm not only accounts for the stochastic fluctuations that arise due to the random nature of genetic mutations, but also accounts for the small-number fluctuations due to the growth of new species that are spawned as single cells. Each iteration of the BNB algorithm generates the time of the next mutation and the abundances of all species just after the mutation. This algorithm is exact when the birth, death, and mutation rates do not change between consecutive mutations. Although similar in spirit to approximate leaping schemes developed for modeling stiff stochastic chemical kinetics (Gillespie, 2001; Rathinam et al., 2003; Cao et al., 2005; Rathinam and El Samad, 2007; Jahnke and Altintan, 2010; Zhu et al., 2011), it differs significantly by providing an exact sampling at (irregular) intervals corresponding to mutational events. The method yields a substantial speed advantage over a straightforward stochastic simulation algorithm when the mutations are rare compared with birth and death events. The method is accessible, since the central part in implementing BNB is constructing fast methods that generate binomial and negative binomial pseudorandom numbers, both of which are available in standard code libraries (Press et al., 2007). More generally, the BNB algorithm is applicable to the simulations of systems in which underlying reactions are all first order and their rates remain unchanged between coarse-grained simulation steps.

Using the exact BNB algorithm, we simulated several evolution models for a hypothetical bioreactor that performs abrupt dilutions of cell culture when the total cell population exceeds a prescribed value. An analogous experimental bioreactor would periodically reduce the total number of cells, replenish nutrients, and remove wastes in order to maintain log-phase growth of bacterial populations. In contrast to the classical theoretical setting, where the total number of cells is often kept constant, our model bioreactor maintained periodic 10-fold variations in the total number of cells. Despite these wild fluctuations in total population size, most phenomena and population size scaling were preserved. We found the classical scaling laws of adaptation velocity with the population size, as well as the evidence of a phase transition in the case of rugged linear models.

Real cell cultures almost always exhibit some degree of interaction within and among species, and so we showed how the BNB algorithm can also be extended to an approximate algorithm that is competitive with τ-leap and hybrid schemes adapted for evolutionary dynamics simulations (see, for example, (Zhu et al., 2011)). A practical advantage of the approximate BNB algorithm is its uniformity; a BNB step is implemented with identical code for all population sizes. A specific model for species competing for common nutrients was introduced to test BNB, and BNB was found to readily provide good accuracy with minimal workload when compared to analogous τ-leap simulations. We anticipate the advantage of BNB to be maintained in the case where simulations require accurate and fast simulation of exponential growth of species that routinely are found at low population counts, as is the case when new fitter species grow to overtake the population. It should be noted, however, that even though the BNB algorithm can be used to simulate rather general systems, there are systems where BNB performs comparably to or even worse than τ-leap.

The present work presents the foundation for the BNB algorithm, but there exist several immediate directions for future refinement. We anticipate that simple modification of the BNB algorithm should enhance the accuracy for a wide variety of models with interacting species, analogously to a proposed midpoint method for τ-leaping (Anderson et al., 2010). Similarly straightforward modifications may also lead to a BNB formalism that approximates time-dependent birth, death, and mutation rates, as needed for externally driven metabolic networks, e.g. the GAL network (Bennett et al., 2008). A less trivial extension would be to remove the assumption that birth, death, and mutation rates are constant across species. Experimentally, cells within a common species exhibit variability in their cellular state (Elowitz et al., 2002), which could lead to a distribution of growth rates within a single species. Such a modified BNB could then be useful for answering questions concerning how species evolution couples to cellular state.

ACKNOWLEDGEMENT

Funding: This work was supported by the National Institutes of Health, grants P50GM085764 [WM]; RO1GM069811 [JH]; R01GM089976 [LT].

REFERENCES
