A Fast Solver for Modeling the Evolution of Virus Populations

Gerhard Niederbrucker
University of Vienna,
Research Group Theory and
Applications of Algorithms
gerhard.niederbrucker@univie.ac.at

Wilfried N. Gansterer*
University of Vienna,
Research Group Theory and
Applications of Algorithms
wilfried.gansterer@univie.ac.at

ABSTRACT

Solving Eigen's quasispecies model for the evolution of virus populations involves the computation of the dominant eigenvector of a matrix whose size N grows exponentially with the chain length of the virus to be modeled. Most biologically interesting chain lengths are so far well beyond the reach of existing algorithms and hardware.

We show how to exploit the special properties of the problem under consideration and design a fast and accurate solver which reduces the complexity to $\Theta(N \log_2 N)$. Our solver is even faster than existing approximative strategies and contrary to those it can also be applied to more general formulations of the quasispecies model. Substantial further improvements and high parallelism can be achieved for special fitness landscapes in the evolution model.

Beyond theoretical analysis, we evaluate the performance of our new solver experimentally on a GPU with an OpenCL implementation and illustrate that it achieves speedup factors of more than 10^7 over standard approaches.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*Computations on matrices*; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra—*eigenvalues and eigenvectors (direct and iterative methods); sparse, structured, and very large systems (direct and iterative methods)*; J.3 [**Life and Medical Sciences**]: Biology and genetics

General Terms

Algorithms, Performance

Keywords

evolution models for virus populations, quasispecies, GPU computing, large eigenproblems

*Corresponding author.

SC11, November 12-18, 2011, Seattle, Washington, USA

1. INTRODUCTION

Under some assumptions on the environment the evolution and the long-term behavior of a virus population can be modeled by the *quasispecies model* [5]. In this model, each virus is represented by an RNA molecule and each RNA molecule is represented as a string over a finite alphabet with a fixed length ν (the so called *chain length*). In this paper, we consider a binary alphabet. The probability of a single point mutation in an RNA sequence is modelled by the *error rate* p satisfying $0 < p \leq 1/2$. Given a fixed chain length ν, all $N := 2^\nu$ possible RNA sequences have to be considered since any RNA molecule can potentially mutate into any other one (although the overall probability for some of these mutations may be very low).

The overall goal is to numerically compute the relative concentrations of the so-called *quasispecies* [7] which describe an ordered stationary distribution in the model. At the core of the effort summarized in this paper is the numerical computation of the eigenvector corresponding to the largest eigenvalue of a non-negative matrix $W \in \mathbb{R}^{N \times N}$ which basically describes the constitution and the mutation probabilities of the N different RNA molecules with chain length ν (see Section 1.1). This eigenvector contains the information about the concentration of the quasispecies in the course of the evolution of the virus population. Since the dimension N of this eigenvalue problem grows *exponentially* with the chain length ν of the RNA molecules to be modelled, numerical solutions of the model were so far restricted to very small chain lengths and most biologically interesting cases were out of scope of computational methods, even on high-end supercomputers. In this paper, we show how to derive very fast algorithms for solving this eigenvalue problem by exploiting the specific problem properties and structure. Beyond a fast algorithm for the general case we show how to achieve further improvements for special cases of the fitness landscape. In combination with a hardware-efficient implementation on GPUs we are able to significantly extend the range of computationally tractable problem instances.

We pursue a similar approach as in [10], where we investigated the power iteration method based on an implicit matrix vector product for the problem at hand. In this paper, we go a big step further and show that the matrix vector product to be performed can be expressed as a transformation with similar properties as the FFT. This implies that there is no need to store any element of the matrix and that we have an $\Theta(N \log_2 N)$ algorithm for directly performing the required matrix vector product without additional stor-

age requirements.

By definition our algorithms enable equally fast computations for more general formulations of the quasispecies model than the ones usually considered. In particular, the assumption of a *uniform* error rate, which goes back to the origins of the quasispecies model [5], can easily be replaced by far more general error rate models. Additionally, we also present results about the efficient solution of models with specially structured fitness landscapes. Overall, the new algorithms discussed compute fully accurate solutions of the problem at a cost which is even lower than the one of approximative approaches considered earlier. Moreover, they exhibit a high degree of parallelism and are consequently suitable for exploiting the characteristics of modern GPU hardware.

Synopsis.

The remainder of Section 1 summarizes the background of the quasispecies model for the evolution of virus populations, the resulting eigenvalue problem, and related work on solving this problem. The derivation of fast implicit matrix vector operations for this problem is topic of Section 2. In Section 3, solution methods for the occurring eigenvalue problem are developed. Section 4 discusses efficient GPU implementations and summarizes experimental performance evaluations. Finally, in Section 5 it is shown how further performance improvements can be achieved if additional structural properties are present.

1.1 Biochemical Background

We briefly review the model which leads to the eigenvalue problem we are dealing with. For a more detailed discussion and further references see [13, 15].

As already mentioned, each RNA molecule is represented over a binary alphabet. In particular, X_i with $0 \leq i < N$[1] denotes the RNA molecule and the corresponding species which is represented by the binary encoding $(b_0, \ldots, b_{\nu-1})$ of length ν of the integer i. The error-free sequence X_0 associated with the integer 0 is called the *master sequence*. As a distance measure between species X_i and X_j we use the Hamming distance $d_H(X_i, X_j)$ which represents the minimal number of elementwise mutations required to transform X_i into X_j.

Due to Eigen [5] the following system of ODEs models the evolution of RNA molecules for $i = 0, \ldots, N-1$:

$$\begin{aligned} \frac{dx_i}{dt} &= \sum_{j=0}^{N-1} f_j \cdot Q_{i,j} \cdot x_j(t) - x_i(t) \cdot \Phi(t) \qquad (1) \\ \Phi(t) &= \sum_{j=0}^{N-1} f_j \cdot x_j(t), \quad \sum_{j=0}^{N-1} x_j(t) = 1. \end{aligned}$$

In this model, x_i denotes the relative concentration of the molecular species X_i (initially, $x_0 = 1$), the positive fitness value f_i describes the constitution of the molecular species X_i, and the (i, j) entry of the mutation matrix Q is the probability that sequence X_i mutates into sequence X_j:

$$Q_{i,j} = p^{d_H(X_i,X_j)} \cdot (1-p)^{\nu - d_H(X_i,X_j)}. \qquad (2)$$

By definition, the values $Q_{i,j}$ depend only on $d_H(X_i, X_j)$ and therefore the entire matrix Q contains only $\nu + 1$ different values. It should be pointed out that this standard model assumes that p is an *average* error rate over all possible mutations since it does not depend on the position or on the overall number of the mutations. Equation (2) also shows that an index i (row or column) corresponds to the sequence X_i. It would be possible to reorder the sequences by applying a permutation π such that index i corresponds to sequence $X_{\pi(i)}$.[2]

The ODE system (1) is a Bernoulli system, and thus a proper change of variables leads to a linear system with constant coefficients $\dot{z} = W \cdot z$ with $W = Q \cdot F$ where Q is defined by Equation (2) and F, the *fitness landscape*, is a diagonal matrix with the fitness values $f_i > 0$ along the diagonal. The search for the quasispecies then reduces to the computation of the eigenvector corresponding to the dominating eigenvalue of W [14]. In fact, there are several mathematically equivalent formulations of the problem with slightly differing structure:

$$\begin{aligned} Q \cdot F \cdot x_R &= \lambda \cdot x_R \qquad (3) \\ F^{\frac{1}{2}} \cdot Q \cdot F^{\frac{1}{2}} \cdot x_S &= \lambda \cdot x_S \qquad (4) \\ F \cdot Q \cdot x_L &= \lambda \cdot x_L \qquad (5) \end{aligned}$$

Since F is diagonal, their solutions can easily be transformed into each other:

$$x_R = F^{-\frac{1}{2}} \cdot x_S, \quad x_S = F^{-\frac{1}{2}} \cdot x_L, \quad x_R = F^{-1} \cdot x_L.$$

Note that in the special case where all values in F are equal the problem reduces to the computation of the dominating eigenvector of a bistochastic matrix, which is trivial and leads to an eigenvector where all entries are equal. This is not at all surprising since for equally fit sequences we clearly expect the uniform distribution as result.

Since the x_i represent relative concentrations, we are only interested in solutions where all components of the computed eigenvector are non-negative (negative concentrations do not have a physically meaningful interpretation). W satisfies the conditions of the Perron-Frobenius theorem [16], and thus this nonnegativity property is guaranteed. Based on the computed eigenvector with the relative concentrations for each sequence, one can compute cumulative concentrations of so-called *error classes*. The error class $\Gamma_{k,i}$ contains all sequences j which have a fixed Hamming distance k from the fixed sequence i:

$$\Gamma_{k,i} := \{j \mid 0 \leq j < N \wedge d_H(X_i, X_j) = k\} \qquad (6)$$

Since the error classes with respect to the master sequence are particularly relevant, we denote $\Gamma_k := \Gamma_{k,0}$. Γ_k contains $\binom{\nu}{k}$ sequences. In the style of the definition of the error classes we denote the $\nu+1$ different values of Q by Q_{Γ_k}, i. e., $Q_{\Gamma_k} := p^k \cdot (1-p)^{\nu-k}$ for $0 \leq k \leq \nu$. When the meaning is clear from the context we also use just i instead of X_i for denoting the binary string corresponding to the integer i.

After the dominating eigenvector of W has been computed, the cumulative concentrations $[\Gamma_k] := \sum_{j \in \Gamma_k} x_j$ of the error classes Γ_k in the stationary distribution are obtained. Plotting these cumulative concentrations for different error rates p leads to curves as the ones shown in

[1] Throughout the entire paper, we use zero-based indexing for vectors and matrices.

[2] For example, using the Gray code as permutation would deliver a matrix Q where the first diagonal above and below the main diagonal are constant. This comes from the basic definition of the Gray code which states that $d_H(X_i, X_{i+1}) = 1$ for all i.

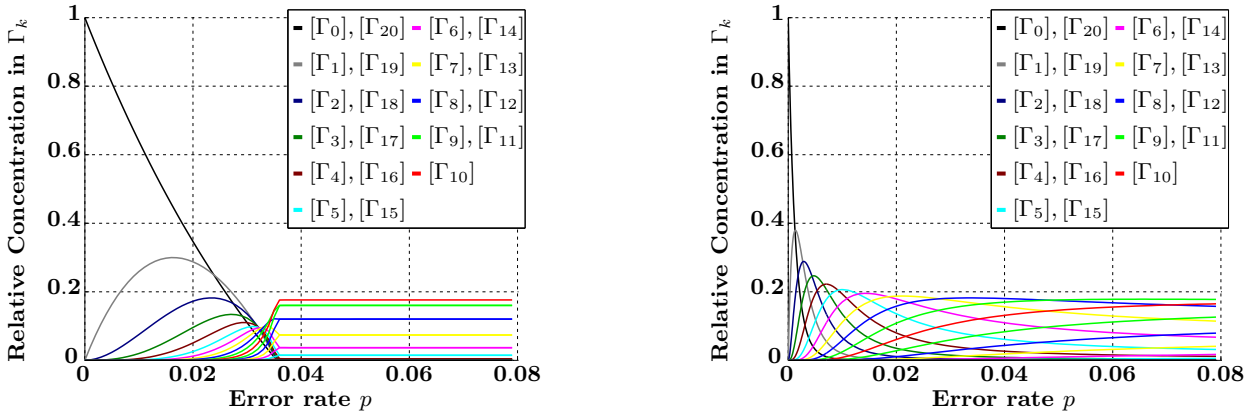

Figure 1: A visualization of the error threshold phenomenon is shown on the left for $\nu = 20$ and the simple *single peak fitness landscape* with $f_0 = 2$, $f_i = 1$ for all $1 \leq i < N$. An ordered stationary distribution results up to $p_{\max} \approx 0.035$, and for $p > p_{\max}$ a sudden change to the uniform distribution of all sequences occurs. Error classes with the same number of elements (Γ_k and $\Gamma_{\nu-k}$) are shown in the same color and therefore their curves meet when the uniform distribution is reached at the error threshold. On the right, the behavior for $\nu = 20$ and the so-called *linear landscape* defined as $f_i = f_0 - (f_0 - f_\nu) \cdot d_H(i,0)/\nu$ for all $0 \leq i < N$ with $f_0 = 2$ and $f_\nu = 1$ are shown. For this landscape a smooth transition into the uniform distribution is observed and no error threshold phenomenon occurs.

Figure 1. Such figures can be used to get a better understanding of how a certain fitness landscape influences the evolution of the virus population. They would be even more interesting at the level of granularity of single sequences but they are very rare in the literature due to the limitations in chain lengths which can be handled computationally. Depending on the concrete fitness values, the *error threshold phenomenon* may occur or not. If it occurs there is an ordered stationary distribution of the concentrations up to a critical value $p_{\max}$ for the error rate p, where some sequences clearly dominate, whereas others do not appear at all or only in very low concentrations. For $p > p_{\max}$, the structure of the population changes immediately into a uniform stationary distribution where all sequences occur in the same concentration, which is equivalent to random replication. Note that in Figure 1 *cumulative* concentrations are shown for the error classes. Although all sequences have the same concentration for $p > p_{\max}$, the cumulative concentrations of the error classes differ because their cardinality differs. Typical values for $p_{\max}$ on certain fitness landscapes are in the range $0.01 - 0.1$ [14, 15], depending on the concrete fitness values and the chain length. Such small values for $p_{\max}$ are quite surprising since random replication as exact solution of the ODE system (1) is obtained only for $p = 0.5$ [14]. This sudden change from an ordered distribution to random replication is of potential interest as a building block for new antiviral strategies [6] because the error rates of RNA viruses are usually close to this critical value [2] and an increase of p is possible by the use of pharmaceutical drugs.

1.2 Related Work

So far, computational investigations in the large body of literature about the quasispecies model are all limited to the case where the landscape F is defined via the Hamming distance $f_i := \varphi(d_H(i,0))$ for some function φ (see e. g., [15]). Practically this means that all sequences with the same distance to the master sequence are equally fit, which is often a rather unrealistic assumption. With this simplifying assumption one expects that the problem reduces from an $N \times N$ to a $(\nu+1) \times (\nu+1)$ problem since all sequences in the same error class Γ_k are considered equivalent. For this special constellation efficient *approximative* schemes have been developed [11, 17], and this is the way how the quasispecies model and the error threshold phenomenon have usually been studied so far.

In contrast to these approximation methods our objective is to deal with the general form of the problem without any special assumptions, where so far no fast solvers are available [15]. For the general problem without assumptions on the structure of the fitness landscape we showed in [10] how to achieve a speedup factor of about 700 already for relatively short chain lenghts ($\nu = 25$) over standard approaches on mainstream hardware (see also Section 4). This was achieved by introducing an implicit sparse matrix vector product for the matrices arising in the quasispecies model. Our approach in [10] is based on the binary bitwise XOR operator and also sparsifies the matrix Q by taking into account only sequences within a maximum Hamming distance of $d_H^{\max}$. Henceforth we refer to the matrix vector product introduced in [10] as $\textsc{Xmvp}(d_H^{\max})$, whereas the standard matrix vector product which does not take into account the structure of W is called Smvp. As shown in [10], $\textsc{Xmvp}(\nu)$ is basically identical to Smvp, and $\textsc{Xmvp}(d_H^{\max})$ reduces the space complexity to $\Theta(N)$ as well as the time complexity to $\Theta\left(N \cdot \sum_{k=0}^{d_H^{\max}} \binom{\nu}{k}\right)$ from $\Theta(N^2)$ in both cases for Smvp. This illustrates that the sparsfied XOR-based matrix vector product nicely reduces the memory requirements, but the potential runtime reduction strongly depends on the level of tolerable inaccuracy in the computed concentrations of the species. Nevertheless, in the context of this paper $\textsc{Xmvp}(\nu)$ is an important benchmark for evaluating our new exact approach.

The main contributions in this paper are threefold. First, we develop fast algorithms for general formulations of the quasispecies model without special assumptions on the fit-

ness landscape. More specifically, we derive an exact implicit matrix vector product FMMP. It has a runtime complexity of $\Theta(N \log_2 N)$ and clearly outperforms all the previous approaches in terms of speed as well as in terms of accuracy. In contrast to [11, 17], the only assumption on F is that it is a diagonal matrix. We partly use randomly generated landscapes to illustrate the generality of our results. Second, we also show in Section 5 that solving the eigenvalue problems (3)-(5) with special assumptions on the fitness landscape can even be *exactly* reduced from an $N \times N$ to a $(\nu+1) \times (\nu+1)$ problem and that the solution of the original problem can be recovered exactly from the solution of the reduced problem. Third, we present efficient high performance implementations of the new solvers on GPUs and show how to exploit the high parallelization potential arising for fitness landscapes with Kronecker structure.

2. FAST MUTATION MATRIX PRODUCT

Most important algorithms for computing extreme eigenpairs of large scale eigenvalue problems are based on a matrix vector product, and we derive an efficient implicit matrix vector product with $Q \cdot F$ in this section. Since F is diagonal we can focus on a matrix vector product with the mutation matrix Q.

According to the definition of Q in Equation (2), mutation is modeled in the quasispecies model by independent coin flips with probability p for each position in the sequence. Each entry $Q_{i,j}$ is the result of the product of ν independent stochastic processes (coin flips). As already observed in [3, 12], this property allows for using the following Kronecker product representation which avoids the use of the Hamming distance and combines the ν independent stochastic single point mutation processes implicitly:

$$Q(\nu) = \bigotimes_{i=1}^{\nu} \begin{pmatrix} (1-p) & p \\ p & (1-p) \end{pmatrix}. \tag{7}$$

This representation can also be described recursively:

$$\begin{aligned} Q(0) &:= 1 \\ Q(\nu) &:= \begin{pmatrix} (1-p)\cdot Q(\nu-1) & p \cdot Q(\nu-1) \\ p \cdot Q(\nu-1) & (1-p)\cdot Q(\nu-1) \end{pmatrix}. \end{aligned} \tag{8}$$

It was already noted in [12] that as an immediate consequence of the Kronecker product based representation (7) the eigenvalues $\Lambda(\nu)$ and corresponding eigenvectors $V(\nu)$ of $Q(\nu)$ can be directly computed as

$$\Lambda(\nu) := \bigotimes_{i=1}^{\nu} \begin{pmatrix} 1 & 0 \\ 0 & 1-2p \end{pmatrix}$$

$$V(\nu) := 2^{-\frac{\nu}{2}} \cdot \bigotimes_{i=1}^{\nu} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

Since $V(\nu)$ is symmetric, this leads to the representation $Q(\nu) = V(\nu) \cdot \Lambda(\nu) \cdot V(\nu)$. This implicit description of the eigendecomposition can be transformed back into a componentwise representation:

$$\begin{aligned} (\Lambda(\nu))_{i,i} &= (1-2p)^{d_H(i,0)} \\ (V(\nu))_{i,j} &= 2^{-\frac{\nu}{2}} \cdot (-1)^{1/2 \cdot (d_H(i,0)+d_H(j,0)-d_H(i,j))}. \end{aligned}$$

One can conclude that $Q(\nu)$ has the eigenvalues $(1-2p)^k$ for $0 \le k \le \nu$ with multiplicities $\binom{\nu}{k}$. For $p < 1/2$ this implies that the mutation matrices Q are always positive definite. By $(F^{\frac{1}{2}}x)^T \cdot Q \cdot (F^{\frac{1}{2}}x) > 0$ for any landscape F and vector x, we also see that the eigenvalue problems from Equations (3)-(5) deal with positive definite matrices, and therefore all eigenvalues of W are positive real numbers.

Moreover, the eigendecomposition of $Q(\nu)$ also indicates that we may expect an efficient matrix vector product for $Q(\nu)$, since multiplication with the eigenvector matrix $V(\nu)$ (which is a Hadamard matrix) can be performed by the well known fast Walsh-Hadamard transform (FWHT, [8]) with $\Theta(N \log_2 N)$ operations. Similar to the FWHT, we derive an efficient matrix vector product with W in the following.

2.1 Kronecker Product Representations

Fast transformations (matrix vector products) usually appear hand in hand with a Kronecker product representation of the underlying matrix [18]. Based on Equation (8), the $N \times N$ matrix vector product

$$Q(\nu)\cdot v = \begin{pmatrix} (1-p)\cdot Q(\nu-1) & p \cdot Q(\nu-1) \\ p \cdot Q(\nu-1) & (1-p)\cdot Q(\nu-1) \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

can be computed efficiently in two ways. We can either first compute the two $N/2 \times N/2$ matrix vector products $\bar{v}_1 := Q(\nu-1)\cdot v_1$ and $\bar{v}_2 := Q(\nu-1)\cdot v_2$ and then

$$Q(\nu)\cdot v = \begin{pmatrix} (1-p)\bar{v}_1 & + & p\bar{v}_2 \\ p\bar{v}_1 & + & (1-p)\bar{v}_2 \end{pmatrix}, \tag{9}$$

or we directly compute

$$Q(\nu)\cdot v = \begin{pmatrix} Q(\nu-1)\cdot((1-p)v_1 + pv_2) \\ Q(\nu-1)\cdot(pv_1 + (1-p)v_2) \end{pmatrix}. \tag{10}$$

LEMMA 1. *Computing the matrix vector product $Q(\nu)\cdot v$ using Equation (9) or Equation (10) recursively has a runtime complexity $\Theta(N \log_2 N)$.*

PROOF. The complexity result can be deduced easily by using the Master Theorem from [1, p.73]. Setting $a = b = 2$ and $f(n) \in \Theta(n^{\log_b(a)})$ in the recurrence $T(n) = aT(n/b) + f(n)$ ($f(n)$ describes the cost for combining the solutions of the recursively solved subproblems) and applying the theorem leads to a runtime complexity $\Theta(N \log_2 N)$. □

Due to the Kronecker product representation of Q we also observe that both matrix vector products (9) and (10) work *in situ* as it is common for such transformations (cf. the FFT, for example). Since F is diagonal, matrix vector products with the matrices $Q \cdot F$, $F^{\frac{1}{2}} \cdot Q \cdot F^{\frac{1}{2}}$ or $F \cdot Q$ (cf. Equations (3)-(5)) also have a complexity of $\Theta(N \log_2 N)$. We refer to the fast matrix vector product based on (9) or (10) as *fast mutation matrix product* (FMMP). The efficient implementation of FMMP on GPUs is discussed in Section 4.

Comparison with approximative approaches.

A natural question arising now is how this newly defined implicit matrix vector product compares to the sparse matrix vector product XMVP($d_H^{\max}$) investigated in [10] for different $d_H^{\max}$. The maximum possible degree of sparsification in XMVP($d_H^{\max}$) is achieved by setting $d_H^{\max} = 1$ (consider only the "neighboring" sequences with Hamming distance one). This leads to a time complexity of $\Theta(N \cdot (\nu+1)) = \Theta(N \log_2 N + N)$ for evaluating the matrix vector product. This shows that our new implicit matrix vector product

FMMP with the *full* information of the matrix W is asymptotically even faster than the approximative matrix vector product XMVP($d_H^{\max}$) with the coarsest approximation $d_H^{\max} = 1$. Figure 2 illustrates that in practice FMMP even outperforms XMVP(1) already for quite small ν. This is actually not surprising since the implementation of XMVP($d_H^{\max}$) requires some additional computational overhead compared to the standard matrix vector product (see [10]), and due its memory access patterns it tends to get less competitive for increasing chain lengths.

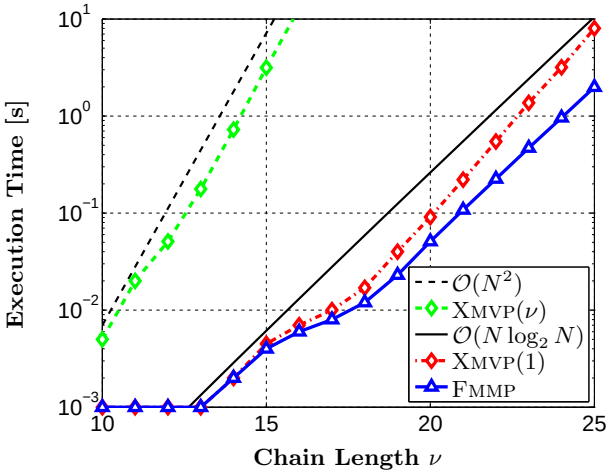

Figure 2: Runtimes of different implicit matrix vector products $W \cdot x$ on a single CPU core: Xmvp(ν) (fully accurate, corresponds to Smvp up to some small constant factor), Xmvp(1) (lowest possible accuracy), and Fmmp (fully accurate).

2.2 More General Mutation Processes

One of the well known points of criticism about the quasispecies model concerns the assumption of a uniform error rate, which states that mutations in different positions of the sequence are entirely independent and that they happen with the same probability p, which leads only to a coarse approximation of the evolution behavior of the population. A more general formulation of the model (1) only requires that Q is column stochastic. In the following, we discuss how to apply our fast algorithms in this more general setting with weaker assumptions on the structure of Q.

In the representation of Q in Equation (7) we see that there is actually no need for the single point mutations to have the same properties. Because the Kronecker product of two column stochastic matrices is again column stochastic (which can be easily verified), the only property which is required is that each of the 2×2 matrices is column stochastic.

Consequently, for arbitrary diagonal fitness landscapes F a fast matrix vector product with $Q \cdot F$ can be constructed as long as the mutation process is modeled as the composition of ν independent processes, one for each position in the RNA sequence. For such generalizations, Q may lose some properties such as symmetry, but this does not affect the performance of our approach since it just relies on the Kronecker product based definition of Q.

Even the restriction to 2×2 column stochastic matrices can be relaxed when needed for modeling more general situations. In principal, our approach allows for efficiently handling mutation matrices Q of the more general structure

$$Q = \bigotimes_{i=1}^{g} Q_{G_i}, \qquad Q_{G_i} \in \mathbb{R}^{2^{g_i} \times 2^{g_i}}, \quad \sum_{i=1}^{g} g_i = \nu, \qquad (11)$$

which models $1 \leq g \leq \nu$ groups of independent mutation processes where each group contains g_i single point mutations which are dependent on each other. Again the only requirement is that the matrices Q_{G_i} be column stochastic such that the model is valid. Clearly the size of the g_i influences the complexity of the resulting algorithms, but as long as they are not too large we still get efficient methods also for this very general setting. More formally, the generalization (11) may increase the complexity of $f(n)$ when applying the Master theorem in Lemma 1, and therefore possibly higher complexity bounds may result.

3. COMPUTING THE QUASISPECIES

Up to now we derived an efficient matrix vector product for the matrix W in the quasispecies model, but we did not yet address the problem of computing the desired extremal eigenvector. Remember that we are free to choose any of the problem formulations shown in Equations (3)-(5). In particular, this implies that without loss of generality we may restrict our investigation to a symmetric eigenproblem with real eigenvalues.

As already noted, at this point we are considering problems without any specific assumptions on the fitness landscape beyond diagonality. F is thus a general diagonal matrix and all its N values have to be stored. An unstructured landscape F also implies an unstructured resulting extremal eigenvector with $N = 2^\nu$ values to be stored. Due to this exponential growth with ν it is crucial to consider methods with minimum storage requirements beyond that.

Potentially relevant methods for computing an extremal eigenvector on the basis of an (implicitly available) matrix vector product are the basic power iteration or Lanczos/Arnoldi iterations. The latter require storing more intermediate vectors than the simpler power iteration and are thus less attractive for very large scale instances of the problem considered here. A completely different approach to the problem of computing the extremal eigenvector of a matrix would be to use randomized methods as suggested in [4, 9]. Although they usually allow for a strong reduction of the problem dimension, this reduction tends to strongly reduce the accuracy of the result. In the problem at hand, increasing ν not only leads to growing storage requirements, but also to growing accuracy requirements since the eigenvector to be computed contains relative concentrations. Especially for large ν we need higher accuracy and thus randomized methods are not the first choice in this context.

Power Iteration-Based Approach.

In the following, we focus on the investigation of a power iteration (PI) approach, which provides the best balance between storage requirements and accuracy. The matrix vector product used will be stated as argument in the notation, e. g., PI(FMMP) denotes the power iteration method on the basis of the fast FMMP matrix vector product introduced in Section 2.1. Since W is positive definite and the Perron-Frobenius theorem is applicable, it is assured that

the eigenvalues of W satisfy

$$\lambda_0 > \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{N-1} > 0,$$

and thus convergence of the power iteration is guaranteed.

As starting vector s for the power iteration we use the fitness landscape itself $s = \mathrm{diag}(F)/\|\mathrm{diag}(F)\|_1$, since we can expect the extremal eigenvector of $W = Q \cdot F$ to be of a similar shape (remember that the extremal eigenvector of Q is a multiple of $(1, 1, \ldots, 1)^T$. As stopping criterion we monitor the residual $R(\tilde{\lambda}, \tilde{x}) = \|W\tilde{x} - \tilde{\lambda}\tilde{x}\|_2$.

Shifting for Convergence Acceleration.

The convergence rate of the power iteration depends on the ratio $\lambda_1/\lambda_0 < 1$. In the following we derive a shift μ from the properties of the problem considered in order to enhance the convergence rate to $(\lambda_1 - \mu)/(\lambda_0 - \mu)$. Of course, μ has to be chosen such that $\lambda_0 - \mu$ remains the largest eigenvalue of $W - \mu I$.

In order to derive a generally applicable shift μ, we again examine the Kronecker product representation of $Q(\nu)$ in Equation (7). By exploiting the property $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ we can represent the inverse $Q(\nu)^{-1}$ as

$$Q(\nu)^{-1} = (1-2p)^{-\nu} \bigotimes_{i=1}^{\nu} \begin{pmatrix} (1-p) & -p \\ -p & (1-p) \end{pmatrix}. \qquad (12)$$

According to Equation (12), the absolute row and column sums of $Q(\nu)^{-1}$ are all $(1-2p)^{-\nu}$. With this knowledge we can compute $\|\cdot\|_1$ of the matrices $W = Q \cdot F$ and $W^{-1} = F^{-1} \cdot Q^{-1}$ in order to derive bounds for the largest eigenvalue λ_0 as well as for the smallest eigenvalue λ_{N-1} of W:

$$\lambda_0 \leq \|W\|_1 \leq \|Q\|_1 \cdot \|F\|_1 = f_{\max} := \max_{i=0}^{N-1} f_i$$

and

$$\begin{aligned} \lambda_{N-1}^{-1} \leq \left\|W^{-1}\right\|_1 &\leq \left\|F^{-1}\right\|_1 \cdot \left\|Q^{-1}\right\|_1 \\ &= f_{\min}^{-1} \cdot (1-2p)^{-\nu}, \end{aligned}$$

where $f_{\min} := \min_{i=0}^{N-1} f_i$. Since $f_{\min} > 0$ and $p < 1/2$ we can conclude that $\lambda_{N-1} \geq (1-2p)^{\nu} f_{\min}$ and therefore applying the shift $\mu := (1-2p)^{\nu} f_{\min}$ is in any case possible, since $\lambda_0 - \mu$ remains the largest eigenvalue and hence the shifted power iteration still converges to the desired eigenvector. Although this choice of the shift μ is very conservative, using it results in a clearly measurable reduction of the number of iterations of about ten percent and more for the random landscapes we considered.

Towards a Shift-and-Invert Method.

We would like to point out that for Q there even exists a $\Theta(N \log_2 N)$ implicit shift and invert matrix vector product since

$$\begin{aligned} (Q(\nu) - \mu I)^{-1} v &= (V(\nu)\Lambda(\nu)V(\nu) - \mu I)^{-1} v \\ &= V(\nu) \left(\Lambda(\nu) - \mu I\right)^{-1} V(\nu) v, \end{aligned}$$

where $V(\nu)v$ can be computed efficiently by the FWHT and the cost of the multiplication with the diagonal matrix $(\Lambda(\nu) - \mu I)^{-1}$ is linear in N.

The construction of an efficient solver for $Q(\nu)F - \mu I$ with arbitrary diagonal F is more complicated and is one of the topics of our current work. With such a building block the application of inverse iteration or Rayleigh quotient iteration becomes possible. Other improvements for special structures of the landscape F are discussed in Section 5.

4. EFFICIENT GPU IMPLEMENTATION

In this section a parallel implementation of the power iteration based on FMMP and, for comparison, based on XMVP($d_H^{\max}$), is discussed. We illustrate that the new FMMP-based approach is well suited for exploiting the potential of state-of-the-art GPU hardware. For improving portability we implemented our algorithms using OpenCL. In particular, this implies that our codes can also make use of modern multi-core CPUs besides GPUs.

The implementation presented only uses standard techniques and is not highly optimized yet. There is still space for further performance gains by applying advanced code optimization techniques known from optimized implementations of FFT and FWHT, respectively. Nevertheless, in view of the exponential growth of the problem at hand, the pivotal aspect is the algorithmic improvement. Relatively speaking, the improvements achievable by further code optimizations are expected to be lower.

Our implementation of the fast matrix vector product FMMP shown in Algorithm 1 is based on Equation (9), but in an iterative instead of a recursive formulation. The corresponding algorithm for Equation (10) can be obtained by turning around the outermost i-loop. Note that up to some multiplications with constants this formulation is very similar to basic implementations of the FWHT or of the FFT.

Algorithm 1 FMMP based on Equation (9)

Input: $v \in \mathbb{R}^N$
Output: $Q(\nu) \cdot v$

```
1: for i ← 1 to N/2 by 2 · i do
2:     for j ← 0 to N − 1 by 2 · i do
3:         for k ← 0 to i − 1 do
4:             t1 ← v[j + k]
5:             t2 ← v[j + k + i]
6:             v[j + k] ← (1 − p) · t1 + p · t2
7:             v[j + k + i] ← p · t1 + (1 − p) · t2
8:         end for
9:     end for
10: end for
```

For execution on a GPU or, more general, for the computations with OpenCL we need to provide a so-called kernel function where the current thread id is queried for deciding which computations have to be executed. A corresponding formulation is shown in Algorithm 2, where ID denotes the thread id and "&" denotes the bitwise AND operation.

Like Algorithm 1, Algorithm 2 performs exactly $\log_2 N$ iterations of the outer i-loop, but in contrast to Algorithm 1 the second ID-loop has a constant range of $N/2$. However, we see that those $N/2$ iterations of the ID-loop are entirely independent of each other and therefore a very high degree of parallelism is available. The index computation in line 3 can be easily verified starting from the usual formula for this

Algorithm 2 GPU ready implementation of Equation (9)

Input: $v \in \mathbb{R}^N$
Output: $Q(\nu) \cdot v$

1: **for** $i \leftarrow 1$ **to** $N/2$ **by** $2 \cdot i$ **do**
2: **for** ID $\leftarrow 0$ **to** $N/2 - 1$ **by** 1 **do**
3: $j \leftarrow 2 \cdot \text{ID} - (\text{ID} \;\&\; (i-1))$
4: $t_1 \leftarrow v[j]$
5: $t_2 \leftarrow v[j+i]$
6: $v[j] \leftarrow (1-p) \cdot t_1 + p \cdot t_2$
7: $v[j+i] \leftarrow p \cdot t_1 + (1-p) \cdot t_2$
8: **end for**
9: **end for**

task:

$$\begin{aligned} & 2 \cdot i \cdot \lfloor \text{ID}/i \rfloor + \text{ID} \bmod i \\ = \;& 2 \cdot (\text{ID} - (\text{ID} \bmod i)) + \text{ID} \bmod i \\ = \;& 2 \cdot \text{ID} - \text{ID} \bmod i \\ = \;& 2 \cdot \text{ID} - (\text{ID} \;\&\; (i-1)) \end{aligned}$$

The last step exploits that i is always a power of two and that in this case the modulo computation can be performed via the cheaper bitwise AND operation. When using Algorithm 2 as base for an OpenCL implementation, the i-loop runs at the host, in each iteration of the i-loop the kernel is called with $N/2$ threads, and lines 3-7 correspond exactly to the kernel function where ID is determined by the proper OpenCL command.

This implementation of the fast matrix vector product FMMP scales nicely with the parallelization potential of the underlying hardware. Since there is a relatively high number of memory operations compared to floating-point operations, one can expect that the memory bandwidth of the hardware has an important influence on the overall performance. This can be observed in the experiments, since the performance achieved on the GPUs used exactly corresponds to their particular memory bandwidth.

Beyond the matrix vector product, the power iteration method only needs a fast procedure for the summation of the components of a vector for computing norms and the residual after each iteration. Since the summation of the components of a vector can be relatively well parallelized, this part of the power iteration method has almost no influence on the overall execution time.

Experimental Results

For our experiments, we did not make any special assumptions on the values of the fitness landscape F, but similar to [15] we used the following random landscapes:

$$f_0 = c, \quad f_i = \sigma \cdot (\eta_{\text{rnd}}(i) + 0.5), \tag{13}$$

where $c > 0$, $\sigma \in (0, c/2)$ and $\eta_{\text{rnd}}(i)$ is a call to a uniform random number generator on the interval $[0, 1]$.

For evaluating the methodical improvement provided by our fast matrix vector product we compare PI(FMMP) with PI(XMVP(ν)) and PI(XMVP(5)). The choice $d_H^{\max} = \nu$ in the approximative matrix vector product XMVP($d_H^{\max}$) illustrates how FMMP compares to the standard matrix vector product SMVP $\equiv$ XMVP(ν), and the choice $d_H^{\max} = 5$ in XMVP($d_H^{\max}$) has been shown in [10] to yield an approximation error around 10^{-10}. The accuracy achieved with smaller values for $d_H^{\max}$ is usually too low [10] and thus these values are not considered here (remember that we have shown in Section 2.1 that FMMP is faster than XMVP($d_H^{\max}$) even for the lowest possible accuracy with $d_H^{\max} = 1$!).

Figure 3 shows the great improvement FMMP delivers as building block of the power iteration. In all cases $p = 0.01$ was chosen and the iteration was stopped when the residual $R(\tilde{\lambda}, \tilde{x})$ for the approximated eigenpair $(\tilde{\lambda}, \tilde{x})$ was less than an appropriate threshold τ ($\tau = 10^{-15}$ for $d_H^{\max} = \nu$, $\tau = 10^{-10}$ for $d_H^{\max} = 5$). All computations were performed on an NVIDIA TESLA C2050 with 3Gb memory. The runtimes shown are overall execution times which include the data transfer from and to the GPU.

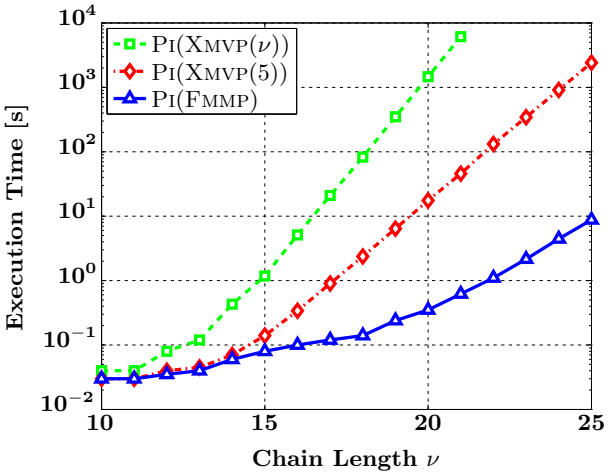

Figure 3: Overall execution times on a GPU for finding the dominating eigenvector of $Q \cdot F$ ($p = 0.01$) on a random landscape (13) with $c = 5$ and $\sigma = 1$ for increasing chain length ν.

In Figure 2 we illustrated execution times different variants of the matrix vector product Wv on a CPU core, and in Figure 3 we showed GPU execution times of the entire power iteration for computing the desired dominant eigenvector based on these matrix vector products. In order to better differentiate the influence of different algorithms from the influence of different hardware platforms, we depict the speedups for solving the quasispecies model for various combinations of algorithms and hardware in Figure 4. As reference value for evaluating the speedup we used the execution times of PI(XMVP(ν)) running on a CPU (INTEL I5 750 @ 2.67GHZ with 4Gb memory), which achieves the same accuracy as PI(FMMP). For $\nu \geq 22$ the execution times for PI(XMVP(ν)) are so long that they had to be extrapolated based on the curves in Figures 2 and 3 for estimating the speedup values.

Figure 4 basically summarizes the results discussed in this paper and in [10]. For different algorithms on the same hardware we always observe a different slope of the speedup curve as we would expect from theory. This means that our algorithms scale well with the underlying hardware. We also see that the speedup delivered by FMMP over $O(N^2)$ algorithms is of the expected order of magnitude. The speedup values of FMMP on the CPU are partly above the theoretically expected value of $N^2/(N \log_2 N)$, which is caused by the computational overhead in XMVP(ν). For the same algorithms

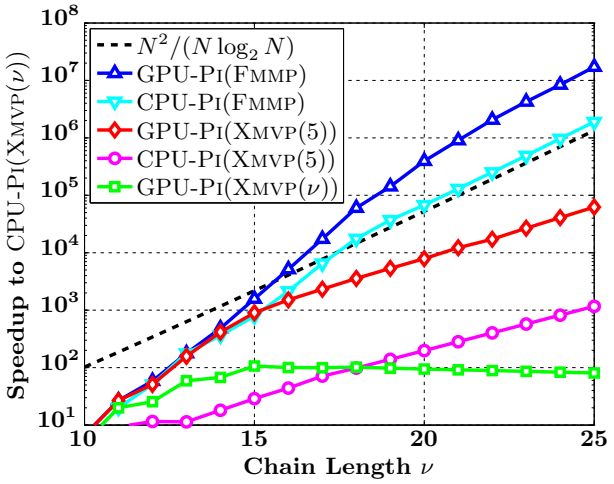

Figure 4: Speedup factors for solving the quasispecies model based on various combinations of algorithms and hardware over power iteration based on Xmvp(ν) running on a CPU. For a given algorithm, different hardware platforms lead to (asymptotically) parallel speedup curves with the same slope, whereas the curves for different algorithms have different slopes.

on different hardware we observe parallel shifted speedup curves and great performance benefit from exploiting the potential of GPUs. One should keep in mind that the reference computation and the fastest combination, PI(FMMP) on the GPU, deliver the same results. The observed speedup factor of about $2 \cdot 10^7$ for $\nu = 25$ underlines the combined benefits of algorithmic improvements and efficient implementation which exploits the potential of GPUs.

5. SPECIAL FITNESS LANDSCAPES

Up to now the presented methods were designed for the most general formulation of the quasispecies model when there are no special assumptions on the diagonal fitness landscape F. We made already the observation that the mutation matrix Q may be defined via the Hamming distance or somewhat more generally via the Kronecker product. If the fitness landscape also has such a representation one can expect that the matrices Q and F in some sense "fit together" and that this structure translates to the product $W = Q \cdot F$. How this additional structure in the matrix W can be used to derive further enhancements is discussed in Section 5.2. In fact, Kronecker product structure of F does not only translate into the matrix W, it even translates into the dominating eigenvector, which emphasizes that such structure should also translate into our algorithms and their complexity, respectively. The resulting structure in the dominating eigenvector reflects the loss of generality in the results since the usually N degrees of freedom in the fitness landscape F are drastically reduced by further assumptions on the structure of F. How well such specially structured landscapes agree with observations in nature has to be clarified.

5.1 Hamming Distance-Based Landscapes

Since efficient solvers for general landscapes have not been available so far and since good approximative methods have been developed for landscapes based on the Hamming distance [11, 17], the existing literature almost exclusively deals with landscapes of this very special type. In this section we show that given a fitness landscape based on the Hamming distance with $f_i = \varphi(d_H(i,0))$, it is sufficient to solve a $(\nu+1) \times (\nu+1)$ eigenproblem to get the *exact* eigenvector of the full $N \times N$ eigenproblem and therefore approximative methods are not really needed in this case. One should note that this dimensionality reduction also takes place in the extremal eigenvector which corresponds to a reduction from N to $\nu + 1$ degrees of freedom in the structure of the eigenvector.

The usual way to treat Hamming distance-based landscapes (see, e. g., [14]) is by defining a reduced mutation matrix Q based on the observation that mutations do not longer have to be considered on the granularity of single molecules, but instead as mutations from some fixed element of an error class Γ_d to any element of another error class Γ_k. For $0 \le d, k \le \nu$ the reduced mutation matrix is defined as

$$Q_{\Gamma_{d,k}} := \sum_{j=k+d-\nu}^{\min(k,d)} \binom{\nu-d}{k-j} \binom{d}{j} \frac{p^{k+d-2j}}{(1-p)^{(k+d-2j)-\nu}}, \quad (14)$$

which can be derived by explicitly computing the probability that some fixed molecule from error class Γ_d mutates into some molecule from error class Γ_k. Based on this definition, approximative methods apply low order pertubation theory [11, 17].

In order to develop a better understanding of this problem reduction, we present a more computationally oriented perspective. We start by considering the power iteration for the full $N \times N$ problem as we did so far for the general case, and then we simplify it step by step. At the end we basically also arrive at the same reduced formulas but with the advantage of a clear interpretation and understanding of the structures arising. This knowledge can then be used to efficiently compute the desired eigenvector.

We call $v = (v_0, \ldots, v_{\nu-1})$ an *error class vector* if

$$\{i, i'\} \subseteq \Gamma_k \Rightarrow v_i = v_{i'}$$

holds for all $0 \le k \le \nu$ and all $0 \le i, i' < N$. We identify the $\nu + 1$ different values of v by $v_{\Gamma_0}, \ldots, v_{\Gamma_\nu}$, therefore we have the relationship $v_i = v_{\Gamma_{d_H(i,0)}}$. Analogously to error class vectors we define *error class landscapes*: We call F an error class landscape iff $\mathrm{diag}(F)$ is an error class vector.

For i, i' with $\{i, i'\} \subseteq \Gamma_k$ we define a permutation $\sigma_{i,i'}$ for ν-bit vectors which maps the k bits which are set in i to the k bits which are set in i'. Let $\{\beta_i^0, \ldots, \beta_i^{k-1}\}$ and $\{\beta_{i'}^0, \ldots, \beta_{i'}^{k-1}\}$ denote the indices of the bits set in i and i', respectively, then $\sigma_{i,i'}$ is in cycle notation defined as

$$\sigma_{i,i'} := (\beta_i^0 \ \beta_{i'}^0)(\beta_i^1 \ \beta_{i'}^1) \ldots (\beta_i^{k-1} \ \beta_{i'}^{k-1}).$$

When writing $\sigma_{i,i'}(j)$ we mean that the bits of the ν-bit vector j are permuted according to $\sigma_{i,i'}$. Just by definition it is clear that for $\sigma_{i,i'}$ with any two integers i, i' satisfying $d_H(i,0) = d_H(i',0)$ the following statements hold:

(I) $d_H(j,0) = d_H(\sigma_{i,i'}(j),0)$ for all $0 \le j < N$

(II) $\{\sigma_{i,i'}(j) \mid j \in \Gamma_k\} = \Gamma_k$ for all $0 \le k \le \nu$

(III) $\sigma_{i,i'}(i) = i'$

(IV) $d_H(i,j) = d_H(\sigma_{i,i'}(i), \sigma_{i,i'}(j)) = d_H(i', \sigma_{i,i'}(j))$ for all $0 \leq j < N$

Using these properties we can show the following Lemma 2.

LEMMA 2. *Given $W = Q \cdot F$ with F representing an error class landscape and an error class vector v. Then the vector $W \cdot v = \bar{v}$ is also an error class vector.*

PROOF. *Examining the multiplication componentwise we get*

$$\sum_{j=0}^{N-1} Q_{i,j} \cdot f_j \cdot v_j = \bar{v}_i \qquad \text{for all } 0 \leq i < N.$$

We know that v is an error class vector and have to prove that

$$\{i, i'\} \subseteq \Gamma_k \Rightarrow \bar{v}_i = \sum_{j=0}^{N-1} Q_{i,j} \cdot f_j \cdot v_j = \sum_{j=0}^{N-1} Q_{i',j} \cdot f_j \cdot v_j = \bar{v}_{i'}$$

holds because then $\bar{v}$ is clearly also an error class vector. Now suppose we are given any i, i' with $d_H(i,0) = d_H(i',0)$, then the following equivalences hold:

$$\begin{aligned}
& \sum_{j=0}^{N-1} Q_{i,j} \cdot f_j \cdot v_j = \sum_{j=0}^{N-1} Q_{i',j} \cdot f_j \cdot v_j \\
\Leftrightarrow \quad & \sum_{j=0}^{N-1} (Q_{i,j} - Q_{i',j}) \cdot f_j \cdot v_j = 0 \\
\Leftrightarrow \quad & \sum_{k=0}^{\nu} \sum_{j \in \Gamma_k} (Q_{i,j} - Q_{i',j}) \cdot f_j \cdot v_j = 0 \qquad (15) \\
\Leftrightarrow \quad & \sum_{k=0}^{\nu} F_{\Gamma_k} \cdot v_{\Gamma_k} \cdot \sum_{j \in \Gamma_k} (Q_{i,j} - Q_{i',j}) = 0 \qquad (16) \\
\Leftrightarrow \quad & \sum_{k=0}^{\nu} F_{\Gamma_k} \cdot v_{\Gamma_k} \cdot \sum_{j \in \Gamma_k} \underbrace{(Q_{i,j} - Q_{i',\sigma_{(i,i')}(j)})}_{=0 \text{ (by (IV))}} = 0 \qquad (17) \\
\Leftrightarrow \quad & \sum_{k=0}^{\nu} F_{\Gamma_k} \cdot v_{\Gamma_k} \cdot 0 = 0 \\
\Leftrightarrow \quad & 0 = 0
\end{aligned}$$

(15) $\Leftrightarrow$ (16) *because v is an error class vector and F is an error class landscape.* (16) $\Leftrightarrow$ (17) *since error classes are invariant under $\sigma_{i,i'}$ (by (II)).* □

Using Lemma 2 we can now conclude that the dominating eigenvector of $Q \cdot F$ under the assumption that F is an error class landscape is also an error class vector. This follows immediately by using any error class vector as starting vector for the power iteration. We also know now that by using the power iteration all intermediate vectors are error class vectors. Apparently we should not compute the resulting value for all vector components during an iteration, but instead of that it is more efficient to compute the resulting value only for an arbitrary representative index from each error class. The natural and most obvious choices of such representatives would probably be the set $\{2^k - 1 \mid 0 \leq k \leq \nu\}$.

So far, these insights reduced the original $N \times N$ problem

$$\bar{v}_i = \sum_{j=0}^{N-1} Q_{i,j} \cdot f_j \cdot v_j$$

to a $(\nu + 1) \times N$ problem

$$\bar{v}_{\Gamma_d} = \sum_{k=0}^{\nu} F_{\Gamma_k} \cdot v_{\Gamma_k} \cdot \sum_{j \in \Gamma_k} Q_{2^d - 1, j}.$$

Now remember the definition of $Q_{\Gamma_{d,k}}$ in Equation (14) with the interpretation that $Q_{\Gamma_{d,k}}$ is the probability that some fixed element from Γ_d mutates into any element of Γ_k. Observe that this is exactly what the last sum in the previous equation computes and therefore we arrive at the $(\nu + 1) \times (\nu + 1)$ problem

$$\bar{v}_{\Gamma_d} = \sum_{k=0}^{\nu} Q_{\Gamma_{d,k}} \cdot F_{\Gamma_k} \cdot v_{\Gamma_k}.$$

Thus, efficiently computing the eigenvector of the original problem by the power iteration shows that the original eigenvector can be computed out of the eigenvector of the reduced $(\nu + 1) \times (\nu + 1)$ matrix. But note that when v denotes the original and v_Γ the eigenvector of the reduced problem it is not the case that we get $[\Gamma_k] = v_{\Gamma_k} = \sum_{j \in \Gamma_k} v_j$, because the reduced matrix itself does not correspond to a change of variables from single molecules to error classes. As we have seen, it corresponds to a change from single molecules to representatives of classes. In summary, this means that given v_Γ, the dominating eigenvector of the reduced problem, we can compute the cumulative concentrations for the original problem by

$$[\Gamma_k] = \binom{\nu}{k} \frac{v_{\Gamma_k}}{\sum_{j=0}^{\nu} \binom{\nu}{j} v_{\Gamma_j}}.$$

This rescaling reflects the interpretation that the eigenvector of the reduced problem contains the concentrations of some representatives of the error classes Γ_k.

Besides its low computational complexity, the main advantage of this exact problem reduction is that we do not have to worry about approximative aspects or about the details of the perturbation theory used. It is sufficient to set up the needed matrices by the well known formulas and to use a standard solver for computing the extremal eigenvector of a small $(\nu + 1) \times (\nu + 1)$ matrix.

5.2 Kronecker Product-Based Landscapes

In Section 2.2 we showed already that using a Kronecker product-based definition of Q resulted in a significant increase in the generality of Q without causing higher computational cost. Analogously, if we have a Kronecker product representation of the fitness landscape F, we can derive solution methods more efficient than the ones for arbitrary landscapes.

When we considered the Kronecker product-based representation of Q in Section 2, we saw that this representation resulted in a great reduction of the complexity in the matrix vector product with Q. The product with the diagonal matrix F is inherently fast, but a Kronecker product representation of F has additional benefits: on the one hand, it leads to a reduction of the memory requirements, on the other hand, it allows for splitting up the problem into independent subproblems which can be solved in parallel.

Following the same scheme as for Q in Equation (11), we define Kronecker landscapes F via the Kronecker product of

matrices F_{G_i} of dimension $2^{g_i} \times 2^{g_i}$:

$$F = \bigotimes_{i=1}^{g} F_{G_i}, \qquad F_{G_i} \in \mathbb{R}^{2^{g_i} \times 2^{g_i}}, \quad \sum_{i=1}^{g} g_i = \nu. \tag{18}$$

The big advantage of a Kronecker product-based representation is that it decouples the problem entirely and we therefore get an implicit description of our solution. In terms of complexity this means that the usual multiplicative connection becomes an additive one. More concretely, if all g_i are equal, it allows for reducing the problem for chain length ν into g subproblems of size $2^{\nu/g}$ which can all be solved independently instead of solving one problem of size 2^{ν}. For example, the quasispecies model for a chain length $\nu = 100$ (which occurs in existing viruses of interest) is by far out of reach of any of the currently available computational technology. However, for a Kronecker fitness landscape with $g = 4$ it could be reduced to four subproblems of dimension 2^{25}, which can be solved fast with the methods introduced in this paper.

We also observe that Kronecker product-based landscapes have a much richer structure than the ones based on Hamming distances, since they have $\sum_{i=1}^{g} 2^{g_i}$ degrees of freedom compared to only $\nu + 1$ in the case of Hamming distance based landscapes.

If Q and F both have a Kronecker product representation and if these representations are also compatible (the simplest case would be that Q and F have the same g_i in Equations (11) and (18)), then it is not even necessary to compute the entire eigenvector. By the theory of the Kronecker product we get an implicit description of the desired eigenvector, since the eigenvalues and eigenvectors of a matrix $M = \bigotimes_{i=1}^{g} M_i$ can be computed out of the eigenvalues and eigenvectors of the component matrices M_i.

We still have to clarify how well Q and F fit together. This is limited by the applicability of the so called mixed product formula

$$(A \otimes B)(C \otimes D) = AC \otimes BD$$

This formula is obviously only valid if AC and BD exist and this is exactly the point where we can see a possible increase in the joint group sizes of Q and F when considering $W = Q \cdot F$. But since our new solvers easily handle chain lengths of $\nu \approx 25$ within a few seconds, one can still expect efficient computations for fairly complex structures.

Based on an implicit description of the eigenvalues and eigenvectors of the component matrices one could develop algorithms for retrieving informations about the desired eigenvector of the full matrix W. More specifically, it is quite simple to compute the extremal eigenvalue or relevant parts of it efficiently. For example, one could compute the minimum and maximum concentration for each error class which should provide sufficient information for investigating the question whether the error threshold phenomenon occurs or not. So far, we did not investigate such methods in great detail because the biological relevance of Kronecker product-based fitness landscapes is not yet clear. From a computational point of view they obviously have many interesting properties and together with efficient methods for extracting information out of the implicit eigenvector description it would be possible to model chain lengths far beyond what is possible today, easily including complex viruses with long RNA chains. Moreover, for Kronecker product-based landscapes it is relatively easy to extend the quasispecies model beyond a binary alphabet to the full four element RNA alphabet.

6. CONCLUSIONS

We showed how to efficiently and accurately solve the very large scale eigenvalue problem occurring in Eigen's quasispecies model for the evolution of virus populations. By exploiting the special structure and properties of the eigenvalue problem, we developed the fast matrix vector product FMMP which allows for partly utilizing concepts similar to the fast Walsh-Hadamard Transform and for reducing the arithmetic complexity from $\Theta(N^2)$ to $\Theta(N \log_2 N)$. In contrast to earlier approximative approaches [10, 11, 17] our new solver produces results as accurate as possible within the limitations of floating-point accuracy. Moreover, it can handle models with more realistic mutation processes which are too complex for earlier approaches.

With an efficient parallel implementation of a power iteration approach based on FMMP and by utilizing modern GPU hardware we achieved enormous speedups up to $2 \cdot 10^7$ (for a chain length $\nu = 25$) over equivalent standard solvers. Even compared to existing approximative methods which loose about 5 decimal digits of accuracy [10], we could achieve speedups up to 250 (for a chain length $\nu = 25$).

Beyond the improvements for general fitness landscapes in the quasispecies model we outlined how to reduce the computational cost even further for special cases of fitness landscapes, leading to further substantial reductions of the space and runtime complexity as well as to an even higher parallelization potential.

Given the new solver presented in this paper, the main limiting factor in computationally solving the quasispecies model is not any more the runtime, but the memory requirements. Consequently, in the future we will focus on distributed memory approaches, on approximative strategies for a fast matrix vector product, and on efficient methods which allow for computing quasispecies concentrations at various resolution levels.

Acknowlegdements.

This work was partly supported by the Austrian Science Fund (FWF) under contract S10608 (NFN SISE). We would like to thank Peter Schuster for introducing us to the quasispecies model.

7. REFERENCES

[1] Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[2] Drake, J. Rates of spontaneous mutation among RNA viruses. *Proceedings of The National Academy of Sciences 90*, 9 (1993), 4171–4175.

[3] Dress, A. W. M., and Rumschitzki, D. S. Evolution on sequence space and tensor products of representation spaces. *Acta Applicandae Mathematicae 11* (1988), 103–115.

[4] Drineas, P., Drinea, E., and Huggins, P. S. An experimental evaluation of a Monte-Carlo algorithm for singular value decomposition. *Proceedings of the 8th Panhellenic conference on Informatics* (2003), 279–296.

[5] Eigen, M. Selforganization of matter and the evolution of biological macromolecules. *Naturwissenschaften 58* (1971), 465–523.

[6] Eigen, M. Error catastrophe and antiviral strategy. *Proceedings of the National Academy of Sciences of the United States of America 99*, 21 (2002), 13374–13376.

[7] Eigen, M., and Schuster, P. A principle of natural self-organization. *Naturwissenschaften 64* (1977), 541–565.

[8] Fino, B., and Algazi, V. Unified matrix treatment of the fast Walsh-Hadamard transform. *IEEE Transactions on Computers C-25*, 11 (1976), 1142 –1146.

[9] Mascagni, M., and Karaivanova, A. A parallel quasi-monte carlo method for computing extremal eigenvalues. *Monte Carlo and Quasi-Monte Carlo Methods 2000* (2000), 369 – 380.

[10] Niederbrucker, G., and Gansterer, W. N. Efficient solution of evolution models for virus populations. *Procedia Computer Science 4* (2011), 126 – 135.

[11] Nowak, M., and Schuster, P. Error thresholds of replication in finite populations mutation frequencies and the onset of Muller's ratchet. *Journal of Theoretical Biology 137*, 4 (1989), 375 – 395.

[12] Rumschitzki, D. S. Spectral properties of Eigen evolution matrices. *Journal of Mathematical Biology 24* (1987), 667–680.

[13] Schuster, P. Prediction of RNA secondary structures: from theory to models and real molecules. *Reports on Progress in Physics 69* (May 2006), 1419–1477.

[14] Schuster, P. *The mathematics of Darwinian systems.* 2008. Appendix for the book: Manfred Eigen, 'From Strange Simplicity to Complex Familiarity, Vol.I'.

[15] Schuster, P. Mathematical modeling of evolution. Solved and open problems. *Theory in Biosciences 130* (2011), 71–89.

[16] Seneta, E. *Non-negative Matrices and Markov Chains (Springer Series in Statistics).* Springer, January 2006.

[17] Swetina, J., and Schuster, P. Self-replication with errors: A model for polynucleotide replication. *Biophysical Chemistry 16*, 4 (1982), 329 – 345.

[18] van Loan, C. F. The ubiquitous Kronecker product. *J. Comput. Appl. Math. 123* (November 2000), 85–100.