Efficient Simultaneous Simulation of Markov Chains

Carsten Wächter[1] and Alexander Keller[2]

[1] Ulm University, Germany, carsten.waechter@uni-ulm.de
[2] Ulm University, Germany, alexander.keller@uni-ulm.de

Summary. Markov chains can be simulated efficiently by either high-dimensional low discrepancy point sets or by padding low dimensional point sets. Given an order on the state space, both approaches can be improved by sorting the ensemble of Markov chains. We analyze deterministic approaches resulting in algorithmic simplifications and provide intuition when and why the sorting works. Then we discuss the efficiency of different sorting strategies for the example of light transport simulation.

I spent an interesting evening recently with a grain of salt.

Shannon: A Mathematical Theory of Communication, 1948

1 Introduction

A Markov chain describes a memoryless stochastic process, where the transition probabilities do not depend on the history of the process. For example Shannon modeled the English language by a Markov chain, where he computed the relative frequencies of one word following another from an English book. Using these transition probabilities he generated seemingly English sentences. Many more evolution processes, like e.g. the Brownian motion or particle trajectories, can be described as Markov chains.

Properties of processes can be estimated by averaging the contributions of multiple Markov chains. Instead of simulating each trajectory independently, it has been found that simultaneously simulating Markov chains using correlated samples and sorting the ensemble of states after each transition step can notably improve convergence.

2 Simultaneous Simulation of Markov Chains

The idea of improving the simultaneous simulation of Markov chains with quasi-Monte Carlo methods by an intermediate sorting step was originally introduced by Lécot in a series of papers dealing with the Boltzmann equation [Léc89a, Léc89b, Léc91, LC98] and later on the heat equation [KL99]. This idea was then used and refined for solving the heat equation on a grid by Morokoff and Caflisch [MC93] and recently extended by L'Écuyer, Tuffin, Demers et al. [LL02, LT04a, LT04b, DLT05, LLT05, LDT06, LLT06, HLL07, LDT07] to incorporate randomized versions of the algorithm and splitting for rare event simulation. Independent research, but in a way related to the above approaches, was conducted by [BNN$^+$98] in the field of computer graphics.

In the following we simplify the deterministic version of the scheme. For the derivation we use the algorithm from [LT04b]. The results give practical insight, when and why the scheme is superior to approaches without sorting and how to implement it.

2.1 Analysis of a Deterministic Algorithm in One Dimension

The algorithm presented in [LT04b] simultaneously simulates Markov chains on a discrete state space E with an initial distribution $\mu := (\mu_i)_{i\in E}$ and a transition matrix $P := (p_{i,j})_{i,j\in E}$. Using a $(t,2)$-sequence $(x_i)_{i\in\mathbb{N}_0}$ in base b [Nie92], $N = b^m$ chains are simulated in parallel, where $\mathbf{X}_{n,l}$ is the state of chain l at time step n and $\mathbb{N}_0$ are the natural numbers including zero. Further the algorithm requires that for $m > t$ the $N = b^m$ subsequent components $x_{i,1}$ form a $(0,m,1)$-net in base b. As shown in [LT04b] the algorithm converges if

$$\forall k \in E : \sum_{l=1}^{N-1} \left| \sum_{n=1}^{k-1} p_{l+1,n} - \sum_{n=1}^{k-1} p_{l,n} \right| \leq 1 \tag{1}$$

holds.

Simplification of the Algorithm

We now consider the Sobol' sequence $x_l = (x_{l,1}, x_{l,2}) = (\Phi_2(l), \Phi_S(l)) \in [0,1)^2$, which is a $(0,2)$-sequence in base $b=2$ and fulfills the assumptions required for the convergence condition to hold. For the definition of the original Sobol' sequence see [Sob67], while a simple code example for $(\Phi_2(l), \Phi_S(l))$ is found in [KK02b].

The simulation itself starts at time step $n = 0$ initializing state $\mathbf{X}_{0,\lfloor N\cdot x_{l,1}\rfloor}$ for $0 \leq l < N$ using $x_{l,2}$ for the realization of μ. The algorithm then continues by sorting the states (this will be discussed in detail in Section 3) and continues the chains by computing $\mathbf{X}_{n,\lfloor N\cdot x_{(l+n\cdot N),1}\rfloor}$ using $x_{(l+n\cdot N),2}$ for $0 \leq l < N$ to realize transitions according to P. The index

$$\sigma(l) := \lfloor N \cdot x_{(l+n\cdot N),1} \rfloor$$

for selecting the next state for transition in fact uses the van der Corput sequence Φ_2 in base 2, which is a $(0,1)$-sequence and thus a sequence of

$(0, m, 1)$-nets [Nie92]. For example choosing $m = 3 > t = 0$ we have $N = 2^3 = 8$ and

$$(\lfloor 8 \cdot \phi_2(l + n \cdot 8) \rfloor)_{l=0}^{7} \equiv \{0, 4, 2, 6, 1, 5, 3, 7\}.$$

for $n \in \mathbb{N}_0$. Hence all indices used during the different timesteps n are in fact identical for all m.

Assuming uniform probabilities $p_{i,j} = \frac{1}{|E|}$ the convergence theorem still applies, but more important, stable sorting does not change the state order. It thus follows that in fact the index permutation can be chosen as identity without touching the convergence conditions. The same applies for selecting the initial states $\mathbf{X}_{0,l}$ and it results the simplified, but equivalent algorithm

- $n := 0$
- initialize $\mathbf{X}_{0,l}$ using $x_{l,2}$ for $0 \leq l < 2^m$
- loop
  - sort state vector using a suitable order
  - $n := n + 1$
  - continue chain by computing $\mathbf{X}_{n,l}$ using $\Phi_S(l + n \cdot 2^m)$ for $0 \leq l < 2^m$

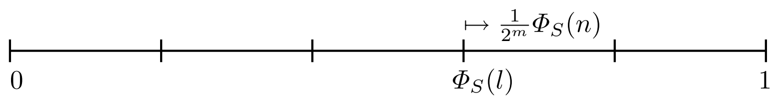
using only the second component of the $(0, 2)$-sequence x_l.

When and Why it Works

The improved convergence of the scheme, which has been observed in many applications (see the references at the beginning of Section 2), now must be caused by the structure of the samples $\Phi_S(l + n \cdot 2^m)$ used to realize the transitions of $\mathbf{X}_{n,l}$ according to P. This can be understood by decomposing the radical inverse (see also [Kel06])

$$\Phi_S(l + n \cdot 2^m) = \Phi_S(l) + \frac{1}{2^m}\Phi_S(n),$$

which reveals an implicit stratification: $\Phi_S(l)$ is an offset with spacing $\frac{1}{2^m}$ depending on the state number l, while the shift $\Phi_S(n)$ is identical for all the intervals at timestep n.



Here the low dimensional setting allows for a misleading interpretation of the samples being a shifted lattice or stratified samples, as the entirety of the $\Phi_S(l)$ for $l = 0, \ldots, 2^m - 1$ in fact must be an $(0, m, 1)$-net and thus an equidistant set of samples.

However, the good performance stems from the property that $\Phi_S(l)$ is a (t, s)-sequence and thus a sequence of (t, m', s)-nets for any m' with $t \leq m' \leq m$. This means that $b^{m'}$ states, that are similar in state space and therefore subsequent by order after sorting, will sample their transition by a (t, m', s)-net, which guarantees for good discrete density approximation. The maximum improvement would be obtained if all 2^m chains were in the same state. The more the states of the chains are separated in state space, the smaller the performance improvements will be.

2.2 Simplified Algorithm in s Dimensions

Using a (t, s)-sequence in base b, which is a sequence of (t, m, s)-nets, the scheme also works in s dimensions: Markov chains, whose states are similar after sorting are guaranteed to sample the transition probability by low discrepancy samples. The simplified algorithm in s dimensions now looks like:

- $n := 0$
- initialize $\mathbf{X}_{0,l}$ using quasi-Monte Carlo points x_l
- loop
  - sort state vector using a suitable order
  - $n := n + 1$
  - continue chain by computing $\mathbf{X}_{n,l}$ using subsequent samples x_l from a (t, s)-sequence

Some simulations require trajectory splitting in order to capture certain local subtle effects. While this already has been addressed in [DLT05, LDT06, LDT07], it in fact can be achieved in a simpler way by just drawing more samples out of the (t, s)-sequence for states to be split.

This is a consequence of the fact that it is not even necessary to simultaneously simulate exactly b^m chains. It is only important to draw subsequent samples from the (t, s)-sequence and to minimize the number b^m of points in the subsequent (t, m, s)-nets in order to enable the maximal performance gain. The choice of the (t, s)-sequence, however, is restricted by the condition that $(0, s)$-sequences only exist for $b \geq s$ and that $m > t$ [Nie92]. Note that other radical inversion based points sets like the Halton sequence or its scrambled variants fulfill properties similar to (t, s)-sequences [Mat98] and will result in similar performance gains.

2.3 Randomization

While there exists no general proof for convergence of the deterministic algorithm in higher dimensions yet, the algorithm becomes unbiased by freshly randomizing the quasi-Monte Carlo points in each time step n [LLT06]. Since this is in fact an instance of padded replications sampling as introduced in [KK02a, KK02b] the argument for unbiasedness becomes simpler than in [LLT06]. Randomization, however, deserves special attention.

The most efficient implementation along the lines of [KK02b] consists of choosing a (t, s)-sequence in base $b = 2$, from which subsequent samples are drawn, which are `XOR`-ed by an s-dimensional random vector. This random vector is freshly drawn after each transition step. However, as random scrambling changes the order in which the points are enumerated, the local properties of the sequences of (t, m, s)-nets are changed, too.

This observation can be taken as an explanation for some of the effects seen in [LLT05]: Sobol and Korobov points used in the array-(R)QMC simulation are worse up to an order of magnitude in variance reduction than their transformed (Gray-code for Sobol, Baker transform for Korobov) counterparts. The explanation for this is found in the structure of the points. The sequence of (t, m, s)-nets extracted from the Sobol sequence is locally worse than its Gray-code variant. The same goes for the Korobov lattice and its transformed variant.

3 Sorting Strategies

In order to have the states as closely together as possible, they have to be enumerated in an order such that the sum of the distances of neighboring states is minimal. This in fact relates to the traveling salesman problem[3], where for a given set of cities and the costs of traveling from one city to another city, the cheapest round trip is sought that visits each city exactly once and then returns to the starting city.

Our problem is very similar except for it is not necessary to return from the last state of the route to the first state. Some techniques are already available to efficiently calculate approximate, but close to optimal, solutions for the traveling salesman problem [DMC91, Rei94]. However, running times of these algorithms are not acceptable in our simulations, as the calculation of the distance matrix alone exhibits an $\mathcal{O}(N^2)$ complexity, while we want to keep the algorithm as close as possible to the $\mathcal{O}(N)$ complexity of classic Monte Carlo methods.

In the following we discuss some possible orders to achieve fast sorting for high-dimensional state spaces.

3.1 Norm of State

The average complexity of quicksort is $\mathcal{O}(N \log N)$, but for certain scenarios even $\mathcal{O}(N)$ algorithms exist, like e.g. radixsort, which, however, requires additional temporary memory. In order to use these one-dimensional sorting algorithms, the multi-dimensional state must be reduced to one dimension. Amongst many choices often some norm $\|\mathbf{X}_{n,l}\|$ is used to define an order on the state space. However, similar norms do not necessarily indicate proximity

[3] This problem already was investigated by Euler in the early 18th century and unfortunately can be shown to be NP-hard [Kar72]. For a historical survey on the problem see Lawler et al. [LLKS85].

in state space. A simple example for this is similar energy of particles in a transport simulation that are located far away in space.

3.2 Spatial Hierarchy

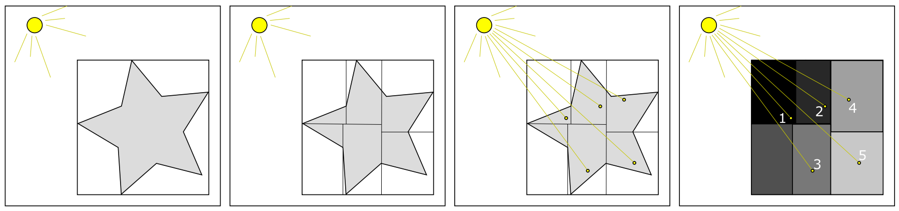


Fig. 1. Sorting the states into the leafs of a spatial hierarchy defines an order of proximity by traversing the hierarchy in in-order.

A second possibility to enable multidimensional sorting is the usage of a spatial hierarchy to define an order on the states [Wie03]. Efficient data structures for this purpose are the BSP-tree [SBGS69, Abr95], its specialized axis aligned subset, the kD-tree [Ben75], or bounding volume hierarchies [RW80, KK86, WK06]. The construction of such binary hierarchies is simple: The space is recursively subdivided using planes selected by some heuristic [WK07]. The construction runs in $\mathcal{O}(N \log N)$ on the average. Traversing the hierarchy in in-order enumerates the leaves in an order of proximity. This traversal becomes trivial, if the tree is left-balanced and in consequence can be stored in an array.

If a spatial hierarchy must be used anyway, for example to accelerate ray tracing, there is no additional construction time for the hierarchy. The particles then are stored as linked lists attached to the leaves of the hierarchy (see Figure 1). Unfortunately the quality of this order is strongly determined by the quality of the spatial hierarchy used for simulation, which is especially problematic if the number of leafs in the hierarchy is much smaller than the number of chains N as this results in several states being mapped to the same leaf.

3.3 Bucket Sorting and Space Filling Curves

In order to guarantee linear time complexity, bucket sorting can be used. In the s-dimensional extension [HLL07] of the simple algorithm sketched in Section 2.1, multidimensional states were sorted into buckets by the first dimension of the state, then the states of each bucket were sorted into buckets according to the second dimension, and so forth. This procedure works well, but has the problem that states close in state space can be separated by the sorting procedure. In addition, a stratification of each dimension has to be used, which induces the curse of dimension in the number of Markov chains to be simulated simultaneously.

We therefore divide the state space into equal voxels, which serve as buckets. The bucket of each state is found by truncating the state coordinates according to the resolution of the voxel grid. Note that this applies for continuous as well as discrete state spaces. Enumerating the voxels by proximity yields the desired order on the states and can be done in linear time in the number of voxels.

Orders that enumerate the voxels by proximity are given by space filling curves [Sag94] like e.g. the Peano curve, Hilbert curve, or H-indexing. These curves guarantee every voxel to be visited exactly once and an overall path length being relatively short. For problems with large geometry, which is the case in our own simulations, this can be even one of the few possibilities to generate fast and memory efficient approximate solutions to the traveling salesman problem [Rei94]. However, these curves are rather costly to evaluate, need to be tabulated to be efficient, or are not available for higher dimensions.

Fortunately, the Z-curve, also known as Lebesgue-curve or Morton order, avoids these problems. Given integer coordinates of a bucket in multidimensional space, its one dimensional Z-curve index is simply calculated by bitwise interleaving the coordinate values (see Figure 2). This is very easy and fast to compute for any dimension and problem size, and requires no additional memory. Unfortunately the results are not as good as for example the Hilbert-curve in a global context. However, the average case partitioning quality and average/worst case logarithmic index ranges are comparably good [Wie03]. Problems can arise in highly symmetrical scenes like Shirley's "Scene 6" (see Figure 6) used for our numerical experiments: States on the walls parallel to the (x, y)-plane will be sorted very well, but the states located on the other two walls parallel to the (y, z)-plane will be visited by the curve in an alternating manner, which can lead to correlation artifacts in some scenarios.

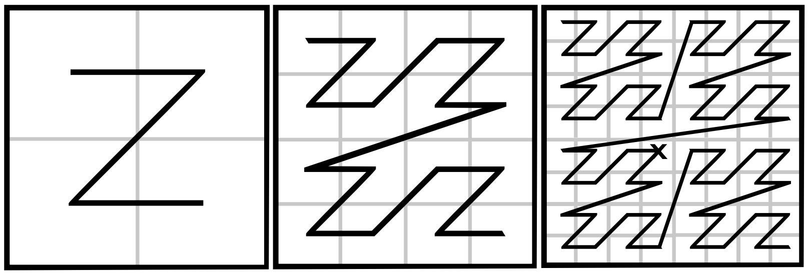


Fig. 2. The Z-curve in two dimensions for 2×2, 4×4, and 16×16 buckets. With the origin $(0, 0)$ top left the point marked by $\times$ has the integer coordinates $(3, 4)$, which corresponds to $(011, 100)_2$ in the binary system. Its binary Z-curve index 100101_2 is computed by bitwise interleaving the binary coordinates.

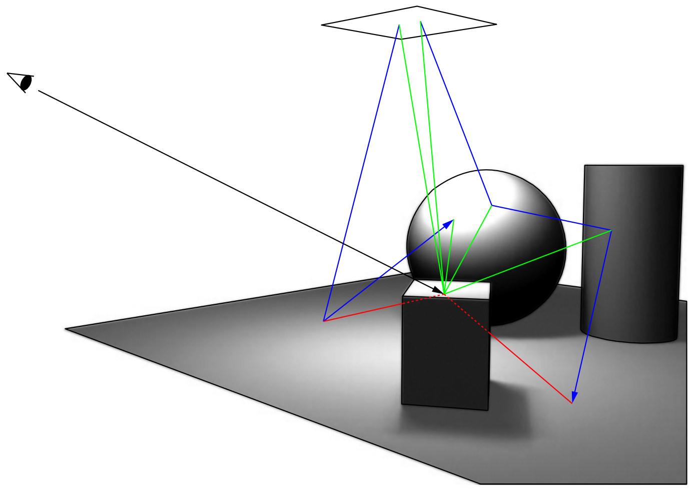


Fig. 3. Sampling transport path space by bidirectional path tracing. Trajectories from the eye and the light sources are generated by Markov chains and connected to determine the transported amount of light.

4 Application to Light Transport Simulation

For numerical evidence, we apply the algorithm developed in the previous sections to light transport simulation for synthesizing realistic images. The underlying integral equation can be reformulated as a path integral [Vea97]. Sampling path space (see Figure 3) corresponds to simulating Markov chains, where the paths are established by ray tracing and scattering events. The initial distribution is determined by the emission characteristics of the light sources and the transition probabilities are given by bidirectional reflectance distribution functions on the surface.

To solve the path integral, one can think of two basic strategies, which are either using high dimensional low discrepancy points or padding low dimensional low discrepancy points [KK02b]. The latter approach fits our findings in Section 2.2, where high dimensional events are composed as subsequent transitions of a Markov chain. As measured in [KK02a] the difference to using high dimensional sequences for Markov chain simulations is small to none, but using padded sequences is computationally faster and requires less implementation effort. It is also simpler for practitioners in rendering industry.

In addition the low dimensional approach allows for much better results, because the stratification properties of (t, s)-sequences or the Halton sequence and its scrambled variants are much better for small dimensions (see Section 2.2).

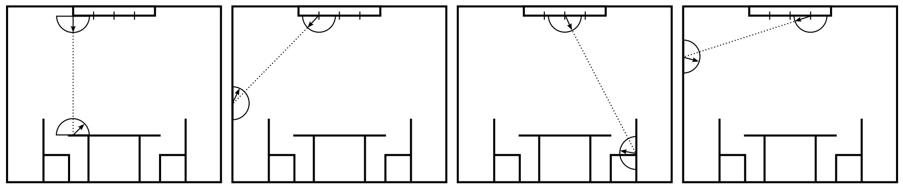


Fig. 4. Photon trajectories are started from the light sources. Upon hitting a surface after tracing a ray, the bidirectional reflectance distribution function is sampled to determine a direction of scattering to continue the path.

4.1 Fredholm or Volterra?

The integral equation underlying light transport can be considered either as a Fredholm or Volterra integral equation, which matters for implementation.

$$L_r(x,\omega) = \int_{S^2_-(x)} f_r(\omega_i, x, \omega) L(x, \omega_i)\ (n(x) \cdot \omega_i) d\omega_i$$

is the radiance reflected off a surface in point x in direction ω, where the domain $S^2_-(x)$ of the integral operator is the hemisphere aligned to the normal $n(x)$ in x (see the illustration in Figure 4). f_r is the bidirectional reflectance distribution function describing the optical surface properties and L is the incident radiance. Using this integral operator results in a Volterra integral equation of the second kind, as the integration domain depends on x.

$$L_r(x,\omega) = \int_{S^2} f_r(\omega_i, x, \omega) L(x, \omega_i)\ \max\{n(x) \cdot \omega_i, 0\} d\omega_i$$

on the other hand results in a Fredholm integral equation of the second kind, as we are integrating over all directions of the unit sphere S^2 independent of x.

Using the latter approach of generating global directions [SKFNC97] and rejecting directions with negative scalar product with respect to the surface normal $n(x)$ is computationally attractive, while the first approach requires to generate directions in the hemisphere that have to be transformed into the local surface frame, which is more expensive. Mappings from the unit square to S^2 or $S^2_-(x)$ are found in [SC94, Rus98, Shi00].

An even more important argument for generating global directions is related to our algorithmic approach (see Section 2.2): By sorting it can happen that two close by surface points with different surface normals (e.g. in a corner) use subsequent samples of a (t, s)-sequence to generate scattering directions. Generating global directions now works fine, whereas generating directions in the two different local frames using subsequent samples will destroy the low discrepancy properties. These discontinuities become clearly visible. Using global directions, however, does not allow for importance sampling according to f_r or the cosine term, which often is a disadvantage and deserves further investigation.

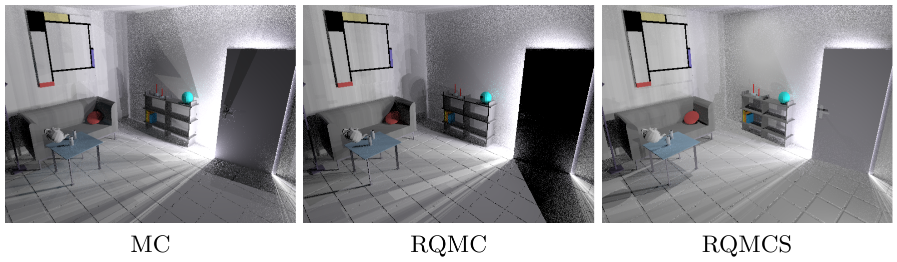


Fig. 5. Visual comparison for the test scene "Invisible Date" using 300 chains for simulation. The only lightsource is not visible as it is placed on the ceiling of the neighboring room. Due to the better distribution of the photons randomized quasi-Monte Carlo (RQMC) outperforms Monte Carlo (MC) visually, as can be seen by the reduced shadow artifacts. RQMC with sorting (RQMCS, using 256^3 voxels for the bucket sort) is even superior as more photons made it into the second room and even the back of the door is lit very well.

4.2 Numerical Evidence

Following the arguments in Sections 2.2 and 2.3, we use the Halton sequence with permutations by Faure [Kel06] randomized by a Cranley-Patterson-rotation [CP76] in order to have unbiased error estimates. For the sorting the Z-curve order (see Section 3.3) worked best in our setting and was used for the following experiments. We further note that we numerically verified that omitting the randomization has no notable effect on the precision of the results. In our numerical experiments we compared four approaches to simulate Markov chains:

MC: Uniform random numbers generated by the Mersenne Twister [SM07] were used for classical Monte Carlo sampling.

RQMC: Used the high-dimensional Halton sequence with permutations by Faure randomized by a Cranley-Patterson rotation, where pairs of components were used to sample the two dimensional emission and scattering events.

lo-dim RQMCS: Used the two-dimensional Halton sequence randomized by a Cranley-Patterson rotation. The Z-curve was used to enumerate the bucket-sorted states.

hi-dim RQMCS: Used the high-dimensional Halton sequence with permutations by Faure randomized by a Cranley-Patterson rotation. The Z-curve was used to enumerate the bucket-sorted states.

In a first experiment the robust global illumination algorithm [KK04, Kol04] was used to compute the path integrals. The resulting graphs are depicted in Figure 6 and display the RMS error to a master solution and the variance averaged over the whole image as well as the pixel-based variance. The numbers were obtained by averaging 10 independent runs for a varying number of Markov chains. The measured numbers only convince for the simple test scene. In the complicated cases even no performance gain over Monte Carlo sampling can be measured, because the number of independent runs is too small and more experiments were not possible due to excessive running times. However, the visual error tells a dramatically different story as can be seen in Figure 5, where a clear superiority of the new algorithm in even very difficult settings becomes obvious. This case is not an exception, but can be observed for many test cases. It only emphasizes that standard error measures are not appropriate error measures for visual quality, which is a known but unsolved problem in computer graphics.

Figure 7 shows measurements for a very difficult light transport problem, where we directly traced photons from the light sources and connected their final path segment to the camera (one technique of bidirectional path tracing [Vea97]). Opposite to the above measurements only one number of simultaneously simulated Markov chains is considered. Now a sufficient amount of experiments was computationally feasible and the superiority of the new algorithm became clearly visible.

5 Conclusion

We successfully simplified the algorithms to simultaneously simulate Markov chains and provided intuition when and why sorting the states can improve convergence. In addition the algorithm no longer is bounded by the curse of dimension and there is no restriction to homogenous Markov chains, because the simulation just can use transition probabilities $P \equiv P_n$ that can change over time.

Our experiments also revealed that not all (t, s)-sequences or radical inversion based points sequences are equally good. This deserves further characterization.

The algorithm would be even simpler, if rank-1 lattice sequences could be applied. The constructions so far, however, lack the properties of (t, s)-sequences that are required for the improved performance. In the future we will investigate whether it is possible to construct suitable rank-1 lattice sequences.

As we are using global directions, i.e. integrate over products of spheres, it is also interesting to establish connections to recent research in that direction [KS05].

Acknowledgments

The authors would like to thank mental images GmbH for support and for funding of this research. Jan Rzehak created the illustration in Figure 3.

References

[Abr95] M. Abrash. BSP Trees. *Dr. Dobbs Sourcebook*, 20(14):49–52, 1995.

[Ben75] J. Bentley. Multidimensional Binary Search Trees used for Associative Searching. *Commun. ACM*, 18(9):509–517, 1975.

[BNN$^+$98] P. Bekaert, L. Neumann, A. Neumann, M. Sbert, and Y. Willems. Hierarchical Monte Carlo Radiosity. *Eurographics Rendering Workshop 1998*, pages 259–268, June 1998.

[CP76] R. Cranley and T. Patterson. Randomization of number theoretic methods for multiple integration. *SIAM Journal on Numerical Analysis*, 13:904–914, 1976.

[DLT05] V. Demers, P. L'Écuyer, and B. Tuffin. A Combination of Randomized quasi-Monte Carlo with Splitting for Rare-Event Simulation. In *Proceedings of the 2005 European Simulation and Modelling Conference*, pages 25–32. SCS Press, 2005.

[DMC91] M. Dorigo, V. Maniezzo, and A. Colorni. Positive Feedback as a Search Strategy. Technical Report 91016, Dipartimento di Elettronica e Informatica, Politecnico di Milano, Italy, 1991.

[HLL07] R. El Haddad, C. Lécot, and P. L'Écuyer. Quasi-Monte Carlo Simulation of Discrete-Time Markov Chains in Multidimensional State Spaces. In A. Keller, S. Heinrich, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*, page in this volume. Springer, 2007.

[Kar72] R. Karp. Reducibility among Combinatorial Problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations, Advances in Computing Research*, pages 85–103. Plenum Press, 1972.

[Kel06] A. Keller. Myths of Computer Graphics. In H. Niederreiter, editor, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 217–243. Springer, 2006.

[KK86] T. Kay and J. Kajiya. Ray Tracing Complex Scenes. *Computer Graphics (Proceedings of SIGGRAPH 86)*, 20(4):269–278, August 1986.

[KK02a] T. Kollig and A. Keller. Efficient Bidirectional Path Tracing by Randomized Quasi-Monte Carlo Integration. In H. Niederreiter, K. Fang, and F. Hickernell, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2000*, pages 290–305. Springer, 2002.

[KK02b] T. Kollig and A. Keller. Efficient Multidimensional Sampling. *Computer Graphics Forum*, 21(3):557–563, September 2002.

[KK04] T. Kollig and A. Keller. Illumination in the Presence of Weak Singularities. In D. Talay and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 245–257. Springer, 2004.

[KL99] F. El Khettabi and C. Lécot. Quasi-Monte Carlo Simulation of Diffusion. *J. Complexity*, 15(3):342–359, 1999.

[Kol04] T. Kollig. *Efficient Sampling and Robust Algorithms for Photorealistic Image Synthesis*. PhD thesis, University of Kaiserslautern, Germany, 2004.

[KS05] F. Kuo and I. Sloan. Quasi-Monte Carlo Methods can be efficient for Integration over Products of Spheres. *J. Complexity*, 21(2):196–210, 2005.

[LC98] C. Lécot and I. Coulibaly. A quasi-Monte Carlo Scheme using Nets for a linear Boltzmann Equation. *SIAM J. Numer. Anal.*, 35(1):51–70, 1998.

[LDT06] P. L'Écuyer, V. Demers, and B. Tuffin. Splitting for Rare-Event Simulation. In *Proceedings of the 2006 Winter Simulation Conference*, pages 137–148, 2006.

[LDT07] P. L'Écuyer, V. Demers, and B. Tuffin. Rare Events, Splitting, and quasi-Monte Carlo. *ACM Transactions on Modeling and Computer Simulation*, 17(2): Art. No. 9, 2007.

[Léc89a] C. Lécot. A Direct Simulation Monte Carlo Scheme and Uniformly Distributed Sequences for Solving the Boltzmann Equation. *Computing*, 41(1-2):41–57, 1989.

[Léc89b] C. Lécot. Low Discrepancy Sequences for solving the Boltzmann Equation. *Journal of Computational and Applied Mathematics*, 25(2):237–249, 1989.

[Léc91] C. Lécot. A quasi-Monte Carlo Method for the Boltzmann Equation. *Mathematics of Computation*, 56(194):621–644, 1991.

[LL02] P. L'Écuyer and C. Lemieux. Recent Advances in Randomized quasi-Monte Carlo methods. In M. Dror, P. LEcuyer, and F. Szidarovszky, editors, *Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications*, pages 419–474. Kluwer Academic Publishers, 2002.

[LLKS85] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. John Wiley, 1985.

[LLT05] P. L'Écuyer, C. Lécot, and B. Tuffin. A Randomized quasi-Monte Carlo Simulation Method for Markov Chains. Technical report g-2006-64, to appear in Operations Research, GERAD, Université de Montréal, 2005.

[LLT06] P. L'Écuyer, C. Lécot, and B. Tuffin. Randomized quasi-Monte Carlo Simulation of Markov Chains with an Ordered State Space. In H. Niederreiter and D. Talay, editors, *Monte Carlo and quasi-Monte Carlo Methods 2004*, pages 331–342. Springer, 2006.

[LT04a] C. Lécot and B. Tuffin. Comparison of quasi-Monte Carlo-Based Methods for the Simulation of Markov Chains. *Monte Carlo Methods and Applications*, 10(3-4):377–384, 2004.

[LT04b] C. Lécot and B. Tuffin. Quasi-Monte Carlo Methods for Estimating Transient Measures of Discrete Time Markov Chains. In H. Niederreiter, editor, *Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing 2002*, pages 329–344. Springer, 2004.

[Mat98] J. Matoušek. On the L_2-discrepancy for anchored boxes. *J. Complexity*, 14(4):527–556, 1998.

[MC93] W. Morokoff and R. Caflisch. A Quasi-Monte Carlo Approach to Particle Simulation of the Heat Equation. *SIAM Journal on Numerical Analysis*, 30(6):1558–1573, 1993.

[Nie92] H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM, Philadelphia, 1992.

[Rei94] G. Reinelt. *The Traveling Salesman - Computational Solutions for TSP Applications*. Springer, 1994.

[Rus98] D. Rusin. Topics on Sphere Distributions, http://www.math.niu.edu/~rusin/known-math/95/sphere.faq, 1998.

[RW80] S. Rubin and J. Whitted. A 3-dimensional representation for fast rendering of complex scenes. *Computer Graphics (Proceedings of SIGGRAPH 80)*, 14(3):110–116, 1980.

[Sag94] H. Sagan. *Space-Filling Curves*. Springer, 1994.

[SBGS69] R. Schumacker, R. Brand, M. Gilliland, and W. Sharp. Study for Applying Computer-Generated Images to Visual Simulation. Technical report AFHRL-TR-69-14, U.S. Air Force Human Resources Laboratory, 1969.

[SC94] P. Shirley and K. Chiu. Notes on Adaptive Quadrature on the Hemisphere. Technical Report TR-411, Dept. of Computer Science Indiana University, 1994.

[Shi00] P. Shirley. *Realistic Ray Tracing*. AK Peters, Ltd., 2000.

[SKFNC97] L. Szirmay-Kalos, T. Fóris, L. Neumann, and B. Csébfalvi. An Analysis of Quasi-Monte Carlo Integration Applied to the Transillumination Radiosity Method. *Computer Graphics Forum*, 16(3):271–282, 1997.

[SM07] M. Saito and M. Matsumoto . SIMD-oriented Fast Mersenne Twister: A 128-bit Pseudorandom Number Generator. In A. Keller, S. Heinrich, and H. Niederreiter, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2006*. Springer, in this volume.

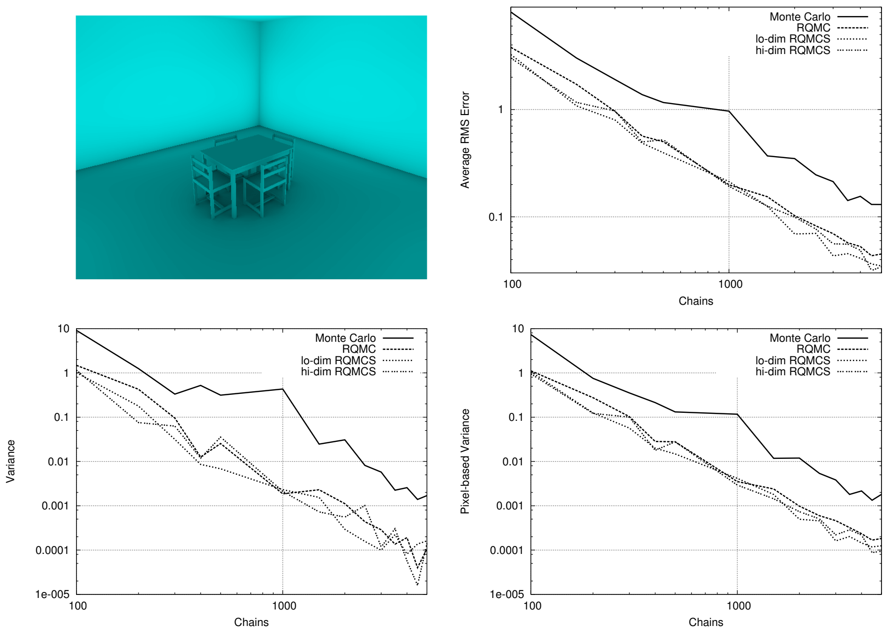
[Sob67] I. Sobol'. On the Distribution of Points in a Cube and the approximate Evaluation of Integrals. *Zh. vychisl. Mat. mat. Fiz.*, 7(4):784–802, 1967.

[Vea97] E. Veach. *Robust Monte Carlo Methods for Light Transport Simulation*. PhD thesis, Stanford University, 1997.

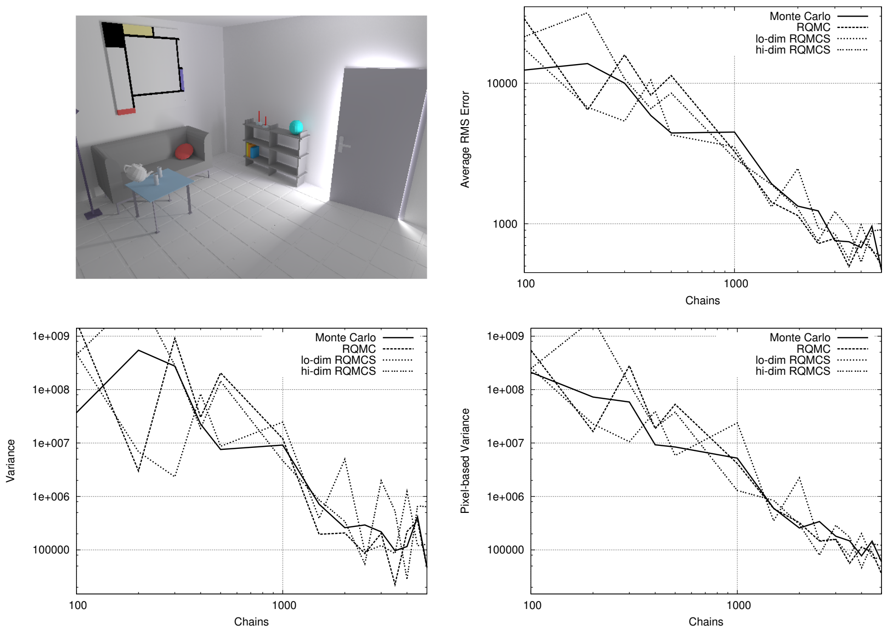
[Wie03] J.-M. Wierum. *Anwendung diskreter raumfüllender Kurven - Graphpartitionierung und Kontaktsuche in der Finite-Elemente-Simulation*. PhD thesis, Universität Paderborn, 2003.

[WK06] C. Wächter and A. Keller. Instant Ray Tracing: The Bounding Interval Hierarchy. In T. Akenine-Möller and W. Heidrich, editors, *Rendering Techniques 2006 (Proc. of 17th Eurographics Symposium on Rendering)*, pages 139–149, 2006.

[WK07] C. Wächter and A. Keller. Terminating Spatial Partition Hierarchies by A Priori Bounding Memory. Technical report, Ulm University, 2007.

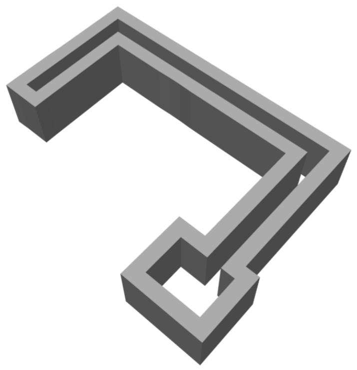


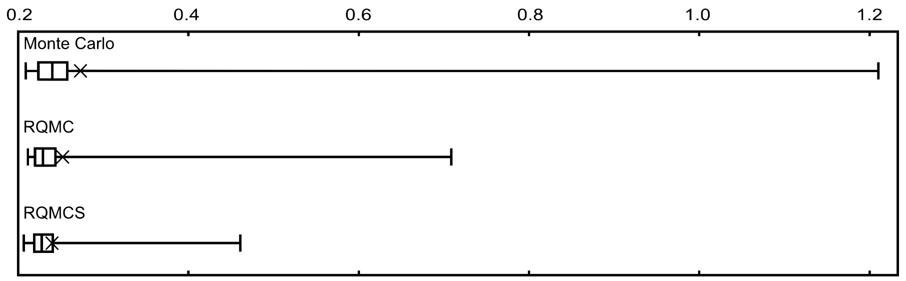
a) Test scene "Shirley 6".

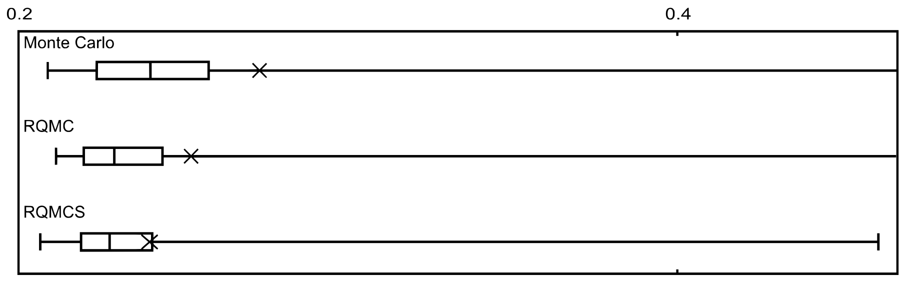


b) Test scene "Invisible date".

Fig. 6. RMS error, global, and per pixel variance for an a) simple and b) more complicated light transport setting.







	L_1 per camera	L_2 per camera	L_1 per pixel	L_2 per pixel
Monte Carlo	0.0626524	0.02368870	0.202377	463.397
RQMC	0.0407531	0.00773514	0.162835	62.0994
RQMCS	0.0247592	0.00178907	0.138360	9.79994

Fig. 7. Schematic view of the labyrinth test scene, where floor and roof have been removed for illustration. The camera is situated in the room at the bottom, while a light source is located on the other end of the connecting tunnel. The graphs show the Box-and-Whisker plots and the average amount (marked by $\times$) of the total radiance received by the camera for 1048576 simulated light paths for 50 independent realizations using each technique. The lower graph enlarges the interesting part of the upper graph. The table finally displays the deviation from a master solution using the L_1- and L_2-norm (variance) for the total radiance received by the camera and received by each pixel (256×256 pixels resolution) averaged over the 50 independent realizations. In this difficult setting the new method (RQMCS) is clearly superior.