CHAPTER

3 Massively parallel kinetic Monte Carlo simulations

Abstract

In this chapter, a parallel lattice based Kinetic Monte Carlo simulation is developed that runs on a GPGPU board and includes Coulomb like particle-particle interactions. The performance of this computationally expensive problem is improved by modifying the interaction potential due to nearby particle moves, instead of fully recalculating it. This modification is achieved by adding dipole correction terms that represent the particle move. Exact evaluation of these terms is guaranteed by representing all interactions as 32-bit floating numbers, where only the integers between -2^{22} and 2^{22} are used. We validate our method by modeling the charge transport in disordered organic semiconductors, including Coulomb interactions between charges. Performance is mainly governed by the particle density in the simulation volume, and improves for increasing densities. Our method allows calculations on large volumes including particle-particle interactions, which is important in the field of organic semiconductors.

N. J. van der Kaap and L. J. A. Koster, *Massively parallel kinetic Monte Carlo simulations of charge carrier transport in organic semiconductors*, J. Comput. Phys. 307 (2016) 321–332

3.1 Introduction

Kinetic Monte Carlo (KMC) methods are widely used for simulating the time evolution and equilibrium behaviour of systems of particles. These methods have proven to be valuable techniques for describing chemical reactions [101], physical systems [102, 103], layer growth [104–106] surface reactions [107–109], defect mobility [110, 111] and biological networks [112, 113]. KMC models systems by calculating all possible transition rates that transform a system from one state into another, followed by randomly picking one of these transitions for execution [114]. Depending on the system under consideration, it may take billions of transitions before steady state is reached. This computational burden is further exacerbated by the large simulation volumes that are required to reduce measurement errors or by inclusion of particle-particle (p-p) interactions. These interactions may for instance originate from Coulomb interaction between charges [115] or elastic forces between other particles [116].

Various methods exist for reducing the extra workload due to these interactions. One method is by applying the fast multi-pole method (FMM) [117], where distant charges are clustered to decrease the number of interactions that need to be calculated [118, 119]. A great benefit of this method is the existence of very precise error bounds that allow for a custom choice between precision and speed. In the work of Van der Holst *et al.* long range Coulomb interactions are calculated by solving the discrete 1D Poisson equation, which is much faster than calculating direct interactions [120]. Because the short range interactions are still calculated directly, they introduce a method to prevent double counting of charges in the Poisson equation. For some types of interaction, accurate estimations of the overall interaction potential can be made. For simulations of hetero epitaxial growth, Schulze *et al.* determined an approximate, easy to calculate, upper bound for the elastic energy of a certain configuration [116, 121]. This provides an upper limit for the transition rate to the corresponding state. Once this transition is selected for execution, the real interaction is calculated, and a rejection algorithm determines whether the transition is performed.

Instead of directly reducing the computational effort in calculating the interaction potential, the simulation itself can also be adapted. In the next reaction method (NRM) [74], transition rates are only updated when the corresponding particle moved. This reduces the amount of interaction calculations dramatically, but the transition rates become insensible for nearby particle movement. Another approach for increasing performance is numerically solving an approximation of a master equation [95, 122].

Because this type of simulation operates on particles densities rather than on explicit particles, the interaction potential can be solved using a finite differences approach. However, this type of simulation is not able to capture all physical detailed that can be acquired using a KMC simulation.

An approach to accelerate KMC simulations of large systems is to perform the calculations on parallel computers: the simulation volume is divided into multiple sections that are each treated as a single simulation [123–126]. It is important that the simulation time of all sections remains synchronized. This can either be achieved by independently running all sections for a fixed period of time [124] or by equalizing all transition rates by introducing null-transitions [125]. Particles that leave their section are simulated in a ghost region, until a periodic synchronization event communicates all particles to their corresponding segments [123]. Rigorous methods use a roll-back procedure to restart the simulation at the time that the boundary transition was received [126, 127]. Computationally less expensive non-rigorous algorithms deal with these transitions periodically, introducing an error with respect to the serial algorithm [124, 128].

One special type of parallel computer is the General Purpose GPU (GPGPU), an interface that supports thousands of simultaneous threads. Previous work on lattice based KMC algorithms that were implemented on a GPGPU used a similar approach as the CPU version, but did not consider p-p interactions [128, 129]. The work of Plimpton *et al.* was one of the first implementations of the parallel KMC code on a GPGPU for simulating thin film growth, and used a similar method as the parallel KMC algorithms discussed above [129]. Another example of GPGPU on lattice based KMC is the simulation framework that was designed by Arampatzis *et al.* [128]. In their general approach, they introduce an architecture that supports simultaneous independent execution of segments for a short time span on different processors. They also provide a thorough error analysis of the non-rigorous parallel KMC algorithm. An example of a rigorous off-lattice KMC simulation is the work by Andersson *et al.* [130]. They described a 2D system of hard disks that can move in free space. During each iteration, all disks make a trial move. If the move is accepted by the metropolis acceptance criterion, it is executed. Boundary conflicts are avoided by rejecting trial moves that leave a cell. The boundaries of different regions are changed periodically, to allow particle movement throughout the entire volume.

In this chapter, we study a method for implementing a parallel lattice KMC simulation including long range p-p interactions that runs on a GPGPU. We use a

synchronous method that is similar to the work by Martínez *et al.* [125]. Boundary conflicts are solved non-rigorously by using a sublattice approach [131]. We use this method because the inclusion of p-p interactions is computationally expensive, and roll-back mechanisms decrease performance further. Typically, the bottleneck in GPGPU is caused by copying CPU data back and forth to the GPU interface. We prevent this by performing the entire procedure on a GPGPU interface.

As discussed above, proper inclusion of these interactions requires segment synchronization after every iteration. In order to minimize simulation errors due to incorrect values of the interactions potential, segments are synchronized after every iteration. Although relatively long communication times between remote processors may hinder this process in typical parallel computers, this is not the case for GPGPU architectures. Still, full recalculation of the interaction potential after each iteration is time consuming. Instead, the algorithm corrects the current potential by adding dipole contributions for every nearby charge that hopped during the previous iteration. Full updates of the interaction potential are only required for the grid points that are related to charges that hopped during the last iteration. Accumulative rounding errors that arise due to repetitive addition and subtraction are solve this by rounding all interaction potentials to a uniformly spaced range of floating point numbers

Finally, we validate the simulation by modelling charge transport in organic semiconductors [12]. Charge transport through these materials is characterized by an effective mobility μ, that depends on the degree of disorder σ, temperature T, the electric field F and the particle density n. We compared the results with and without Coulomb interactions to numerically solving the corresponding master equation [95], and investigated the performance of the algorithm.

3.2 Kinetic Monte Carlo simulation

The simulation is implemented on a cubic grid of sites, that contains N particles. Each particle has transition rates to its six nearest neighbours. Other transitions are ignored, because the rate expressions in Sec. 3.3 contain a term that decreases exponentially with distance. To prevent multiple particles from occupying the same lattice site, rates between occupied sites are set to zero.

The transition rates depend on the energy difference $\Delta\epsilon_{ij} = \epsilon_j - \epsilon_i$ between the initial state i and final state j. Negative values of $\Delta\epsilon_{ij}$ indicate a decrease in the interaction potential energy of the particle and therefore result in higher transitions

rates than positive values of $\Delta\epsilon_{ij}$. Effects that contribute to the interaction potential are differences in static site energies, external fields and p-p interactions.

3.2.1 General algorithm

The simulation volume is divided into S spatial segments, where segment s_i contains N_i particles $\{p_{ij}|j = 0...N_i - 1\}$ with transition rates $\{w_{ijk}|k = 0...5\}$. During one simulation step, each segment randomly selects a transition w_i with probability w_{ijk}/W_i, where $W_i = \sum_{j,k} w_{ijk}$. The dimensions of each segment are arbitrary, but for simplicity we assume them to be cubic and of equal size. The reason for this is that we expect a homogeneous particle concentration throughout the simulation volume, which results in a constant number of particles in each segment. Moreover, by setting the segment size to the maximum interaction distance, interacting particles are always in the same segment or in an adjacent segment. For cases with inhomogeneous concentrations, however, a custom distribution of variously sized segments is also possible.

Boundary conflicts are prevented by using a checker board approach with 8 colors $\{c_i|i = 0...7\}$ [130, 131]. At the beginning of each iteration, a random sequence of these colors is drawn. If transitions from segments with different c_i result in a conflict, the transition from the segment with lowest position of c_i is rejected. Particles that leave their segment are transferred to a new segment at the end of the KMC cycle.

After the simulation volume has been initialized by randomly placing particles throughout the lattice, the KMC code is executed. The entire algorithm is executed on the GPU, because copying large volumes of information back and forth between CPU and GPU memory is computationally expensive. The flowchart of the simulation is shown in Fig. 3.1. A CPU function repeatedly starts the main GPU function until a preset amount of simulation time has passed. This GPU function performs a number of KMC cycles, and has a separate thread i for each segment s_i. The sequence of operations is as follows:

1. Calculate all w_i and W_i. This is done by starting another GPU function that is explained below.

2. Determine $W_{max} = \max(W_i)$ for all segments. This is done in the main GPU function by a parallel reduction procedure of the S threads.

3. Randomly select transition w_i with probability $\frac{W_i}{W_{max}}$ or the null-transition with probability $\frac{W_{max}-W_i}{W_{max}}$. This is again done in the main function, where every thread draws a random number rnd and rejects r_i if $rnd > \frac{W_i}{W_{max}}$

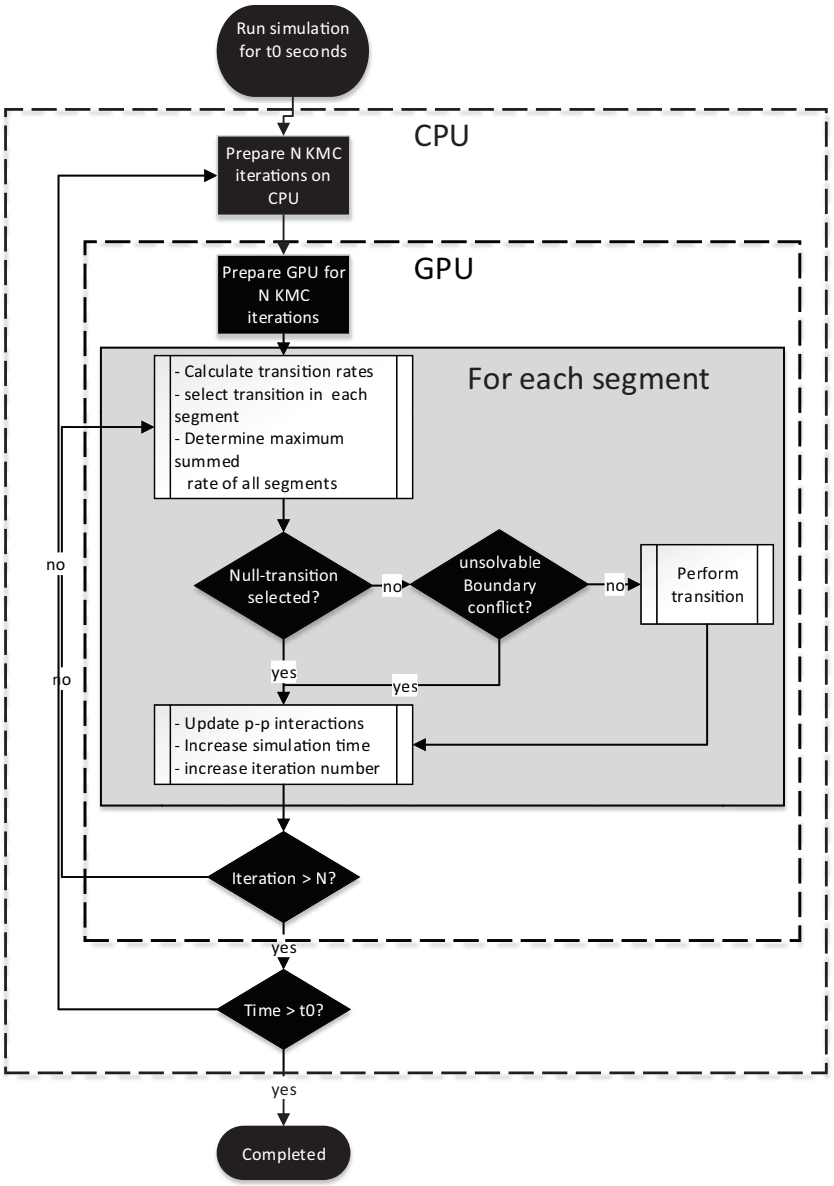

Figure 3.1: Flowchart of a complete KMC iteration. The CPU monitors the simulation time, and executes the GPU procedure until enough time has passed. The GPU procedure performs a fixed number of KMC iterations during one call, in order to minimize overhead that occurs due to communication between the CPU and GPU. First, the transition rates in each segment are calculated parallelly. Next, all segment selects either a transition, which is performed if it is no null-transition and has no unsolvable boundary conflict. Finally, p-p interactions and simulation time are updated.

4. If the null-transition was not selected, mark the locations corresponding to the transition. This procedure is performed in the main GPU function. If a location has already been marked by another thread, a conflict occurred.

5. In case of a conflict, retrieve the position l in the color permutation of the color that corresponds to the segment. The segment with the highest value of l gets to execute its selected transitions. The other transitions are rejected. This action is again performed in the main GPU function.

6. When the null-transition was not selected, perform the transition. This procedure is performed by a separate GPU function that (i) updates the variables containing the locations of the particle and its possible destinations, (ii) updates the transition rates of the particle, (iii) resets the transition rate of particles that are adjacent to the old particle position, and (iv) resets the transition rates of adjacent particles to the new location to zero.

7. Increase simulation time with $-\frac{\ln(rnd)}{W_{max}}$ and calculate μ, where $rnd \in (0.0, 1.0]$. This step is performed by only one thread, because the simulation time in all segments is synchronized.

Calculating W_i and selecting w_i is the most time consuming part of the simulation (when p-p interactions are not taken into account). Optimizing this function is important to improve the overall performance. A single GPU function for calculating w_i and W_i would require thousands of inter-communicating threads. Because our hardware limits this number to 1024, the calculations are split in two levels, as shown in Fig. 3.2.

The first level operates on sets $\{w_{ijk} | j = 32l...32l+31, k = 0...5\}$ of 32 transitions, where i refers to the segment and the maximum value of $l = \frac{N_i - 1}{32} + 1$. For each of these sets, the GPU kernel reduces the transitions to an intermediate sum of transition rates W_{ilk}, and randomly selects a transition w'_{ilk}. The kernel is run in blocks of 256 threads that can simultaneously operate on 8 sets. A large number of these blocks is required to performs all calculations in the simulation volume. Before this procedure is started, an array is build that assigns all sets to a block. A single set of transition rates is processed by 32 parallel threads $m \in \{0...31\}$ that perform the following tasks:

1. All threads load the interaction potentials and site energies from memory.

2. Thread m calculates transition rate $\omega_m = (w_{ilk})_m$.

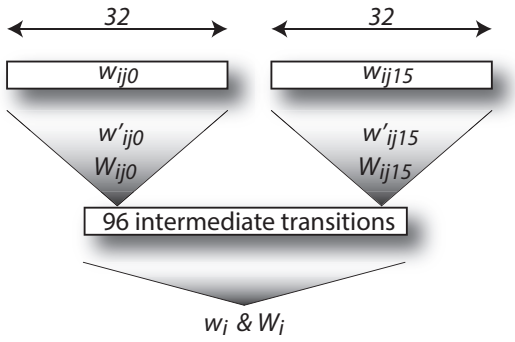

Figure 3.2: Selecting rate w_i and calculating ratesum W_i for block i is split in two levels. The upper level reduces sets of 32 transition rates into intermediate transitions w'_{ijk} with rates W_{ijk}. The lower level selects one of the intermediate transitions and calculates the total transition rate for block i. Both levels are executed by different GPU kernels. For organic semiconductors, it is safe to assume a maximum occupation of 1 in 8 sites. This limits the number of entries at to second level to 96.

3. Each thread m determines the partial sum $\sigma_m = \sum_{i=0}^{m} \omega_m$ by using a parallel scan algorithm.

4. Set the sum of all rates $W'_{ilk} = \sigma_{31}$.

5. Given a random number $c_{i,0} \in [0.0, 1.0)$, find m for which $\sigma_{m-1} \leq \sigma_{31} \cdot c_{i,0} < \sigma_m$. This selects the intermediate transition $w'_{ilk} = w_{ilm}$. Only one unique random number is required for the first level calculations of the sets corresponding to the same segment, because in the end only one transition will be selected.

The second level reduces the intermediate transitions and rates into a W_i and w_i. The total number of interactions per block is now reduced by a factor 32. When segment of $16 \times 16 \times 16$ nodes are used, there are 768 intermediate transitions left for the entire simulation volume. For organic semiconductors, it is safe to assume that the maximum occupancy of each segment is one in eight sides, reducing the number of interactions to 96. The GPU kernel consists of S blocks of each 96 parallel threads $(m, k) \in \{(0...15, 0...5)\}$ that perform the following tasks:

1. Each thread (m, k) loads w'_{imk} and determines the partial sum $\sigma'_{imk} = \sum_{i=0}^{m} w'_{imk}$ by using an efficient parallel scan algorithm.

2. Set the sum of all rates $W_i = \sigma'_{31}$.

3. Given a random number $c_{i,1} \in [0.0, 1.0)$, determine (m, k) for which $\sigma'_{i(m-1)k} \leq \sigma_{31} \cdot c_{i,1} < \sigma'_{imk}$. This selects the transition $w_i = w'_{imk}$.

3.2.2 Correctness

A sufficient condition for the convergence of a KMC simulation is detailed balance [132]:

$$p_i R_{ij} = p_j R_{ji} \tag{3.1}$$

Here, p_i is the probability that the system is in state i, and R_{ij} is the rate at which the system transform to state j. Detailed balance is satisfied when the distribution reaches its thermodynamic equilibrium. In our case, this should only occur when no electric field is applied, as particles do not reach their equilibrium position otherwise. For the Miller-Abrahams hopping rates that are used in Sec 3.3, the hop-rates depend exponentially on the energy difference between the initial and final state. No matter what sequence of states is traversed, a cyclic travel through a random number of states will always result in a net energy difference of zero. The product of the hop-rates that correspond to any cyclic interval will therefore always be equal to 1. This means that Kolmogorov's criterion is satisfied, [133] and that the simulation satisfies detailed balance. The inclusion of p-p interactions does not change this principle, because the overall energy level of a state is determined by the sum of the static energy level and the interaction energies with surrounding particles. The energy level is characteristic for the state, and a cyclic traversal through random states will always result in zero energy difference. Under the condition that boundary conflicts are handled rigorously, the work of Martínez *et al.* [125] shows that the same master equation is solved exactly when using a parallel simulation with null-transitions.

Similar work by Martínez *et al.* [131] introduces a non-rigorous method to prevent boundary conflicts, and gives a quantification of the error that is introduced by this method compared to a simulation where conflicts are handled rigorously. The analysis shows that the deviation from a serial simulation decreased with increasing the segment size. Our approach deals with conflicts in a similar way, and will therefore result in a similar error. The work in Sec. 3.3 is based on segments of 4096 sites. Although the number of particles is lower, the probability that a selected particle is located next to the interface is identical. Therefore, the error in our simulation is comparable to the case with 4096 particles per processor. This results in an error that

deviates at most 0.5 % from the serial simulation (see figure 3 and 4 of [131]). This is well within the error margins that are important for our applications.

3.2.3 Particle-Particle interactions

The interaction potential at location $\vec{x}$ due to p-p interactions for a particle at $\vec{x}_j$, in a system of N particles located at $\vec{x}_i$ is given by

$$\Phi_j(\vec{x}) = \sum_{i \neq j} \frac{c_j \cdot c_i}{|\vec{x}_i - \vec{x}|}, \tag{3.2}$$

where c_i is the sign of the particle charge. Calculating these interactions is a challenging task: the scaling with N^2 causes direct summation of the terms to be computationally expensive, even for a relatively low number of particles . A cutoff radius r that ignores particles for which $|\vec{x}_i - \vec{x}| > r$ reduces the required amount of work, but too small values of r affect the simulation outcome [77]. Our approach also employs a cutoff radius, but further reduces computational work by updating $\Phi_j(\vec{x})$ after a nearby particle moved, instead of by fully recalculating it.

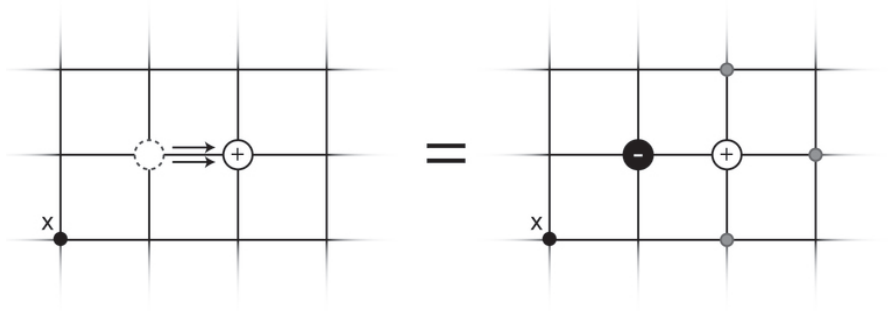

Figure 3.3: The interaction potential at location x after the hole movement is obtained by adding a dipole term to the previous value.

Figure 3.3 shows the principle of the approach. The change in interaction potential of location X with positive c_j, is given by a dipole that has a negative pole at the old location and a positive pole at the new location. Although the locations with a gray point still require a full recalculation using Eq. 3.2, the total number of calculations is dramatically reduced. For each particle, the interaction potential must be calculated at 7 different positions: the current location of the particle as well as its possible destinations. When considering a single particle transition in a system of N interacting

particles, a full update will thus require $7N^2$ interaction calculations, whilst the dipole update method requires only $9N$ calculations.

Although the dipole correction method is not new, it has not been applied yet in parallel KMC simulations to the knowledge of the authors. The spatial ordering of particles and the synchronized nature of our method are beneficial when implementing a parallel version of this method. The segmentation in cubes with dimensions of the cutoff radius allows fast communication of dipole terms to it's interaction particles. And the synchronize nature of the simulation causes multiple dipole terms to be processed simultaneously, which is computationally favourable.

The interactions are included into the simulation as follows. The p-p interactions are calculated after all particle movements have been executed. For segment i, the following operations are performed (Fig. 3.4):

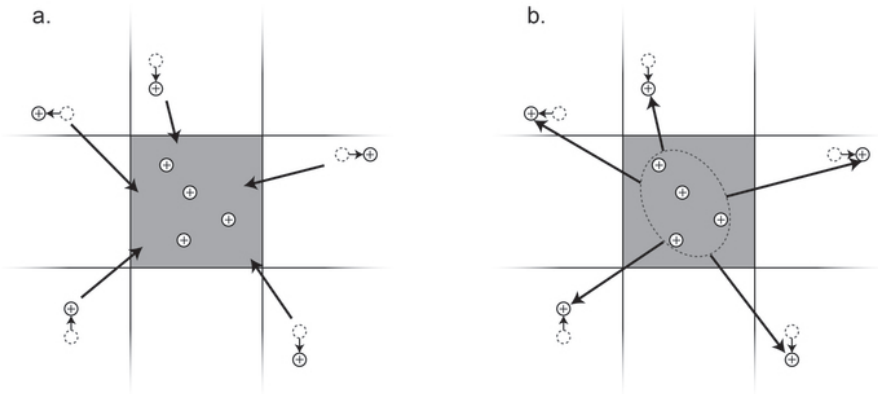

Figure 3.4: The stages for calculating the p-p interactions. (a) shows the dipole update stage: all surrounding charge movements are gathered and translated into dipoles, these are then added to the interaction potentials of all local charges. (b) shows the full update stage of charges that were involved in a hop: the contribution of all local charges to the interaction potential of locations that are related to the moved charge is calculated.

1. Gather all movements from the surrounding segments. These are all the contributions that are within the cut-off radius of the charges inside segment i.

2. Perform the dipole update: iterate through all particles of segment i, and add

a dipole contribution for each of the gathered moves if it is within the cut-off radius.

3. Perform the full update: iterate through the destination locations of all gathered moves, and calculate the contribution of all particles in segment i to the interaction potential moved charges.

Implementing this approach results in problems due to accumulative rounding errors due to repetitive summation and addition of interaction potentials. These rounding errors originate from the non-uniform interval between consecutive floating point numbers. We resolve this issue by transforming all interaction terms to a subset of the floating point numbers that does have uniform spacing. This results in exact calculations, whilst maintaining fast floating point operations. The subset of choice is the integer range of the floating point numbers: $[-2^{22}, 2^{22}]$. Single precision floating point number cannot fully represent integers outside this range, and rounding to the set can be done using intrinsic operations. In order to obtain adequate precision, the maximum possible interaction potential of a particle must map to 2^{22}. This maximum interaction depends on the interaction distance, and the maximum expected particle density. For the calculations done in the next section on segments of $16 \times 16 \times 16$, a cutoff radius of 16 lattice spacings is used. Assuming a maximum filling of 12.125 % and $r = 16a$, the maximum expected interaction is 66.75, which results in a Unit in Last Place of 7.96×10^{-6}.

Even though the calculation of p-p interactions is accelerated by using our method, further increasing the cutoff radius is computationally expensive. This issue can be resolved by replacing the direct calculations of the most distant interactions by modern approximative schemes like the FMM. A good candidate is the massively parallel implementation by Lashuk *et al.* [119]. Because the impact of distant particle movement on the interaction potential is minimal, the long range approximations need to be updated occasionally. Although use of the FMM is a good replacement for the direct calculation of very long interactions, nearby particles still require direct evaluation.

3.3 Validation

We validate our method by modeling the transport of positively charged particles (holes) in disordered organic semiconductors [134]. Charge transport is characterized

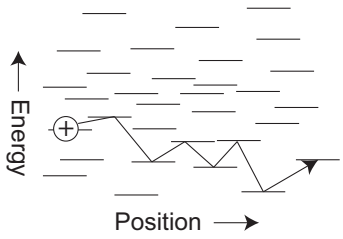

Figure 3.5: Phonon activated hopping of positively charges holes through a Gaussian distributed density of states. An applied electric field causes the charge to move right. The disordered nature of the density of state causes charge transport that changes with temperature, charge carrier density and electric field.

by phonon assisted hopping of charges between localized sites (Fig. 3.5) [12]. Each site i has a static, gaussianly distributed, on-site energy $\epsilon_{0,i}$ with standard deviation σ. We assume Miller-Abrahams hopping, because its results are well known from the literature [16, 18, 135]. The hopping rate ν_{ij} for a hole from site i to site j is given by Eq. 1.3:

$$\nu_{ij} = \nu_0 \exp\left(-\frac{(\epsilon_j - \epsilon_i) + |(\epsilon_j - \epsilon_i)|}{2k_B T}\right). \tag{1.3}$$

where ν_0 is a constant rate, α_{ij} is the inverse localization length, a_{ij} is the inter-site distance, k_B is the Boltzmann constant and T is the temperature of the system. For site i, ϵ_i is given by

$$\epsilon_i = \epsilon_{0,i} + qV_{z,i} + \frac{q^2}{4\pi\epsilon_0\epsilon}\Phi_i, \tag{3.3}$$

where ϵ_0 is the vacuum permittivity, ϵ is the relative dielectric constant, q is the fundamental charge, $V_{z,i}$ is the local electric potential due to an externally applied electric field F_z in the z direction, and Φ_i is the local p-p interaction term. The sequence of hopping events in the lattice can be translated into a hole mobility μ_p, according to

$$\mu_p = \nu_0\sigma\frac{n_+ - n_-}{F_z a \cdot N \cdot \delta t}. \tag{3.4}$$

Here, n_+ and n_- are the number of charge hops in respectively the positive and negative z direction, during the simulation time interval δt.

The behaviour of μ_p can be described in terms of the dimensionless degree of disorder $\hat{\sigma} = \frac{\sigma}{k_B T}$, and the dimensionless electric field $\hat{F} = \frac{qF_z a}{\sigma}$. Increasing disorder results in larger energy differences between sites, reducing μ_p. Large electric fields decrease the energy difference between adjacent sites, thus increasing μ_p.

The simulations are performed on a cubic lattice with dimensions of $512a \times 512a \times 128a$ and full periodic boundary condition. The segment size is set to $16 \times 16 \times 16$ lattice sites. We use a maximum particle density of one particle per ten lattice sites, this is much larger than the typical densities that are found in space charge limited diodes (between one in 100 and 1,000,000). We are mainly interested in the behaviour when including pp-interaction at large particle concentrations, because these are hard to calculate in a serial version of the algorithm. As we will discuss later on, the current configuration is not optimized for calculating situations with low particles density.

The hopdistance a_{ij} is fixed to a and α_{ij} is set to $\frac{10}{a}$ [18]. Energy levels are randomly assigned to all sites, and holes are placed on the sites with the lowest energy levels. For the Coulomb interactions, we employ a cutoff radius of $16a$, as this is adequate for particle densities down to 0.01 % [78]. Finally, a relative dielectric constant of 4.0 is used, which is a common value for organic semiconductors. All calculations were performed on a NVidia k20c board.

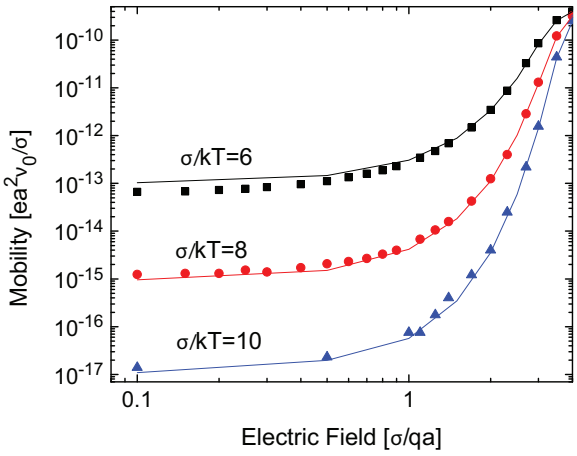

Figure 3.6: Comparison of the field dependent mobility between our method (symbols) and numerically solving the corresponding master equation (lines), for different values of σ and a charge carrier concentration of 10^{-3}. For increasing disorder and decreasing field, the charge transport characteristics deteriorate fast because of increasing energy differences between hop sites. A good agreement is found between the two different methods. However, a deviation will arise for increasing charge carrier concentrations, because site correlations are not taken into account properly in the numerical solution of the master equation.

First, the results of the simulation without p-p interactions are compared to numerically solving the corresponding master equation [95]. This numerical solution requires the same parameters, but uses a simulation volume of only $128a \times 128a \times 128a$. A charge density of $10^{-3}a^{-3}$ is used to prevent artifacts due to site correlations [96]. Figure 3.6 shows μ_p versus F_z for different values of $\hat{\sigma}$. The agreement between both methods validates our approach. Moreover, the difference in simulation volume indicates that finite size effects are not important for the selected density [136].

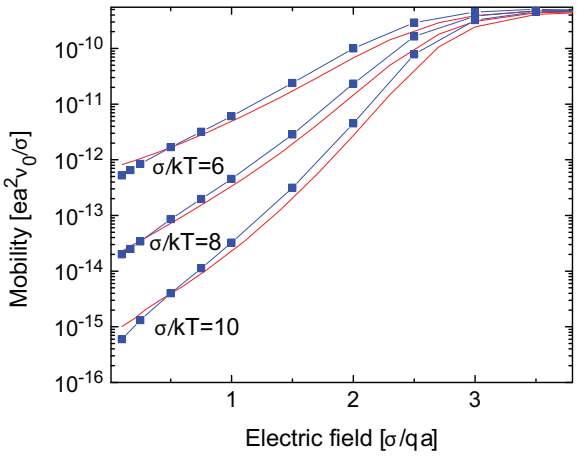

Figure 3.7: Comparison of field dependent mobility with (squares) and without (no symbols) Coulomb interactions, for a charge carrier concentration of 10^{-3}. A small deviation is observed between the two methods, that matches the difference that was found in [78].

Next, we consider the effect of p-p interactions on the field dependence of μ_p. The range of parameters comprises of typical values that are hard to address using serial KMC: low F_z, high $\hat{\sigma}$ and high densities. Figure 3.7 contains the field dependence of the mobility for different values of $\hat{\sigma}$, with (blue lines) and without (red lines) taking into account Coulomb interactions. The results for μ_p including Coulomb interactions deviate slightly from the values without coulomb interactions: μ_p reduces for low F_z and increases for high F_z. This matches the calculations of van der Holst [78] and is contributed to the formation of an unidirectional potential barrier around each charge. At low fields, this barrier reduces the hopping rates to all adjacent sites, which leads to a slight decrease in mobility. At high fields, more and more charge hops are down

in energy, causing the mobility to approach the values without coulomb interactions.

Although the effects of coulombic interactions on the mobility are small, including the interactions is crucial for a good understanding of organic light emitting diodes and solar cells. In general, the materials in these devices possess a low relative dielectric constant (between 3.0 and 4.0). Therefore, electrons and holes have a large attractive force on each other, and a proper description of generation and recombination processes requires the inclusion of these interactions [137].

3.4 Performance

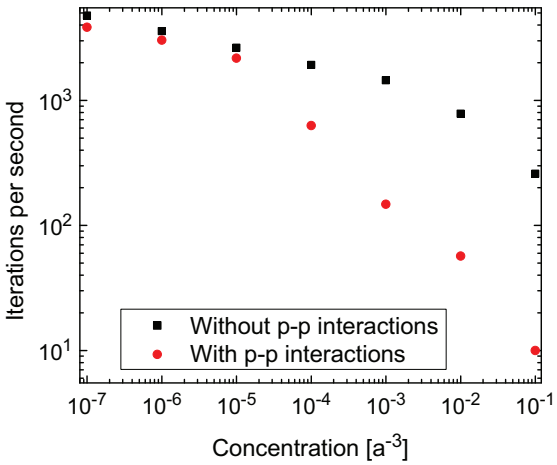

Figure 3.8: Number of iterations that are executed per second. The circles correspond to the case including p-p interactions, whereas the square symbols represent results without p-p interactions. For low concentrations, the simulation is limited by the relatively low serial performance of the GPU. Although 3,000 iterations are performed each second, only a few charges will hop during every iteration. The kink that can be seen at 10^{-5} (including p-p interactions) and at 10^{-3} (excluding p-p interactions) correspond to the regime where the simulation is not limited by the GPU overhead anymore.

We benchmark the simulation by determining the number of KMC iterations per second as a function of the average charge concentration in the device. The same simulation configuration with segments of $16 \times 16 \times 16$ nodes is used as in the

validation part. Figure 3.8 shows the number of iterations per second for different particle densities, both including and excluding p-p interactions.

At low densities, the number of iterations per second is limited by the overhead that occurs because the algorithm is run on a GPU. This effect is even worse for the case including p-p interactions, because additional GPU functions have to be executed. Insufficient calculations can be done in parallel to compensate for the relatively low serial speed of the GPU. Under these conditions, a CPU algorithm will outperform our GPU algorithm. This changes as the charge concentration is increased. The number of iterations per seconds drops, indicating that more time is spent on performing actual calculations. As more calculations are performed in parallel, the GPU is used more efficiently.

For the case without p-p interactions, the turning point occurs around a concentration of $10^{-3}a^{-3}$. At this point, calculating transition rates and selecting transitions becomes the bottleneck of the simulation. The efficiency at which these calculation are now executed is higher than when they are performed serially. For increasing concentrations, this effect becomes stronger, and a GPU is preferred over a CPU.

For the case including p-p interactions, the turning point will occur much earlier, at a concentration of $10^{-5}a^{-3}$. The bottleneck is caused by the calculation of p-p interactions. Given a fast parallel method for determining the interaction, this means that performing KMC simulations including long range interactions in parallel is more beneficial compared to KMC simulations without these interactions. Moreover, a fast implementation of the p-p interactions is crucial, and the implementation of very fast routines for the transition rates becomes less important.

We also looked at the parallel efficiency that is obtained for different charge concentrations. Figure 3.9 contains the parallel efficiency versus N for $\hat{\sigma} = 0.01$ and $F_z = 0.1$. This figure shows the parallel efficiency for the case without p-p interactions. The impact is that p-p interactions should be limited, because it will only modify the interaction potentials slightly. Therefore, we only show the parallel efficiency for the case without p-p interactions. This efficiency is defined as the average fraction of segments where transitions are carried out during each KMC iteration. Because this number is related to the probability that a null-transition is executed, it will depend on the deviation of the overall transition rates of all segments. Since the transition rates of charges should be independent of the segment, the deviation in overall rate is expected to decrease for increasing concentrations. This explains why the parallel efficiency increases with particle density.

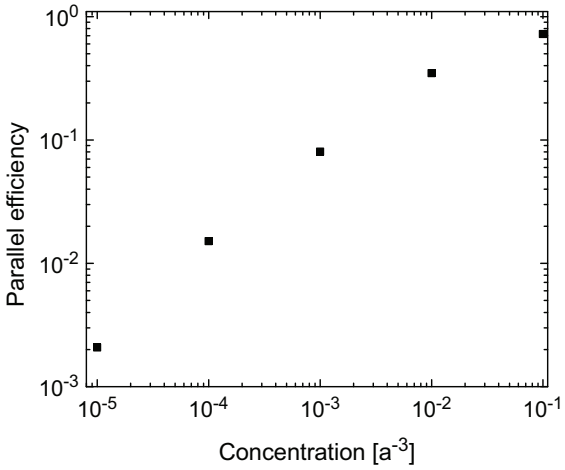

Figure 3.9: The parallel efficiency of the simulation for different particle concentrations. The deviation from the average of the overall transition rate of each segment decreases with increases particle concentration. Therefore, the chance of selecting the null-transition reduces, and the parallel efficiency increases.

Initially, the parallel efficiency is found to increase linearly, this occurs because not all segments contain charge yet. At concentrations above $2 \cdot 10^{-4}$, all segments contain at least one charge on average, and the scaling becomes sub-linear. The maximum efficiency that we reach occurs at a concentration of 10^{-1}, where 7 out of 10 segments perform transitions simultaneous.

Parallel efficiency is also influenced by σ and F_z. Increasing σ will cause more disorder in the overall transition rates of segments, which results in relatively larger maximum. This increases the chance of selecting null-transitions in other segments. For large F_z, the transition rates in the field direction and against the field direction will become more similar. This will cause smaller deviations in the overall transition rates of the segments, resulting in higher parallel efficiencies.

Because the parallel efficiency increases with charge density, and the serial efficiency decreases, an optimum exists where the simulation performs best. The overall efficiency is defined as the product between the parallel efficiency and the number of sequences, and represents the total number of charge moves per second. Figure 3.10 contains the normalized overall efficiency of the simulation, both for the cases includ-

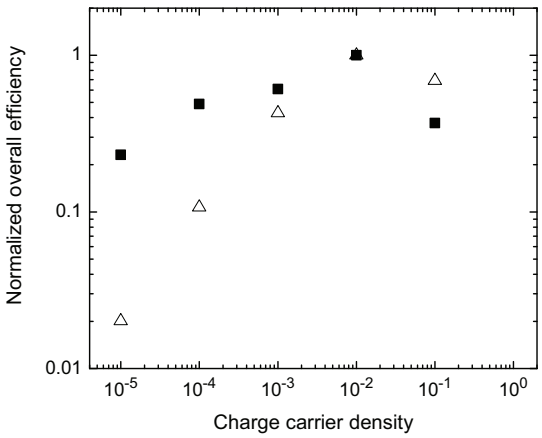

Figure 3.10: The normalized overall efficiency of the parallel KMC simulation. The triangles shows results for the case excluding p-p interactions and the squares represent the case including p-p interactions. The simulation is optimized for particle concentrations around $10^{-2}a^{-3}$. Increasing the segment size will shift this point to the left, because the parallel efficiency will increase for lower concentrations.

ing and excluding p-p interactions. The case including p-p interactions uses the same parallel efficiencies as the case without p-p interactions.

For the current configuration with segments of $16 \times 16 \times 16$ nodes, the efficiency shows a maximum at carrier densities of $10^{-2}a^{-3}$, regardless of the presence of Coulomb interactions. This indicates that the current configuration is optimized for relatively high particle concentrations. For applications involving lower particle concentrations, the simulation can be improved by increasing the segment size.

The procedures for calculating transition rates and p-p interactions operate on groups of 32 particles simultaneously. When the occupation level of segments becomes too low, these calculations are not performed in parallel anymore. Given that the cutoff radius remains equal, larger segments prevent this issue, because the number of particles per segment increases. Also, larger segments reduce the number of segments that is required to span the simulation volume: this lowers execution times. Finally, using more particles per segment will also increase the parallel efficiency, because the overall transition rates of the segments will increase.

Our method can be further improved by dynamically segmenting the simulation

volume, such that an optimal occupancy of all segments is ensured. In an optimal segmentation, the overall transition rates of all segments are equal, leading to a parallel efficiencies close to 100 %. Alternatively, segments can be setup to contain equal amounts of particles. This can also be beneficial for implementing approximative schemes like FMM that calculate the interaction potentials. These improvements also allow efficient simulation of volumes with non-homogeneous particle distributions, for instance in organic solar cells or light emitting diodes.

3.5 Conclusion

We have developed a parallel, time synchronous, lattice based Kinetic Monte Carlo simulation that includes particle-particle interactions and runs on a GPGPU board. Determining transition rates and selecting transitions for execution was done using fast GPU methods, where an eight color checker board prevents boundary conflicts. Boundary transitions were propagated after each iteration, providing a good description of particle-particle interactions. The modifications of the interaction potential due to nearby particle moves were taken into account by adding dipole correction terms, thereby reducing the required number of operations. Exact evaluation of the interactions obtained by representing all interaction terms as 32-bit floating numbers, where only the integer range between -2^{22} and 2^{22} was used. We have validated our method by modeling the charge transport in disordered organic semiconductors including coulomb interactions between charges. The results were in good agreement with values obtained from numerically solving an approximation of the corresponding master equation. Performance is mainly governed by the density of particles in the simulation volume. For low densities, the limited amount of parallel work was unable to gain from the massively parallel architecture. Performance improved for large densities, both due to better use of the architecture and to increased parallel efficiency. The method allows calculations on large volumes, which is crucial in the field of organic photovoltaics. Moreover, the fast evaluation of particle-particle interactions allows simulations with high particle concentration. Further improvement may be obtained by using a dynamic segmentation algorithm that optimized the parallel efficiency, and accelerates the calculation of transition rates and p-p interactions. But we leave this addition for future work.