Beyond-Metropolis Markov chains: From the foundations to applications

Parallel implementation of event-chain Monte Carlo algorithm in long-range systems

Botao Li[1,2]
Supervisor: Werner Krauth[2]

[1]Ecole Polytechnique, Palaiseau, France
[2]Laboratoire de Physique de l'Ecole normale supérieure ENS, Université PSL, CNRS, Sorbonne Université, Université Paris-Diderot, Sorbonne Paris Cité, Paris, France

July 7, 2019

Abstract:

In recent years, the event-chain Monte Carlo algorithm (ECMC), an event-driven Monte Carlo algorithm which implements the idea of irreversible Markov chain, has been proposed. Though ECMC shows promising performance when implemented in hard-sphere systems, long-range systems and spin systems, parallel implementation of ECMC has never been successful. In this project, we show theoretically that ECMC with multiple active particles satisfies the global-balance condition. And we propose dynamic domain decomposition, a scheme which makes parallelizing ECMC possible. According to our experiment and estimation, dynamic domain decomposition is able to speed up the simulation in hard-sphere systems by orders of magnitude.

1 Introduction

Markov-chain Monte Carlo algorithms[1] are widely used when studying equilibrium-statistical-physics systems and when sampling the probability distribution of the system is needed. The vast majority of Markov-chain Monte Carlo algorithms (for example, the Metropolis Hastings algorithm) rely on the detailed-balance condition, which means the probability flow between each of the states should vanish and the dynamics of Markov-chain is reversible. The dynamics of the Markov-chain is diffusive and the algorithms are usually slow.

In recent years, an algorithm which implements the idea of irreversible Markov-chain was proposed, which we refer to as the event-chain Monte Carlo (ECMC)[2]. The ECMC algorithms satisfies the global-balance condition instead of the detailed-balance condition, so exploring the phase space can be much faster. The configurations sampled by ECMC are the steady states for the Markov chain. However, nothing prevent us from choosing a set of physical state in equilibrium to be the steady state of the Markov chain. Thus, it is possible to sample physical equilibrium states as steady states of the Markov chain. And acquiring an independent configuration requires less steps compared to algorithms which satisfy the detailed-balance condition. Because the Markov chain is irreversible so that the dynamics is no longer diffusive. ECMC algorithms are event-driven, which means instead of updating the configuration of the system at each time step, the state of the system is only updated when there is an event (for example, the collision of two particles) taking place. When doing the same amount of jobs regarding the simulation, an event-driven algorithm requires less computation power then a time-driven algorithm (for example, the Metropolis Hastings algorithm).

ECMC algorithms have been applied in hard-sphere systems, systems with long-range interaction and spin models (or lattice field theories)[3][4][5][6]. In these systems, ECMC shows better performance than ordinary Monte Carlo algorithms. However, a real-life problem in such system usually features more than 10^6 particles (sites) and traditional algorithms benefit from parallelization. Unfortunately, parallelizing an event driven algorithm has been a big problem, haunting event-driven algorithms for decades. Currently, there is no successful scheme for the parallel implementation of the ECMC algorithms. In this master thesis, we have designed a scheme which can parallelize ECMC and we estimated its performance. The thesis is organized as follows: In section 2 we will introduce in detail the algorithm and systems we work with. In section 3 we prove that it is possible to parallelize an ECMC algorithm. In section 4 we describe the scheme which making parallel implementation of ECMC possible and show the estimation of speed-up of parallelization. In section 5, we make concluding remarks. The scheme we designed is able to speed up ECMC by more than ten times.

2 Background

In this project, we mainly focus on the implementation of ECMC in 1D hard-sphere system with factor fields. The thermodynamics of the system can be solved analytically, but simulating it is non-trivial. Thus, it is a perfect playground for us to validate and benchmark our implementations.

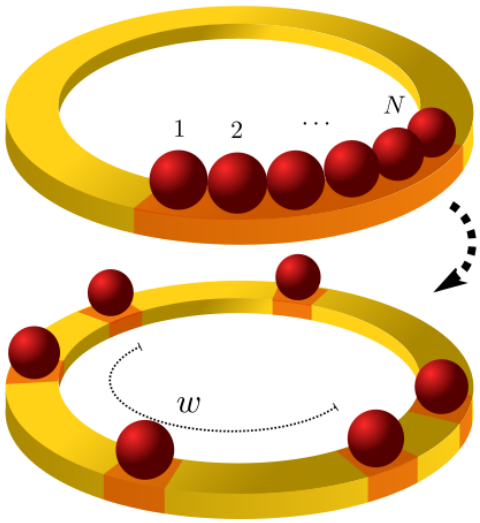

Figure 2: 1D hard-sphere system with periodic boundary condition. The upper configuration is out-of-equilibrium, in which neighbouring particles contact each other. The lower configuration is in equilibrium. ω is the half-system-distance. The process demonstrated by this figure is mixing. Figure cited from Ref[7]

2.1 1D hard-sphere system with Factor field

The system we are interested in is a 1D hard-sphere system with periodic boundary conditions (figure 2). There is a hard-sphere potential for each sphere, which indicates the boundary of the sphere (figure 3). In 1D, changing the size of the spheres is equivalent to changing the scale of the system. For simplicity, we will always set the radius of the spheres to 0. Notice that, due to the presence of hard-sphere potential, even if the radius of the spheres are 0, the spheres cannot go through each other. In addition to hard-sphere potentials, there are factor fields[7] in the system. Factor fields are fictitious long-range interactions between two spheres. They are constant forces and their potentials are linear in respect of distance between two particles (figure 3). The factor field was recently shown[7] reduce the auto-correlation time of the system, which is the number of steps before an equilibrium becoming another independent equilibrium configuration. It was shown that, with the help of the factor field, the auto-correlation time is reduced from $\mathcal{O}(N)$ to $\mathcal{O}(N^{1/2})$ (measured in sweeps), where N is the number of particles in the system.

When the sphere radii are 0, this is trivial to see that in equilibrium, the distribution of the position of each sphere is uniform. Knowing the distribution of the configurations in equilibrium allows us to give our simulation equilibrium initial states directly.

In order to know the properties of the system, we need to calculate the observable of the system. We are mainly interested in two quantities, the half-system distance and the variance of the half-system distance.

The half-system distance is defined as:

$$\omega = x_{i+N/2} - x_i - N\sigma/2 \tag{1}$$

where N is the number of spheres in the system and σ is the diameter of the spheres. The distribution of ω, is a beta distribution, in which the parameters are $N/2$. Knowing the distribution of ω allows us to compare the sampled distribution to the analytically

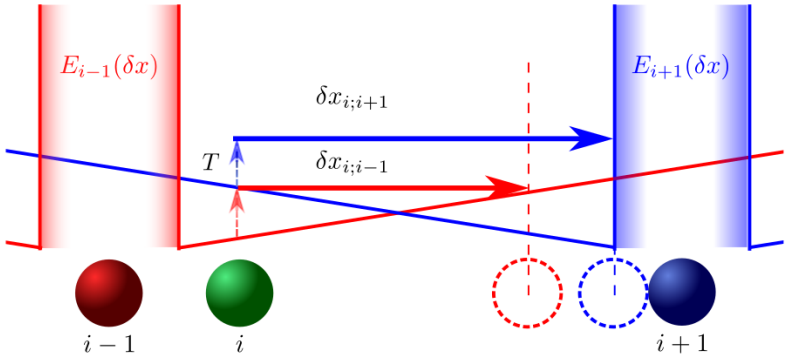

Figure 3: Hard-spheres with factor fields. i is the active particle. $\delta x_{i,i-1}$ and $\delta x_{i,i+1}$ are the proposed events. The presence of hard sphere potential leads to the blue event. Since when the active particle hits another particle, $\Delta U = \inf$. The red event is related to the factor field. Moving the active particle forward will increase the potential between $i-1$ and i, thus there is a chance that the proposed move will be rejected. It is possible to sample directly the total $\Delta U/T$ before the rejection.[5] This potential will be translated into the distance the active particle can move before being rejected, which is $\delta x_{i,i-1}$. Figure cited from Ref[7]

calculated distribution, so that it is possible to detect whether there are bugs in the simulation.

The expression of the half-system distance depends on the position of a specific sphere, thus there could be more than one half-system distance in a configuration and there is a distribution of half-system distances[1]. In this sense, the variance of half-system distance $var(\omega)$ can be defined. If all the spheres are placed at the same point, which corresponds to a out-of-equilibrium state, $var(\omega) \propto N^2$. For a system in equilibrium, $var(\omega) \propto N$. Thus, $var(\omega)$ can be used to determine whether the system is thermalized. This observable can also be used to measure the mixing time of the system. Mixing time is the time it takes to reach a first equilibrium configuration from an arbitrary non-equilibrium state. Like the auto-correlation time, it is also a quantity to measure the speed of a simulation algorithm.

2.2 Factorized event-chain Monte Carlo

In an ordinary Monte Carlo algorithm[1], at each time step, a randomly chosen particle will move a random step. The difference between the energy of the old and new configuration ΔU will be evaluated. The probability of accepting the proposed move is $\min(1, \exp(-\beta\Delta U))$, where $\beta = 1/T$ and T is the temperature of the system. This indicates that if the proposed move lowers the energy of the system, the new configuration will be accepted. If the proposed move raises the energy of the system, this move has a chance of being rejected. After the proposed move being rejected or accepted, another move will be proposed based on the current configuration. It can be shown that this algorithm satisfies the detailed-balance condition.

The factorized event-chain Monte Carlo[8][5] algorithm, which we are going to implement, is different from a ordinary Monte Carlo algorithm in the following aspects:

[1]This distribution is for a single configuration and is not the distribution which related to the ensemble.

- In factorized ECMC, the potential of the whole system is broken into factors. A factor can be expressed as $M = (I_M, T_M) \in \mathcal{P}(\{1, 2, ..., N\}) \times \mathcal{T}$, where $\mathcal{P}(\{1, 2, ..., N\})$ is the power set of the indices and $\mathcal{T}$ is the set of interaction types. I_M is the set which contains the index of all the particles in a factor. The total potential U of the system is $\sum_M U_M$, where U_M means the potential of the corresponding factor. The Boltzmann weight of the system becomes $\pi(c) = \prod_M \exp(-\beta U_m(c_M))$, where c denotes the configuration of the whole system and c_M denotes the configuration of particles in factor M. In ECMC, instead of using a filter for the whole system to determine whether a proposed move is rejected, $\prod_M \min(1, \exp(\beta \Delta U_M))$ becomes the criterion of whether a configuration is accepted. Since the Boltzmann weight can be decomposed in the same way as the filter, the new criterion still satisfies the detailed-balance condition.

- In ECMC, the moving particle always moves in one fixed direction. The moving particle is referred as active particle. Currently in ECMC algorithm, there is only one active particle at a time. In our convention, the active particle always move to the right, or $\hat{x}$ in higher-dimensional systems. When a proposed move is rejected by another particle, the former moving particle will stop moving and the particle which rejects the move will start to move in the same direction. This is referred to as lifting and it substitutes rejection in the ordinary Monte Carlo algorithms. Though the active particle is not a single particle, the trajectory of active particle could still be defined. The active particle could also be depicted as a label. When a particle possesses this label, it moves. The trajectory of particles when they possess the active label constitute the trajectory of an active particle. In order to describe lifting, we extend the phase space. Let c be a configuration of the system. In ECMC, a configuration will be labeled as (c, a), where a is the index of the moving particle.[2] Boltzmann weight of two configuration which only differ by the index of active particle should be identical. Thus, the weight of all the configurations is modified by one constant, which can ignored when we compare the Boltzmann weight of one configuration to another.

- The ECMC algorithm is event-driven, which means the configuration of the system is updated only when there is a lifting event[3]. ECMC can be perceived as a continuous-time Monte Carlo algorithm, which means the size of each time step is infinitesimal.[9] At each time step, a binary random variable is sampled to determine whether the proposed configuration is rejected. However, these binary random variables, as well as the infinitesimal time steps, are conceptual and will not appear explicitly when we implement ECMC.[4] It can be shown that the distance an active particle can move before being rejected can be sampled directly. Thus, practically, the active particle does not propose configurations. Instead, it propose events. There could be more than one event at a time. The time (in terms of Markov dynamics) before each event taking place can be calculated. Only the first event is real, the other event could change after the first event. In one-dimensional hard-sphere system with factor fields, there are only two proposed events at a time.

[2]This is for one moving particle. If there are more than one moving particle, a will be substitute by A, which is the set of all the label of moving particles.

[3]Unless we do time slicing or sampling.

[4]From now on, the phrase 'rejection' only appears infinitesimal-time-step description.

One of them is a collision with the sphere which is to the right of the active particle, the other is lifting the sphere to the left of it through the factor field.

Putting these together, when simulating a system in arbitrary dimension and assuming there are interactions between the particles, a factorized ECMC algorithm operates as follows:

- Find all the factors which contains the current moving particle
- Sample random numbers to find out how long can the active particle move before the lifting event taking place in each factor
- Find the event which takes place first, do the lifting
- Repeat

In this project, when simulating a 1D hard-sphere system, the strength of factor field is tuned so that the event rate of lifting the particle to the right of the active particle and to the left of the active particle are identical. Since we are free to choose a time scale for the simulation, without losing generality, we can always set the velocity of the active particle to be 1. Then, in one time, the expected number of events are

$$2(N - N_{\text{active}}) \tag{2}$$

where N_{active} is the number of active particles in the system[5]. Algorithm 1 is the pseudo code of implementing of ECMC in a 1D hard sphere system with factor field.

```
initialization;
active particle id = 0;
while not having enough data points do
    distance = position[active particle id + 1] - position[active particle id];
    epsilon = exp(factor field strength / temperature);
    if distance < epsilon then
        position[active particle id] += distance;
        active particle id -= 1;
    else
        position[epsilon] += distance;
        active particle id += 1;
    end
end
```

Algorithm 1: A example of ECMC in one-dimensional hard sphere system with factor field. Periodic boundary condition is not explicitly written here. The particles are represented by an array which contains the positions of the particles. The active particle is assumed to travel at unitary speed. $\exp(l)$ means sample a exponentially distributed random number with parameter l.

[5]Now, there is only one active particle in the system, but we have shown that there could be more in the next section.

2.3 Proof of stability of ECMC

The goal of this section is not to give a rigorous proof of the stability of this algorithm, but to provide a general idea of how to prove it and introduce related quantities and notions. A full proof can be found in [5]. Here we will use the infinitesimal-time-step description of ECMC instead of the event-driven one.

The global-balance condition can be expressed as:

$$\pi(x) = \sum_{x'} \mathcal{F}(x' \to x) = \sum_{x'} \pi(x') p(x' \to x) \tag{3}$$

where x denotes configurations, $\mathcal{F}$ denotes probability flow and $p(x' \to x)$ denotes the probability of going from configuration x' and x. Let us assume that the system we are interested in is arbitrary dimensional. And we allow long-range interactions between the particles and the potential of each particle has arbitrary form.

Consider a simple case in which there are only two particles in the system and one of them is moving. There are two probability flows which goes into this configuration. One of them is mass flow, which corresponds to the moving of the active particle, while the other is lifting flow, whose probability is related to the rejection of the movement of the other particle. (figure 4) It is $\mathcal{F}^{\text{mass}} = \pi(c'')p(c'' \to c)$, in which $p(c'' \to c)$ is the accept probability of configuration c when the system is in configuration c''. The mass flow is still constrained by the detailed-balance condition, which means $\mathcal{F}^{\text{mass}} = \pi(c'')p(c'' \to c) = \pi(c)p(c \to c'')$. The lifting flow is $\mathcal{F}^{\text{lift}} = \pi(c)(1 - p(c \to c'))$, where c' is the configuration in which sphere 2 goes one step further. Due to the translational invariance of this factor, the accept probability of moving 2 forward is identical to the probability of moving 1 backward. Thus, we have

$$\mathcal{F}^{\text{lift}} = \pi(c)(1 - p(c \to c')) = \pi(c)(1 - p(c \to c''))$$

Thus

$$\mathcal{F}^{\text{mass}} + \mathcal{F}^{\text{lift}} = \pi(c)$$

and the global-balance condition is satisfied. Now, consider general cases, in which there are more than two particles in the factor. Due to the translational invariance of the factor, we have

$$\sum_{k \in I_M} \partial_{x_k} U_M = 0$$

where I_M is the index of particles in the factor and U_M denote the potential of the factor. Thus, the particles in a factor can be divided into two groups. $k \in I_M^-$ if $\partial_{x_k} U_M \leq 0$ and $k \in I_M^+$ if $\partial_{x_k} U_M > 0$. And we have

$$\sum_{k^+ \in I_M^+} \partial_{x_k^+} U_M = - \sum_{k^- \in I_M^-} \partial_{x_k^-} U_M \tag{4}$$

For a configuration c in which the active particle is in I_M^+, there is no lifting flow going into this configuration. Because by using the argument for two particle factor, it can be shown that the mass flow going into this configuration is $\pi(c)$. Thus the global-balance condition is already satisfied by mass flow. This is due to the fact that moving back particles in I_M^+ can only decrease the potential of the factor.

For a configuration c in which the active particle is in I_M^+, the mass flow is

$$\mathcal{F}^{mass} = \pi(c)\left(1 - \beta\partial_{x_{k^-}} U_M dx\right)$$

where k^- is the index of the active particle in c and dx is the infinitesimal step that the active particles moves. It can be shown that the mass flow of c is $\mathcal{F}^{mass} = \pi(c)\left(1 - \beta\partial_{x_{a^-}} U_M dx\right)$ where a^- is the index of the active particle. The difference between mass flow and $\pi(c)$ should be compensated by lifting flow. Moving active particles in I_M^- never increases U_M, thus active particles in I_M^- never leads to lifting. So for a lifting event, the active particle which propose the event is always in I_M^+ and the target is always in I_M^-. The lifting flow is $\mathcal{F}^{lift} = \pi(c)\beta \sum_{k^+ \in I_M^+} \partial_{x_{k^+}} U_M \gamma_{k^+ \to a^-} dx$. In order to satisfy the global-balance condition, one has to find a set of γs which satisfy

$$\sum_{k^- \in I_M^-} \gamma_{k^+ \to k^-} = 1$$

and

$$\partial_{x_{a^-}} U_M = \sum_{k^+ \in I_M^+} \partial_{x_{k^+}} U_M \gamma_{k^+ \to a^-}$$

The first condition comes from the fact that all the lifting flow comes from the rejection of proposed move of active particles in I_M^+. The values of γs affects how we handle lifting when we implement the algorithm. Thus finding values of γs means finding a lifting scheme for ECMC. It is shown that, if

$$\gamma_{k^+ \to k^-} = \frac{|\partial_{x_{k^-}} U_M|}{|\sum_{k^- \in I_M^-} \partial_{x_{k^-}} U_M|}$$

the conditions are satisfied. Then, we have found a lifting scheme which satisfies global-balance condition.

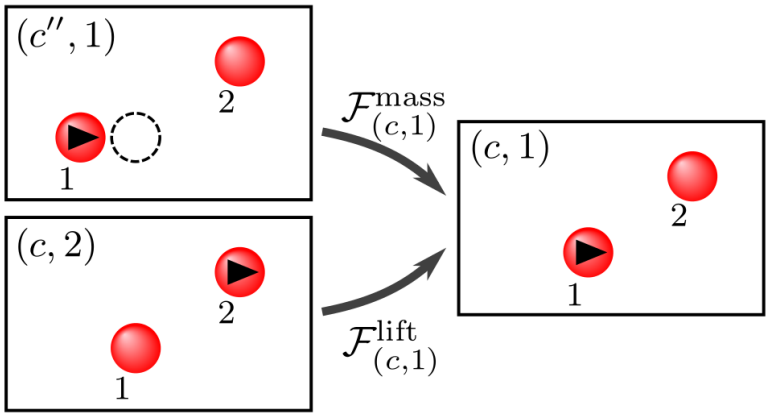

Figure 4: Mass flow and lifting flow corresponding to a configuration. Figure cited from Ref[5]

3 Multiple active particle ECMC

The ECMC algorithm could be parallel in two senses. One of them consists in using multiple processors to solve the problems encountered in the serial ECMC algorithm. For example, we could use parallel algorithms to do the sorting, arithmetic, etc.. However, such an implementation does not scale with the system size(at least for short-range systems), since the number of proposed event does not scale with respect to the size of the system. Thus, at some point, adding processors does not give us any speed-up even for large systems. Not to mention the hardware limitations which may forbid us from doing such parallelization. What we are interested in is have several moving particles, and the event proposed by these moving particles will be processed by different processors. Naively, one can expect that the speed-up is of the order of the number of processors.[6] However, more moving particles means different Markov dynamics compared with the serial ECMC. Thus, the first step toward parallel ECMC is proving that, with more than one active particle, the ECMC algorithm converges toward the steady state. In other words, we need to prove that the ECMC with multiple active particles satisfies the global-balance condition. Notice that there could be more than one algorithm that satisfies the global-balance condition. We will find one of them and prove it works.

3.1 Lifting scheme

Let us first assume that there are only two-particle factors in the system. If only one of the particles in the factor is active, the arguments in the previous section also applies. If both of the particles are active, then there is no lifting and thus the potential of their factor has to remain the same or decrease ($\Delta U_M \leq 0$), or else rejections will be introduced. Since the potential we are considering has arbitrary forms. Thus, in order to keep $\Delta U_M \leq 0$, the active particles should always move in the same direction at identical speed. Thus, it is still possible to use dx to denote the movement of all the active particles. Besides, since ECMC has infinitesimal time steps, two lifting event cannot take place in the same time step.

Let us assume a general case in which there are multiple active particles in a factor which contains several of them. Again, due to the same reason as when there is only one active particle, the active particle of lifting in always in I_M^+ and the target is always in I_M^-. For a configuration (c_M, A), the mass flow which goes into this configuration is (due to detailed-balance condition)

$$\mathcal{F}^{\text{mass}}_{(c_M,A)} = \pi(c_M)\left(1 + \beta \sum_{k^- \in I_M^- \cap A} \partial_{x_{k^-}} U_M dx\right)$$

And the lifting flow is

$$\mathcal{F}^{\text{lift}}_{(c_M,A)} = \pi(c_M)\beta \sum_{k^- \in A \cap I_M^-} \sum_{A'} \sum_{k^+ \in A' \cap I_M^+} \partial_{x_{k^+}} U_M dx \gamma_{k^+ \to k^-}$$

where $\gamma_{k^+ \to k^-}$ it the probability of lifting from k^+ to k^- which is yet to decide, and A' is the set of active particles before lifting. The lifting flow contains all the possible lifting

[6] We assume the system is large enough so that the number of moving particles are always a small fraction of the system.

events. The active particle of lifting can be all the particles which is not active and in I_M^+, while the target can be each of the active particles in I_M^-. And each of these event is associated with a probability which ensures the global-balance condition. If there are valid value for $\gamma_{k^+\to k^-}$, we can conclude that in this factor the global-balance condition is satisfied. Notice that, when an event take place, there is only one active particle which becomes inactive, despite the fact that the lifting event is proposed by all the active particles in I_M^+. Thus, the lifting could be divided into two steps. The first step is to find out the particle which will become inactive, and the second step is to find a target. Thus $\gamma_{k^+\to k^-}$ can be expressed as $\gamma_{k^+\to k^-} = p(k^-|k^+)p(k^+)$, where $p(k^+)$ is the probability of finding a rejected particle and $p(k^-|k^+)$ is the probability of finding a target after finding a rejected particle. Normalization gives the first two constraints on these probabilities:

$$\sum_{k^-\in A\cap I_M^-} p(k^-|k^+) = 1 \tag{5}$$

and

$$\sum_{k^+\in A'\cap I_M^+} p(k^+) = 1 \tag{6}$$

The last constraint that these probabilities needs to satisfy is the global-balance condition:

$$\mathcal{F}^{\text{mass}}_{(c,A)} + \mathcal{F}^{\text{lift}}_{(c,A)} = \pi(c, A)$$

The rejection of a configuration is due to the increase of potential of the whole factor, i.e. conceptually it is problematic to say that it is one of the active particles which leads to the rejection. However, the expression of rejection probability of the whole factor is identical to the probability of being rejected when we judge each active particle whether the proposed move is rejected. These two cases is related by setting $p(k^+) \propto |\partial_{x_{k^+}} U_M|$. If we use this scheme, $\sum_{A'}\sum_{k^+\in A'\cap I_M^+} \partial_{x_{k^+}} U_M \gamma_{k^+\to k^-}$ becomes simply $\sum_{k^+\in I_M^+ - I_M^+\cap A} \partial_{x_k^+} U_M p(k^-|k^+)$. Then after a little calculation, the last constraint can be reduced to

$$\sum_{k^-\in I_M^-\cap A} \partial_{x_k^-} U_M + \sum_{k^-\in A\cap I_M^-}\sum_{k^+\in I_M^+ - I_M^+\cap A} \partial_{x_k^+} U_M p(k^-|k^+) = 0 \tag{7}$$

Finding a set of valid $p(k^-|k^+)$ can be done by a set of rules[10]. And the rules are visualized by figure 5. Without losing generality, we assume that there are 5 particles in this factor, $|I_M^+| = 3$ and $|I_M^-| = 2$. In the plot, the length of red rectangle is proportional to $|\partial_{x_A} U_M|$, and the same is true for the other rectangles. $q_1 = |\partial_{x_A} U_M|\gamma_{A\to D}$, $q_2 = |\partial_{x_b} U_M|\gamma_{B\to D}$ and the same for q_3 and q_4. The basic rules will be explained by several examples:

- If A is active, then the lifting flow of $A \to D$ is q_1 and there is no lifting flow to E.
- If B is active, then the lifting flow of $B \to D$ is q_2 and the lifting flow of $B \to E$ is q_3
- If A, B is active, the active particle after lifting could be $\{D, B\}$, $\{A, D\}$, $\{A, E\}$

If the target of the lifting is active, nothing will take place. Notice that figure 5 contains all the possible circumstances in equation (7). For example, assume that the configuration

after lifting is (C, D). Then, the first term in equation (7) will be $\partial_{x_D} U_M$. Taking into account the constraint k^+ and k^- have to satisfy, the second term contains contribution of $A \to D$ and $B \to D$. If we use the lifting scheme demonstrated by figure 5, these contribution will be q_1 and q_2, and match the first term in term of absolute value. Similarly, this scheme could be applied to larger or more complicated factor. Then, we could say that this lifting scheme satisfies the global-balance condition for arbitrary factors.

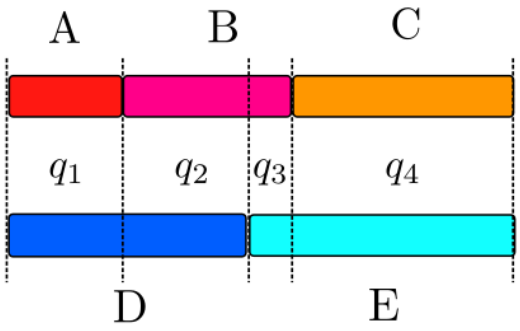

Figure 5: Mass flow and lifting flow corresponding to a configuration. Figure cited from Ref[10]

In general cases, there are multiple active particle in the factors and multiple factors in the system. Since the total acceptance probability is the product of the acceptance probability of each factor, the non-trivial total contribution of lifting flow or mass flow are the sum of corresponding contribution over all the factors when considering a infinitesimal step. Thus, if the global-balance is satisfied factor by factor, it is satisfied for the whole system. And we have proved that ECMC with more than one active particles satisfies global-balance condition.

3.2 Implementation in one-dimensional system

We also check that ECMC with multiple active particles works in practices. We simulate an one-dimensional hard-sphere system with factor field using ECMC with multiple active particles. The algorithm we use is slightly different from ECMC with one active particle. The major difference is that more than two events are scheduled at each step. We use the following strategies to handle this situation:

- Initialization: Find an initial state of the system, either equilibrium or out-of-equilibrium, depending on the problem to be solved.
- Build a list of event candidates: Loop over the active particles, and find out all the event candidates proposed by the active particles. Event is represented by an array, which contains the time before this event taking place, the index of active particle and the index of the target. Put these events in a list.
- Commit event: Find the event which take place first, and evolve the system until this event take place. This includes moving the active particles and update the time of each event in the list of event candidates. The active particle of this event then become inactive and the target of this event become active. Then, the events

whose target is the target of the committed event will be removed from the list of event candidates. And, three events will be added to the list[7]

- the target of the committed event lifts the active particle of the committed event
- if possible, the target of the committed event lifts other particles
- if possible, other active particles (excluding the target of the committed event) lift the active particle of the committed event

Then, the events which takes place first in the list of event candidates will be committed.

There is a little subtlety regarding the random numbers in this algorithm. Currently in this algorithm, when an event is removed from the list of event candidates without being committed, no information of the removed event is preserved. When an event with the same active particle and target is scheduled latter, a new random number will be sampled.

However, sampling a new random number is not necessary if we update the sampled random number properly. Let us assume that an event with time τ_1, active particle a and target t is scheduled, and removed from the list of event candidates because t becomes active particle τ_2 after the event is scheduled. If an event with active particle a and target t is rescheduled, the time of the new event could be $\tau_1 - \tau_2$ instead of a newly sampled random number[8]. To understand this, we need to treat ECMC as an infinitesimal time-step algorithm. For each active particle in each factor , a random number is sampled to decide whether the move of the active particle will be rejected. We could also say that all the random numbers are pre-sampled and all of the random numbers come from a random-number list. If the random-number generator is good enough, we could exchange two random number in the random number list without affecting the statistical result of the simulation. Thus, we could choose several sub list of the random number list and attach them to each of the particles. When a particle in the factor is moving, we use the random numbers attached to the particle to examine whether its movement will be rejected. There is a random number which leads to the rejection of the proposed move of the active particle which lies in the sub-list of the random numbers. The 'distance' between the random number which is being used at current step and the random number which leads to rejection is nothing else than the remaining time before the lifting event taking place. When the other particles in the factor become active and we no longer use the random number to judge whether the move of particles will be rejected, the random-number list will not be used until the next scheduled event. In the event-driven language, this means the targets have become active particles and the event is removed from the list of event candidates. Then, when a new event is scheduled, we will use the old random number list to determine when the rejection event is going to take place. The random number which leads to the rejection still lies in the same place, so the time before the new lifting event taking place is simply the time of the previous event when it is removed from event candidates. Statistically, sampling a new random number or not does

[7]For a more complex system, the events could be more than 3. However, they could be divided into three classes, each of them is represent by a event listed here.

[8]Here we assume we are simulating a 1D hard-sphere system. In more complicated systems, the calculation of the time of new event could be more complicated. However, the calculation requires no new random number.

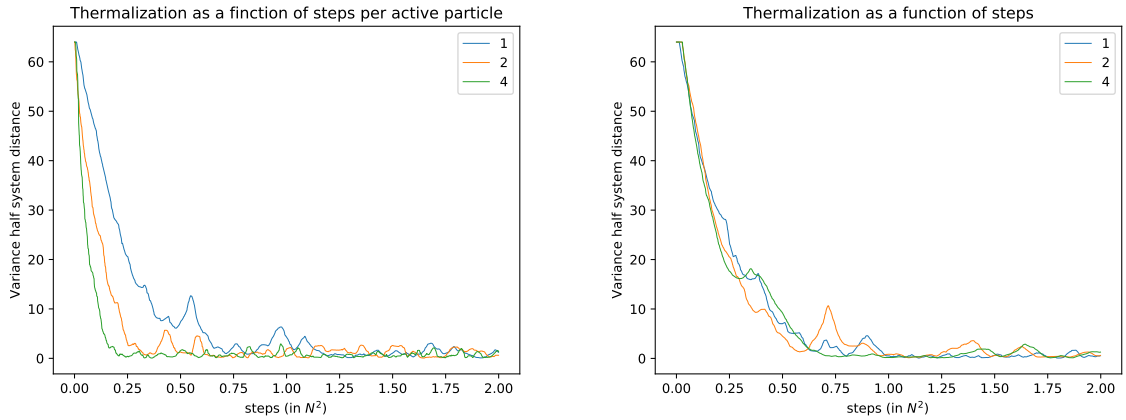

Figure 6: *Left:* $var(\omega)$ as a function of steps per active particle in a system composed of 64 particles. The more active particles there are in the system, the faster the system thermalized. *Right:* $var(\omega)$ as a function of steps for the whole system. The total number of events required for the system to thermalize shows no dependence on the number of active particles.

not change the results of the simulation. However, changing the strategy of managing random number could make a difference when running this algorithm in parallel.

We then test the performance of this algorithm. ω is sampled and the sampled distribution is compared with the analytically calculated distribution. We measure the distance between the sampled cumulative distribution function and the analytically calculated cumulative distribution function[9], and plot it as a function of number of data point we use in the test (figure 7). The error scales as $N_{datapoints}^{1/2}$, which means there is no persisting deviation from the analytically calculated distribution. We also plot $var(\omega)$ as a function of steps for the simulations which has out-of-equilibrium initial states. As shown in figure 6, this algorithm manage to thermalize the system. Then we conclude that we have shown that ECMC with multiple active particle works. The strategy of managing random number makes no difference in terms of performance.

4 Parallel algorithms

We have prove that ECMC satisfies the global-balance condition and we have found a lifting scheme. In order to design a parallel algorithm, we need to find viable parallel schemes. The parallel schemes are how we assign the work load to the processors. In this section, we will introduce two parallel schemes. The basic idea of these schemes are similar to each other. The active particles are divided into groups, in each group there are one or multiple active particles. All the events which are related to active particles in a group will be processed independently. These independent computations are referred to as threads. Creating a thread does not necessarily mean doing calculation on a new processor. However, under some circumstances these two concepts could be equivalent, for example, when there are infinite number of processors and no thread waits for each

[9]In fact, what we do is Kolmogorov Smirnov test. However, real Kolmogorov Smirnov tests use independent data points as input and the observables we sampled are correlated on a time scale. What we are interested in here is the scaling of the statistics given by the KS test.

Figure 7: Scaling of error given by the pseudo Kolmogorov Smirnov test. We run KS test using the sampled half-system-distances, i.e. compare their distribution to the analytically calculated one. The sampled observables are not necessarily independent to each other, so the test we run is not strictly KS test. However, if the statistics given by the test scales with the number of data points, we can still conclude that there is no bug in the simulation (at least for this observable). The blue line in the plot is the result of the test and the orange line is the critical value for the KS test, which is also a reference of the scaling.

other. Assigning the threads to processors is a rather computer science problem and we will not discuss it here. From now on, we will assume that we have a sufficient number of processors at out disposal and will use 'thread' and 'processor' interchangeably.

4.1 Hardware limitations

When designing a parallel algorithm, we need to keep in mind the architecture of the hardware we are going to work with. An illustration of the structure of computer is shown in figure 8. Currently, when doing scientific computations, the performance of computers are mainly limited by the speed of reading(writing) data from(into) memory. The memory of a computer are divided into several levels. The closer the memory to the CPU, the faster the reading and writing are. Notice that if we want the processor to exchange data, we need to access L3 cache or even main memory. According to [11], accessing L3 cache is 10 times slower than accessing L2 cache, and accessing main memory is even worse. Thus, in order to achieve the state-of-the-art performance, we will minimize the data transmission between the threads. Thus, we always assume that the data are not shared among threads unless we state it explicitly.

Usually an affordable computer for scientific computation has $256KB$ of L2 cache per core[12]. For a typical two-dimensional system there are 10^6 particles in the system[2]. Assuming there are only two double precision float number for each particle, the whole system takes up $16MB$ of memory. Thus it is impossible to put the whole system in the cache of one core. Thus, we always assume that we only read a fraction of the system into the caches so that no thread can access the whole system, unless in special cases.

It is possible to design a parallel algorithm without obeying the assumptions. However,

in practice, such algorithm could be even slower than a serial algorithm.

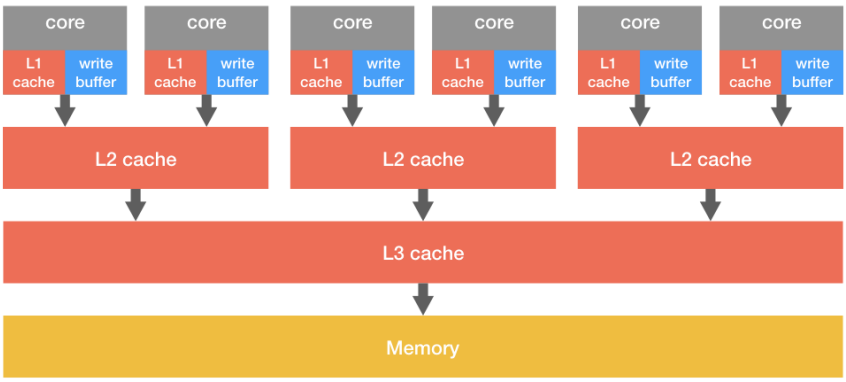

Figure 8: An illustration of the structure of computer. Cores are processors and L1, L2, L3, memory are memories. L1, L2 and L3 will be referred to as cache, while the memory in the figure is usually called main memory or central memory. Figure cited from 'Rust parallelism for non-C/C++ developers'[10]

4.2 light-cone scheme

One undesired feature of the event driven simulation is that there is no upper limit of the speed of the propagation of events. Parallel algorithms require reading a part of the system into the cache associate with the processor. Usually, this fraction of system is the particles in a region in real space. If the events can travel arbitrarily fast, they can always exit the region. In one-dimensional hard sphere system with factor field, a boundary of the events could be found when using pre-sampling. The boundary is time dependent, and the events can never have targets which are out of the boundaries by the time when the event take place. We will refer to this time-dependent boundary as a "light cone".

Firstly, notice that there is an upper limit for the events which involve lifting the particles to the right of the active particles without using pre-sampling. Let us assume that the active particle never lifts the particle to the left of it. The upper limit of the speed of the propagation of events is the traveling speed of the active particle. However, imagine that the active particle always lifts the particle to the left of it (this is possible since the time before a left lifting event could be arbitrarily short). Then, we cannot find an upper limit for such events. However, by doing pre-sampling, we could know the trajectory of the active particle if it always lift the particle to the left of it. The events proposed by this active particle can never cross this trajectory when plotted in a space-time diagram (figure 9). When implementing this scheme, we 'draw' the light cones for each of the active particles, find out the time at which the light cones intersects, and process the events before the first intersection in parallel. Then, we redraw the light cones and repeat. If two active particles are next to each other, we will put them on one thread and draw the light cone for both of them.

This algorithm is vulnerable to one special case. If an active particle lifts the particle to the left of it, then lifts the particle to the right of it in a short time latter, then lifts to the left again, the active particle could exit the light cone, since the time before the new event taking place could be shorter if a new random number is sampled. However,

[10]https://medium.com/nearprotocol/rust-parallelism-for-non-c-c-developers-ec23f48b7e56

using another strategy of managing random number introduced in the previous section, this is no longer a problem.

We have implemented this algorithm on a single processor machine, which simulates a multi-processor machine. We accumulate 10^8 data points. During the simulation, not a single active particle exits its light cone, and the statistical test we run suggests that there is no bug or bias in the simulation. This indicates that the implementation of light-cone scheme is successful. However, the success of light-cone scheme is hard to carry over to the other systems. In a one-dimensional system, there is only one path which connects the active particle and the particle on the light cone. However, this is not the case for $D > 1$. Thus, finding light cones could be computational expensive when $D > 1$. Thus, we will no longer consider this idea.

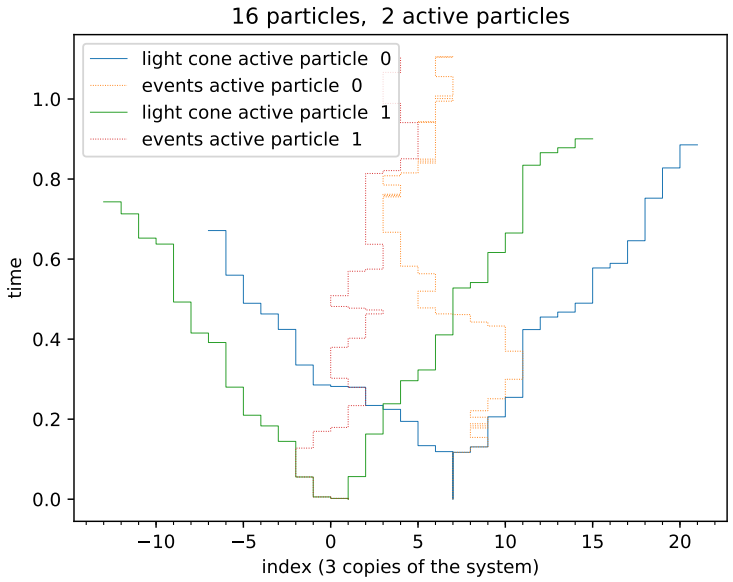

Figure 9: An illustration of light cones and the trajectories of the active particles. There are three replica of the system in the plot, so that we have a clear representation of the trajectories even if we use periodic-boundary condition in the simulation. Notice that the light cones and trajectories we draw are in index space instead of real space. For a 1D hard-sphere system, since no particle could go through each other, there is a monotonic function which maps the index of the particles to their positions. Thus the trajectories in index space and position space should have same topology. The solid lines are the lignt cones and the dashed lines are the real trajectories of the active particles. Before the light cones intersect, the trajectories of active particles can never meet each other.

4.3 Dynamic domain decomposition

Inspired by [13], we propose another parallel scheme for ECMC which we refer to as dynamic domain decomposition. In this scheme, time is divided into time slices, although the simulation remains in continuous time and no errors are introduced. If the simulation starts when the system is at time t, the system will evolve in parallel until $t+\Delta t$, where Δt is the size of the time slices. The simulation waits until all the threads reach $t+\Delta t$, then the system will evolve until $t+2\Delta t$. Though there is no upper bound for the propagation

speed of the events, there is a empirical boundary which contains most of the events. At the beginning of each time slice, a single region, which approximately contains all the events proposed by the active particles in them, are assigned to each of the active particles, as shown in figure 10. There is a chance that two or more regions overlap. Overlap means two active particles are likely to meet each other. If each active particle is assigned to a thread, the position of the same particle could be different on different threads after a time slice, since the data is not shared among threads. And it is impossible to decide which thread has the right configuration. In order to solve this problem, if the regions of two active particles overlap, their regions will be merged. The regions which contain more than one active particles will be referred to as clusters. After merging, each region will be associated with one thread and the events in each region will be processed in parallel. Since these regions are not guaranteed to contain all the events, there will be events which exit the regions. Then, the regions which are related to such events will be marked and the simulation will restart from the beginning of the current time slice. When assigning the regions again, the marked regions will either enlarge, or merge with nearby regions. However, there is a chance that all of the regions merges into one giant cluster and the simulation is serial in practice. In order to study the probability of being serial, we relate the probability of seeing clusters to the birthday problem in probability theory, and show that if the number of active particles is reasonable, there will never be large clusters. Dynamic domain decomposition is designed for system in arbitrary dimensions. When demonstrating this idea, we will use 2D system for simplicity.

4.3.1 Birthday problem

Let there be several regions in a box, what is the probability of overlap? The problem itself is difficult to solve, but solving a similar problem is easy.[11] We cut the box into small pieces whose size is comparable to the regions. Then, we toss points randomly in the box. The probability of seeing several points in the same piece of the box should be approximately the probability of seeing clusters of active particles. Let us assume that there are N particles, N_{active} active particles in the box and there are on average p particles in each region. Usually, the size of the region should depends on how the active particle move and the expression of interaction. Imagine that this parallel scheme is implemented in the simulation of two-dimensional hard disk problem. All the moving particle will move in the same direction. Thus, the trajectory of active particles are similar to parallel lines. In that case, the reasonable region should be a thin rectangle or an ellipse, or triangle if there is no interaction besides hard-sphere potential in the system. If this scheme is implemented in spin models, the trajectory of active sites[12] will be homogeneous random walk. And the region should be round. Here we want to estimate the worst case. Thus we assume no a priori information about the trajectory of the active particle and the regions will be round. The probability of seeing c_i cluster of i active particles is

$$P(\{c_i\}) = \frac{(N/p)!}{(N/p - N_{\text{cluster}})!} \cdot \frac{N_{\text{active}}!}{\prod_{i=1} c_i!(i!)^{c_i}} \cdot \left(\frac{p}{N}\right)^{N_{\text{act}}} \tag{8}$$

[11] When the cluster is really large, there may be large deviation between the answer to these problems. However, as long as we can show the no cluster case dominates, the quantitative results should remain the same.

[12] In spin models, the way of implementing ECMC is to rotate a spin in a constant direction. If the rotation is rejected, the spin which leads to the rejection will continue to rotate. Since the probability of being rejected by each sites are identical, the trajectory of active site is a homogeneous random walk.

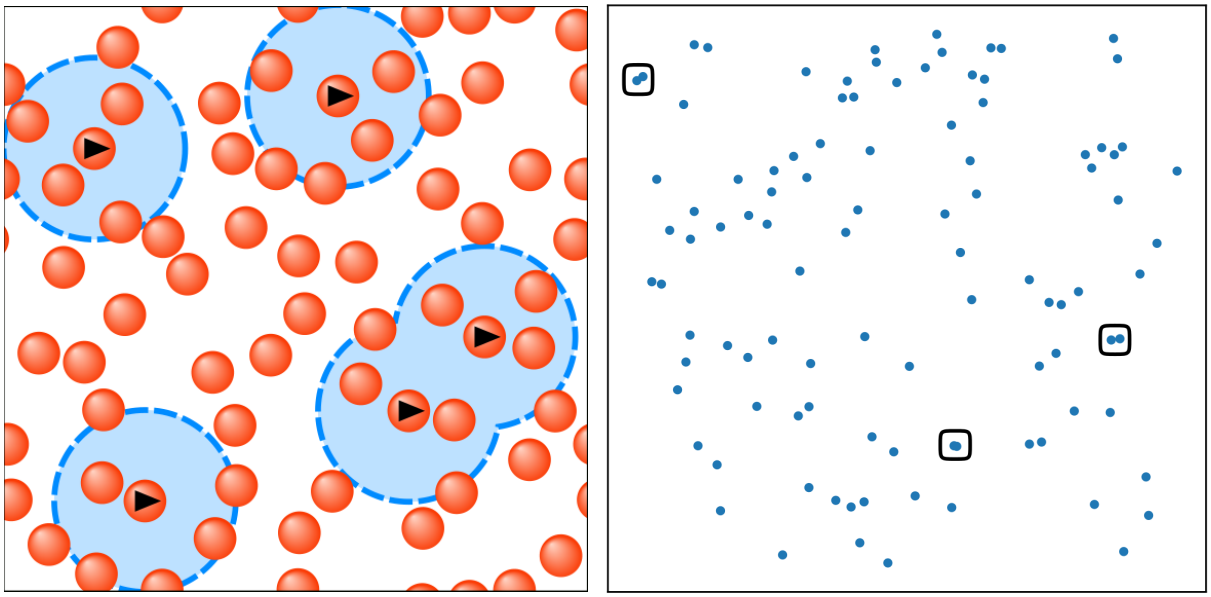

Figure 10: *Left:* An illustration of dynamic domain decomposition. Particles with triangles on them are the active particles. Each active particle is associate with a region. When there is overlap between two or more regions, they will merge into one. Figure cited from Ref[14]. *Right:* The real size of the regions when assuming that there is 10^6 particles, 100 active particles and on average 1000 particles in each region. The configuration is sampled by the simulation of assigning regions in a two-dimensional system. The blue circles are the regions and their sizes are tuned to match the average number of particles in them. Particles are not plotted explicitly in this figure. There are three clusters and they are marked. Notice that, no region lies on the boundary of the system, which indicates that applying periodic boundary condition does not affect the result significantly.

where $N_{\text{cluster}} = N_{\text{active}} - \sum_i c_i(i-1)$ is the number of clusters. In order to study the scaling of probability with respect to the size of the system, we parameterize N_{active} as $N_{\text{active}} = C(N/p)^{\alpha}$. p is the average number of particles in each region and can be treated as a constant which does not depend on N. The probability is plotted as a function of p in fig 11. Several results can be derived using this formula.

- If we put one more active particle in the clusters, the probability would always be suppressed by $\Theta(N^{1-\alpha})$. This indicates that if C is reasonable, the probability of seeing large cluster will be suppressed approximately by the size of the system. Thus, the probability of seeing giant clusters should be vanishingly small.

- Depending on the value of α, there are several values that are constant when changing the size of the system N. For example, if $\alpha = \frac{1}{2}$ and $C = 1$ (and $p = 1$), the probability of have only one two-particle cluster converges when $N \to \infty$. Numerical result shows that this value is roughly 0.4. The probability of having more particles should be suppressed by a factor of the order of $N^{1-\alpha}$ and should be small. The probability of having no cluster should be $N^{1-\alpha}$ times larger. In this case, the approximation breaks and it just means the rest cases are mostly no cluster. Thus, the probability of have a cluster with more than two cookies does not play an important role.

Notice that this formula is independent of the dimension of the system and it is valid for arbitrary dimensions.

4.3.2 Region size

The size of a region could be estimated by modeling the movement of an active particle as a random walk. Let us start by considering a 1D homogeneous random walk. Since we are using an event-driven algorithm, at each step the active particle will move to the particle next to it. The probability of going left and going right are both 0.5. The variance of the position of active particle, in number of particles, is $n/2$ after n events. Assuming that the number of events being processed in each time slice is large enough. The distribution of position of the active particle then become Gaussian. If we want to keep the probability of exiting each region to be smaller than 10^{-3}, the radius of the region should be larger than $3\sqrt{n/2}$ (3σ for a Gaussian). This results is also true for a higher dimensional system. If there are 10^6 particles in a 2D system and we want to process 80 event each time slice, each region should contain roughly 1000 particles. However, the trajectory of active particles are not always homogeneous random walk. If this information is taken into account, there is a chance that more events could be processed in each time slice if p is fixed.

However, these estimation is done in index space and we assign regions in position space. There is a chance that the particle which is next to the active particle is far away from the active particle. Let us consider a 1D system of N hard spheres on a ring of unit length. The PDF of have a particle which is to the left of a reference particle at distance d is

$$p(d) = (N-1)(1-d)^{N-2} \tag{9}$$

which means the probability of having a distance larger than r between two particles is $(1-r)^{N-1}$. On average, the distance between a particle and the particle next to it is $1/N$. The probability of have a distance larger than this average value, when $N \to \infty$,

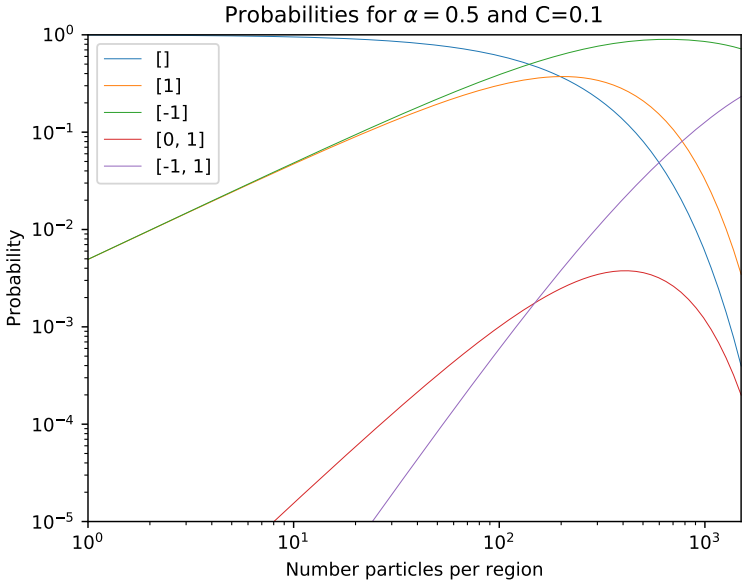

Figure 11: Probabilities of having clusters when there are 10^6 particles and 100 active particles, as a function of p. The list in the legend means the combination of cluster. The position of the number in [ ] means (number of active particles in the cluster -1), and the number means the number of cluster. For example, [ ] means no cluster; [1] means there is one two-particle cluster and [2, 1] means there are two two-particle cluster and a three-particle cluster. '-1' means the contribution of the first 50 terms. $p = 100$ corresponds to $N/p = N_{active}^2$. When $p < 100$, the no-cluster cases dominates. When $p > 100$, clusters appear. When $p = 1000$, the configuration which has two-particle clusters dominates and sometimes there are three-particle clusters. Notice that, when $p < 1500$, we can barely see large clusters.

converges to e^{-1}, which is roughly 0.37. The probability of have a distance larger than a fixed value decreases exponentially when increasing that fixed value. Thus, when there are a lot of particles, the distance between most of them will be small. And when there are a sufficient number of particles in the system and in each region, it is safe to assume that there are plenty of events before the active particle exit its region. This justifies that the trajectory of active particle could be described as a random walk.

There is a trade-off between the probability of existing region and probability of overlap. They are controlled by two parameters: size of time slice and number of particles in the region. Techniques of optimization could be used to find the optimal value of them.

4.4 One-dimensional implementation

4.4.1 Algorithm

In order to verify our idea and measure the performance of dynamic domain decomposition, we implement this scheme in a one-dimensional hard-sphere system with factor fields. At current stage, we use a single processor machine to simulate a multiprocessor machine. For each iteration, the single processor program will loop over all the threads, as if the events on each threads are processed in parallel. The simulation is mainly composed by the following procedures:

- At the beginning of each time slice, the distances between the active particles which are next to each other are calculated. If the distance between two active particles is smaller than the diameter of the region, their regions will be merged into one. Each region is assigned a thread. These threads will be called pseudo-threads. For each region, the position of the leftmost active particle minus region radius and the position of the rightmost active particle plus region radius give the boundaries of the region.

- Simulation starts. Currently, the code will run on a single thread. In each time slice, the simulation on each pseudo-thread will be done one by one. Each pseudo-thread knows the position of all the particles at the beginning of the time slice and the indices of active particles in its region. However, the event which has a target which is out of the region will never be committed, so the simulation is consistent with our assumptions. If there is an event which exits the region, the region in which such event take place will be marked and a flag is on to tell the simulation to repeat. Then the simulation carries on until each of the threads reaches the end of current time slice. All the random number sampled in the simulation will be recorded so that the history of events could be rebuilt when repeating.

- If the flag of repeat is off, the simulation enters the next time slice. If it is on, the simulation returns to the beginning of the current time slice and repeat what has been done. After assigning threads, the regions which were marked will be merged with the regions to the left of each of them respectively. Then the simulation starts. The recorded random number will be used in the repetition.

4.4.2 Performance

We simulated a system with 8 particles and 2 active particles until there are 10^8 data points. Statistical test shows that there is no bug in this simulation.

We then increase the number of particles and measure the speed-up we gain by parallelization. Since there is no parallel implementation yet, the speed-up is estimated from the depth of the algorithm, which is determined by the number of active particles in the largest cluster of active particles. In principle, the time (in real life) it takes to process one event should be almost identical. And, for a fixed period of time (in the system), the number of event being processed per active particle should be roughly the same. By assuming the active particles are a small fraction of the whole system, the number of event being processed during a fixed period of time on each thread is proportional to the number of active particles in the region associated with the thread. Thus, for each time slice, the running time is determined by the number of active particles in the largest cluster, which will be referred to as the size of the largest cluster. The size of the largest cluster is recorded in each time slice and the distribution of the size of the largest cluster could be constructed. Let the size be s, the speed-up S is estimated theoretically by

$$S = (\sum_s p_s * s/N_{active})^{-1} \tag{10}$$

Notice that this estimation is based on the fact that the same number of steps means the same for the simulation regardless the number of active particles in the system. This is justified by figure 6.[13] This estimation oversimplifies the problem. Several corrections need to be taken into account:

- Time for assigning the regions: At the beginning of each time slice, whether there are overlap between each of the regions is determined. The time (in real life) this procedure takes is of the order of the time required for processing an event. On average, the number of events being processed during time slice Δt is

  $$N_{event} = 2\Delta t(N - N_{active}) * N_{cluster}/N_{active} \tag{11}$$

  where $N_{cluster}$ is the number of active particles in the largest cluster. And, the speed-up in each time slice is modified by a factor

  $$\frac{N_{event}}{N_{event} + 1} \tag{12}$$

- Slowing down introduced by repeat: If the target of an event is out of its region, the simulation will restart from the beginning of the time slice. And this contribution of time is included in the factor which modifies the total speed-up

  $$\frac{N_{\Delta t}}{N_{\Delta t} + N_{repeat}} \tag{13}$$

  where $N_{\Delta t}$ is the number of time slices in the simulation and N_{repeat} is the number of repeat in the simulation.

Last but not least, for a system of 10^4 particles, it takes a long time for a sampled configuration to decouple from its initial condition. A solution to this problem is to prepare several replicas of the system, each of them have different equilibrium initial states. When calculating the quantities we are interested in, the values of these quantities are averaged over the replicas. The mean speed-up is calculated by harmonic average.

[13] Strictly speaking, we need to compare the auto-correlation time instead of mixing time. However, calculating auto-correlation time in relatively large systems is time consuming. The auto-correlation time will be studied in the following works.

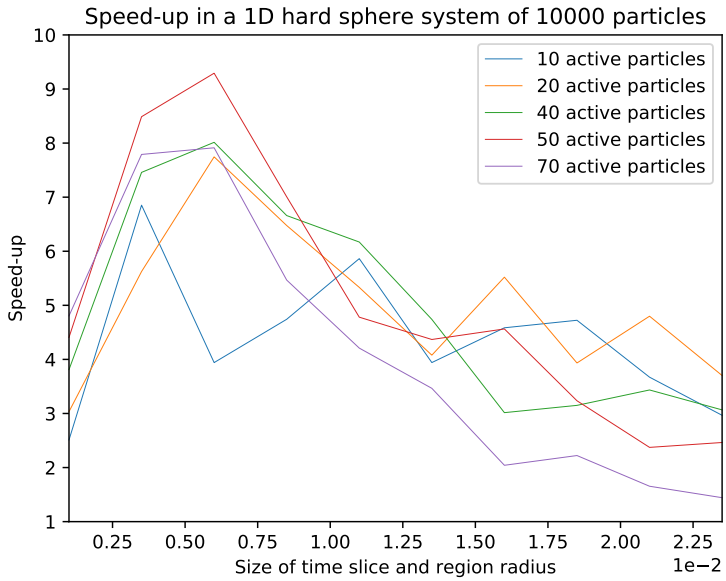

Figure 12: The speed-ups as functions of region sizes and time slices when keeping the number of particles in the system fixed. Different colours stand for different numbers of active particles. We assume the radius of the regions are identical to the size of the time slice in order to simplify the problem. There is a optimal region size, which seems to be independent of the number of active particles in the system. After reaching 50 active particles, increasing number of particles in the system no longer boosts the performance. When the number of particles in a system is fixed, the peak speed-up of dynamic domain decomposition is the highest speed-up when using optimal number of active particles and optimal region size(time slice).

We first studied the dependence of speed-up on the size of the regions, time slice and number of active particles. In principle, the size of regions should be independent to the size of time slice. In order to simplify the problem, we assume that the radius of the regions are identical to the size of the time slice. Since we assume that the active particles move at unitary speed, when an active particle moves to the right, it can never cross the boarder of its region. Doing so, we believe, could suppress the probability of repeat. The center of each region is the position of the active particle associated with the region at the beginning of the time slice. The simulation is run in a one-dimensional system with factor field composed by 10^4 particles. The results is shown in figure 12.

First of all, no matter how many active particles are there in the system, there seems to be a universal optimal region size. When the size of the time slice and region are very large, regions tend to overlap. And large cluster of active particles will slow down the simulation according to (10). When the time slice is too small, only a small number of event can be processed in each time slice. Thus, assigning regions at the beginning of each time slice will consume a considerable fraction of computational power. Meanwhile, small regions will lead to frequent repeats, which also slows down the simulation. Thus, there must be a intermediate size of the time slice and regions which maximizes the performance. This optimal size can be determined by experiment.

Secondly, notice that, while adding more and more active particles into the system, the speed-up saturates at some point. This is due to the fact that, when there are too much active particles in the system, the increase of the size of clusters has compensated

the increase of the number of regions. And if there are way too many active particles in the system, the regions percolates and the simulation becomes serial. Thus, adding more active particles will not always make the simulation more parallel.

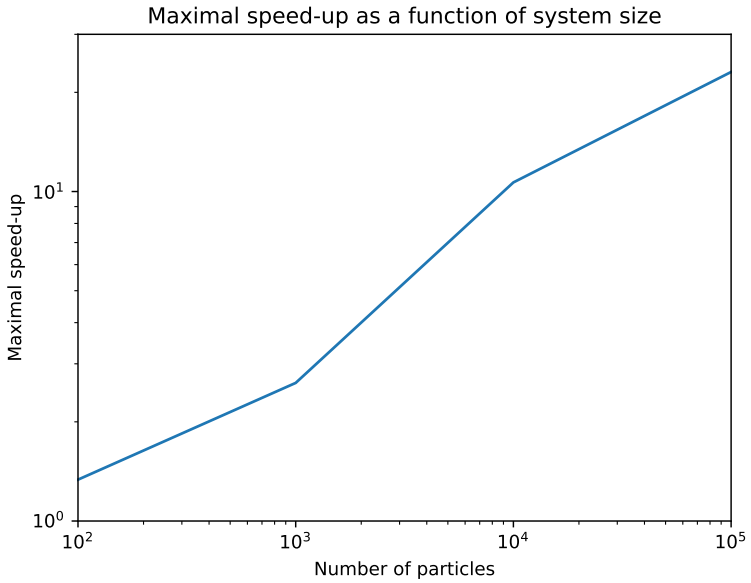

Figure 13: The peak speed-up of dynamic domain decomposition as a function of number of particles. The peak speed-up scales with the number of particles in the system.

Thus, for each system, there is a peak speed-up. The peak speed-up of dynamic domain decomposition is the highest speed-up when using optimal number of active particles and optimal region size(time slice). We plot the peak speed-up of systems made up of different number of active particles in figure 13. For a system with 10^4 particles, the speed-up could reach 10. And for a system with 10^5 particles, the speed-up is more than 20. This means, by parallelization, the simulation of one-dimensional hard sphere system could be an order of magnitude faster. Most importantly, the speed-up scales with the number of particles in the system, which means the speed-up should be higher when simulating a larger system.

4.5 Two-dimensional system

When doing simulation in an one-dimensional system, we observed that the contribution of assigning region and repeat could be ignored if the parameters in the simulation have reasonable values. Thus, the speed-up in a two-dimensional hard-sphere system could be estimated without doing real simulation. Instead, it is possible to estimate the speed-up from the size of the clusters, thus the only simulation needs to be done is to assign regions. Using the expression (10), the speed of the simulation could be accurately estimated.

We simulate the assignment of the regions in a two-dimensional system. The simulation take place in a square box, and the regions are round. Periodic boundary condition is not applied here since the impact on the result should be safely ignored (figure 10). The result of simulation is presented in the table 1. According to the estimation, if there are 1000 particles in each region and $N_{\text{active}} = 100$, the total probability of exiting regions will be roughly 10% and roughly 80 event per active particle could be processed

size of cluster	1	2	3	4	5
probability(%)	0.66	77.2	20.76	1.28	0.1

Table 1: $N = 10^6$, $N_{active} = 100$, $p = 1000$. Theoretical speed-up, ignoring repeat and assigning the regions, is roughly 45.

before the active particle exiting the region. Even if we assume the time that assigning the regions takes is equal to the time of processing 10 events, the simulation would slow down no more than 15%. The theoretical speed-up calculated from the size of the clusters indicates that the simulation could be 45 times faster than a serial simulation if 100 cores is used. Taking into account the assigning and repeat, the speed-up should be at least 30.

5 Conclusion and perspective

In this project, we have proven that, in the most general case (arbitrary interactions and arbitrary dimension), ECMC with multiple active particles satisfies the global-balance condition. Then, we proposed a parallel scheme called dynamic domain decomposition which managed to parallelize ECMC. In this scheme, each active particle is associated with a region. If two regions overlap, they will merge into one. Regions are then assigned to threads, and the events in each region will be processed in parallel until the end of the current time slice. The performance of this scheme is measured. This scheme could speed up the simulation in hard-sphere systems by orders of magnitude.

Though being successful on paper, much work remains to be done before proving that dynamic domain decomposition works in practice. Firstly, we need to study in detail dynamics of the system when there are multiple active particles. Then, there will be a program written in Rust[14] which achieves the state-of-the-art performance for the parallel one-dimensional hard sphere simulation. Then the Rust program will be generalized to handle two-dimensional hard disks. The idea of factor field will also be generalized so that it fits in two-dimensional systems. Our ultimate goal, in this project, is to reduce the auto-correlation time of two-dimensional hard-disk system for applications as in [3] by two orders of magnitude, that is, from the order of a month to the order of hours.

6 Acknowledgement

I would like to thank my supervisor Werner Krauth for giving me the chance of working with him, as well as finding funding for me, which makes the following works in this project possible. I would like to thank Liang Qin and Anthony Maggs for inspiring discussions. I would like to thank Lauranne Chaignon for acquiring Ref[13], which is crucial for this project. And I would like to thank École Normale Supérieure for hospitality. Last but not least, I would like to thank the master program of high energy physics at Ecole Polytechnique for the openness of letting me do a statistical physics project for my master thesis.

[14]Currently, all the simulation related to this project is written in Python. However, in order to achieve the state-of-the-art performance, we need a compiled language which features safe concurrency and efficient memory management. And we choose Rust.

References

[1] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of State Calculations by Fast Computing Machines. *J. Chem. Phys.*, 21:1087–1092, 1953.

[2] E. P. Bernard, W. Krauth, and D. B. Wilson. Event-chain Monte Carlo algorithms for hard-sphere systems. *Phys. Rev. E*, 80:056704, 2009.

[3] E. P. Bernard and W. Krauth. Two-Step Melting in Two Dimensions: First-Order Liquid-Hexatic Transition. *Phys. Rev. Lett.*, 107:155704, 2011.

[4] M. Michel, J. Mayer, and W. Krauth. Event-chain Monte Carlo for classical continuous spin models. *EPL (Europhysics Letters)*, 112:20003, 2015.

[5] M. F. Faulkner, L. Qin, A. C. Maggs, and W. Krauth. All-atom computations with irreversible Markov chains. *The Journal of Chemical Physics*, 149(6):064113, 2018.

[6] M. Hasenbusch and S. Schaefer. Testing the event-chain algorithm in asymptotically free models. *Phys. Rev. D*, 98:054502, 2018.

[7] Z. Lei and W. Krauth and A. C. Maggs. Event-chain Monte Carlo with factor fields. 2018. arXiv e-prints.

[8] M. Michel. *Irreversible Markov chains by the factorized Metropolis filter: Algorithms and applications in particle systems and spin models.* Theses, Ecole Normale Superieure de Paris - ENS Paris, 2016.

[9] M. Michel, S. C. Kapfer, and W. Krauth. Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *J. Chem. Phys.*, 140(5):054116, 2014.

[10] Jellyfysh collaboration. Unpublished. in preparation.

[11] D. Jeppesen. Computer Latency at a Human Scale.

[12] Wikichip. Xeon E5-2690 v4 - Intel.

[13] B. D. Lubachevsky. Simulating billiards: Serially and in parallel. *International Journal in Computer Simulation*, 2:373–411, 1992.

[14] M. F. Faulkner A. C. Maggs P. Höllmer, L. Qin and W. Krauth. JeLLyFysh (Version 1.0)- a Python application for all-atom event-chain Monte Carlo. in preparation.