Asynchronous Event-Driven Particle Algorithms in Computational Materials Science

Aleksandar Donev\textsuperscript{1}

\textit{Lawrence Livermore National Laboratory}\textsuperscript{1}

\textsuperscript{1}Lawrence Postdoctoral Fellow

Scientific Computing and Advanced Computation
Joint Mathematics Meeting, January 2009

\textsuperscript{1}This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-PRES-408395).
Outline

1. Introduction

2. Molecular Dynamics: Packing of Hard Particles

3. First-Passage Kinetic Monte Carlo: Diffusion-Controlled Reactions
Packing of M&Ms

Improving the Density of Jammed Disordered Packings using Ellipsoids, Science, 2004
A. Donev, I. Cisse, D. Sachs, E. A. Variano, F. H. Stillinger, R. Connelly, S. Torquato and P. M. Chaikin
Time-Driven Molecular Dynamics (TDMD) for soft particles (ODE integrators):

1. All of the particles are displaced *synchronously* in small *time steps* $\Delta t$, calculating positions and forces on each particle at every time step.
2. It is *not* rigorous (there is an error $\sim \Delta t$), but it is very well-understood and widely implemented.
3. Discontinuous changes of the state, aka events, occur *a posteriori*, in the middle of time steps (e.g., chemical reactions).
Asynchronous Event-Driven (AED) Algorithms

- **Event-Driven Molecular Dynamics (EDMD)** for hard particles:
  - Time is advanced from one event to the next event.
  - **Asynchronous**: Each particle is at the point in time when the last event involving it happened.
  - Given infinite numerical precision, this kind of approach can rigorously follow the dynamics of the system.

- Note: There also exist synchronous event-driven algorithms, for example, dynamic Markov chain Monte Carlo algorithms.

- Asynchronous event-driven algorithms naturally handle variable time-scales.
Molecular Dynamics (MD) Algorithm

- Event-driven MD (EDMD) packing algorithm *ala Lubachevsky-Stillinger*
- The very first MD simulations (Alder & Wainwright) were event-driven hard-disk MD!
- Non-spherical particles are much more difficult to handle than spheres (collision prediction).

*Neighbor List Collision-Driven Molecular Dynamics Simulation for Nonspherical Particles*
Each particle has its own *current time* $t$ predicts its *impending event* $(t_e, p_e)$.

Types of events: *binary collision*, boundary events, internal events, geometrical events, etc.

Each particle $i$ predicts events with particles and objects in its *neighbourhood* $\mathcal{N}(i)$.

*Collision predictions* must be kept symmetric, that is, if $i$ predicts an event with $j$, it changes $j$’s prediction as well.

*Event schedule* consists of a *priority queue* of time-ordered impending events, one for each particle.
Event Loop

1. Delete (pop) the top of the event queue (heap) to find the next particle $i$ to have an event with $p_e(i)$ at $t_e(i)$.
2. Advance the global simulation time $t ← t_e(i)$.
3. Move $i$ to time $t$, $r(i) ← r(i) + [t - t(i)]v_i$, and set $t(i) ← t$, if necessary.
4. If $p_e(i) ≡ N(i)$, then update $N(i)$.
5. If event is a wall collision, process the collision (update the momentum of $i$).
6. If event is a binary collision, then:
   1. Move particle $j = p_e(i)$ to time $t$ and set $t(j) ← t$ and mark $j$’s event as an update.
   2. Process the binary collision between $i$ and $j$.
7. **Critical**: Predict the next event for particle $i$, checking for collisions with walls and particles in $N(i)$.
8. Insert particle $i$ back into the event heap with key $t_e(i)$. 
Neighbour Search

The **Linked-List Cell** method:

1. Partition the simulation domain into cells, and bin the particles into the bins based on centroid position.
2. Cell partitioning is *independent* of the particle motion.
3. *Transfer* events monitor centroids.

The **Near-Neighbor List** (NNL) method:

1. Each particle has its *bounding neighborhood*: region of space where interacting particles may be present.
2. Each particle has a *list of neighborhoods* its bounding neighborhood overlaps with.
3. Use the cell partitioning when building the NNLs.
Some or all unit events are diffusion hops: a set of $N$ hard objects walking randomly on a lattice or in continuum space.

- Upon collision particles react (collision events).
- Example: Diffusion-controlled annihilation $A + A \rightarrow 0$.
- Great many diffusion hops necessary to bring particles to collisions at low density.

Traditional synchronous $n$-fold event-driven algorithm (BKL). Other types of Poisson events (birth, decay, boundary, etc.) are easy to handle.
Given a region of space $\Omega$, one can determine the probability distributions for when and where (on $\partial \Omega$) a particle will first leave that region (*first-event prediction*).

Given that a particle has not yet left that region, one can determine the probability of finding the particle at some point inside the region at a given time (*no-event propagation*).

For pairs of particles, reduce to two *independent center-of-mass* and *difference walkers*. 
Construct disjoint protective regions (cubes, spheres) at $t = 0$.

Main events are (super)hops to $\partial \Omega$. For each walker (particle or pair) randomly draw first passage time from the appropriate PDF (thus MC).

Find the earliest time in the queue, propagate the particle/pair to boundary/collision, construct a new protective region, insert back into queue with a new event time, repeat!

*First-passage Monte Carlo algorithm: Diffusion without all the hops*, PRL, 2006
T. Oppelstrup, V. V. Bulatov, George H. Gilmer, M. H. Kalos, and B. Sadigh
Advantages of the Algorithm

- The algorithm is **exact** for continuous diffusion problems.
- It is the first use we know of of **time-dependent Green’s functions**.
- The algorithm automatically adjusts to variable timescales: **multiscale**.
- We have a code that implements different types of reactions (annihilation, coalescence, chemical reactions, decay/emission, hard-sphere repulsion).
- Used for modeling **radiation damage in metals**: diffusing and reacting vacancies and interstitials (defects created by radiation).
The method is significantly more complicated to implement than BKL KMC and it requires analytical solutions (1-body and 2-body problems).

Multi-particle reactions cause complications or slowdown (ex., nearly triple collisions).

One can combine the asynchronous super-hops with local synchronous small hops in a **mixed time-driven/event-driven approach** (under development).

Event-driven algorithms are a very efficient alternative to traditional time-driven simulations in situations where the evolution of a system is dominated by discontinuous state changes (events).

ED algorithms are significantly more complicated than TD ones and work best for simple models such as hard-particle systems.

Time-driven and event-driven handling can be combined together: each piece does what it is best for!

Unfortunately, event-driven algorithms are not widely used and most computational scientists are not familiar with them.

AED algorithms are very difficult to parallelize!