18.337J Project Final Report:

Parallel Off-Lattice Monte Carlo Simulations

Ahmed E. Ismail and Cynthia Lo

May 8, 2002

1 Introduction

A principal goal of molecular simulations is to calculate accurately and efficiently the thermodynamics properties of a system containing many atoms or molecules. The individual particles in the system can be simple, non-interacting point particles, representing an ideal gas, or complicated connected structures, simulating large chain molecules such as polymers or proteins. By computing quantities such as free energies or densities, we can attempt to produce phase diagrams, which demonstrate the behavior of a system over a wide range of parameters such as temperature and pressure.

1.1 Grand Canonical Monte Carlo

Since we want to calculate equilibrium properties, we chose to perform off-lattice Monte Carlo (MC) simulations, which employ a sequence of random numbers to sample particle moves from a probability density function. In particular, we used the Grand Canonical (GC) ensemble, where the chemical potential, volume, and temperature are fixed, while the number of molecules fluctuates. GCMC is particularly well suited to study phenomena that occur under conditions in which energy and matter are exchanged between the system and its physical surroundings, such as as adsorption, capillary condensation, and vapor-liquid equilibrium. For simplicity, we performed our simulations on a two-dimensional system.

Monte Carlo simulations consist of sequences of trial moves: for our system, there are

three types of trial moves: relocation, creation, or destruction of a particle. Any such move is associated with a change in the energy E of the system corresponding to the change from old "microstate" m to new "microstate" n. The acceptance or rejection of this trial move is determined by the Metropolis criterion. For a relocation, the move is automatically accepted if the change in energy $\Delta E_{nm} \equiv E_n - E_m \leq 0$; if the change in energy is positive, a random number ξ is generated uniformly in $(0, 1)$, and the move is accepted if

$$\exp\left(-\beta \Delta E_{nm}\right) \geq 0;$$

the factor $\beta = (k_B T)^{-1}$ scales the energy so that the product is dimensionless.

Simulations in the grand canonical ensemble also allow us to create or destroy particles; after each trial relocation, we attempt to create or destroy particles with equal probability. The acceptance criterion for creation of a particle is

$$\left(1 + \frac{N+1}{zV} exp\left(\beta \Delta E_{nm}\right)\right)^{-1} \geq \xi, \tag{1}$$

while the acceptance criterion for destruction of a particle is

$$\left(1 + \frac{zV}{N} exp\left(\beta \Delta E_{nm}\right)\right)^{-1} \geq \xi, \tag{2}$$

where in both (1) and (2) $z \equiv \exp\left(\beta \mu\right)$ represents the *chemical activity* of the system.

1.2 Parallelization Schemes

Since large numbers of calculations need to be performed to obtain accurate thermodynamic behavior, one expects that some form of paralellization can be used to improve the efficiency of our calculations. The dominant computational cost in a MC simulation is the calculation of the energies as a function of the positions of the different molecules in the system. However, it is inefficient on distributed memory machines to divide this calculation among multiple processors, since the communication that is necessary for exchange of information

is too great, given the excessive number of times that this routine must be called (once for every particle in the system for every trial move). For shared machines, however, an OpenMP implementation which shares the loop calculation might be effective. Thus, for distributed memory machines, we look for a parallel algorithm which exploits decomposition or performs multiple simulations simultaneously.

There have been several methods proposed for parallelization, including spatial decomposition of the simulation volume, time decomposition, and parallel tempering. Spatial decomposition makes use of the fact that atoms separated by a distance greater than the maximum range of the potential are independent and can be moved simultaneously. The simulation system is divided into some number of domains; for simplicity these domains are similar in shape and proportion to the original system. Each processor is assigned one of these domains, moves an atom in its domain at the same time as other processors, and is responsible for calculating the energy of the proposed move as well as those of its neighbors. Problems arise when two or more processors simulaneously move atoms that are close to each other but reside on adjacent processors, or when particles need to be moved between processors. Heffelfinger and Lewitt [2] proposed a scheme where each processor's domain is further subdivided into 2^n subdomains. Therefore, by assigning each processor to work in the corresponding subdomains at the same time, the problem is avoided.

In contrast, time domain decomposition involves simultaneous simulations, one per processor, which are identical except for the value of the initial random number generator seed. The thermodynamic quantities are averaged over all processors. The drawbacks include long equilibration times relative to production times, and memory limitations, in which a single processor may not be able to store all the coordinates of the system in memory.

Finally, parallel tempering (PT) [3] is a new algorithm, where MC simulations are simultaneously carried out at $\mathcal{N}$ different temperatures which span the range of interest. Cycles of traditional MC moves, carried out according to the normal Metropolis acceptance criteria, are carried out as usual. Between cycles, a new type of move, representing exchanges between the configurations at different temperatures are attempted and accepted according to detailed balance. The requirements for successful PTMC simulations are that

1. The highest temperature T_n must be high enough so that all barriers on the potential

energy surface can be surmounted.

2. The spacing between adjacent temperatures T_i and T_{i+1} s is chosen such that there is about 20-40% overlap between probability density functions. This method is easily parallized, since the simulations at the $\mathcal{N}$ temperatures can be carried out on $\mathcal{N}$ processors.

2 Methodology

We chose to perform GCMC simulations, attempting first to perform spatial decomposition, and later using parallel tempering (see below). The atoms interact according to the pairwise Lennard-Jones potential, which for short-range and long-range are given as:

$$v_{ij}^{LJ}(r) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij}}\right)^{6}\right] \tag{3}$$

$$V_{LRC}^{LJ}(r) = \frac{8}{9}\pi N\rho\sigma^{3}\epsilon\left[\left(\frac{\sigma}{r_{cut}}\right)^{9} - 3\left(\frac{\sigma}{r_{cut}}\right)^{3}\right] \tag{4}$$

We chose to use a unit potential well depth ϵ. The system size for each processor was 1.0×1.0, and there are initially $n_{atoms} = 100$ atoms per box. The cutoff radius was $r_{cut} = 0.03$, while the Lennard-Jones diameter was $\sigma = 0.01$. The simulation temperatures ranged from $T = 300K$ to $T = 300 + 50(\mathcal{N} - 1)$, where $\mathcal{N}$ is the number of processors utilized. The activity coefficient was chosen to be $z \in \{25, 125, 625\}$ to allow for a range of possibilities in the ratio of creations to destructions.

For simplicity, the atoms are treated as point particles; although the chances of overlap are infinitesimally small, a minimum atom separation of 1×10^{-10} is enforced to prevent the repulsive pairwise energy from becoming too large.

# atoms	box size	temperature	potential energy	density	pressure
100	1.0	300	1e-5	115	34500
			7e-6	111	33300
			1.6e-5	124	37200
400	1.0	300	0.229	137	41100
			-1.72e-4	121	36300
			3.63e-2	112	33600
900	1.0	300	23.63	122	36639
			-4.84e-4	119	35700
			-0.977	119	35700

Table 1: Effect of varying initial number of particles on final density.

# atoms	box size	temperature	potential energy	density	pressure
100	1.0	300	1e-5	115	34500
			7e-6	111	33300
			1.6e-5	124	37200
100	2.0	300	2.54e-4	127	38100
			-0.808	115.25	34575
			2.35e-4	119.5	35850
100	3.0	300	2.63e-2	119.78	35933
			7.71e-3	121.11	36333
			6.12e-4	120.67	36200

Table 2: Variation of simulation box size

3 Results

3.1 Serial Code

We first wrote a serial version of the GCMC code, in order to gain experience with writing MC codes, and to get density and pressure values to compare to those from the parallel code. The results from three trials, on one processor, are shown in Tables 1-3:

Although the error bars resulting from just three data sets are large, some trends are still apparent from these observations. From Table 1, we see that the equilibrium atom density does not vary as a function of initial density—this is entirely consistent with the equilibrium sampling of a grand canonical ensemble. In fact, were this not to be so, this would indicate that there is an error in our simulation. This shows that the Lennard-Jones repulsion is too

# atoms	box size	temperature	potential energy	density	pressure
100	1.0	300	1e-5	115	34500
			7e-6	111	33300
			1.6e-5	124	37200
100	1.0	350	-6.2e-5	109	39150
			-69.47	124	43295
			2.6e-2	108	37800
100	1.0	400	2.48e-2	133	53200
			7.53e-3	126	50400
			1.1e-5	116	46400
100	1.0	450	9e-6	113	50850
			0.477	123	55350
			-794.14	118	52002
100	1.0	500	1.8e-5	128	64000
			1.8e-5	128	64000
			0.632	142	70999
100	1.0	550	0.272	119	65450
			7.42e-3	126	69300
			-0.178	131	72050
100	1.0	600	2.77e-2	109	65400
			5.75e-2	114	68400
			-1.11	132	79195
100	1.0	800	4.07e-3	118	94400
			1.2e-5	119	95200
			2.58e-2	115	92000
100	1.0	1400	2.5e-5	140	140000
			-2.24	130	129993
			1.33e-2	125	125000

Table 3: Variation of system temperature

great for the close-proximity atoms in the high-loading system, and destruction occurs with a much greater probability than creation. In addition, we notice that the equilibrium system pressure increases with temperature at constant initial density and box size, as expected by the ideal gas law.

Finally, the system run times scale as a function of $O\left(L^2\right)$with box size L (t = 1.3 s for box size = 1.0, t = 12.1 s for box size = 2.0, t = 57.3 s for box size = 3.0), but not with the temperature or the density. This is because the larger boxes require a larger number of total particles, and since the change in energy is computed via a loop over all the particles in the system on a given processor, as the system size increases, the number of particles also increases. Also, the number of trial moves needed for a given iteration to be complete also increases, which leads to the $O\left(L^2\right)$ scaling.

3.2 Spatial Decomposition

We next attempted parallel spatial decomposition, but without the subdomains employed by Heffelfinger and Hewitt. Unfortunately, our use of off-lattice MC meant that the border atom positions could not be easily broadcast in array form to the adjacent processors. Therefore all atom moves, not just those of atoms on the border between two processes, needed to be broadcast to all processors, along with a flag indicating the sending and receiving processors. Each processor needed to calculate the energy change at each move, and only if the overall energy of all the processors were favorable would the move be accepted. Also, the timing results were not competitive with the serial GCMC code, so spatial decomposition was not efficient for our system. Perhaps if our system were much larger, in terms of number of particles and volume, then spatial decomposition would be more viable. As it stands, our simulation system only has a small number of particles and a relatively small box size, and the hoped-for benefits of parallelization are not evident.

#Proc.	Box Size/Proc.	Running Time	#Proc.	Box Size/Proc.	Running Time
1	2.0×2.0	53.3s	1	3.0×3.0	251.3s
4	1.0×1.0	56.0s	9	1.0×1.0	302.0s

From this, we see that the cost of communications between the various processors proved too high for the systems that we were considering. One potential cause for this is the high latency of the communications—which we now believe to be on the order of several microseconds—relative to the total size of the message, which was usually on the order of one kilobyte or less. Since these communications had to be carried out after every Monte Carlo move, the total costs of the communications between the processors greatly outweighed the computational benefit of splitting up the work among multiple processors. However, this method may still prove useful for very complicated molecules, with many degrees of freedom, for which the interaction potential v is more difficult to calculate than the Lennard-Jones potential ν_{LJ}.

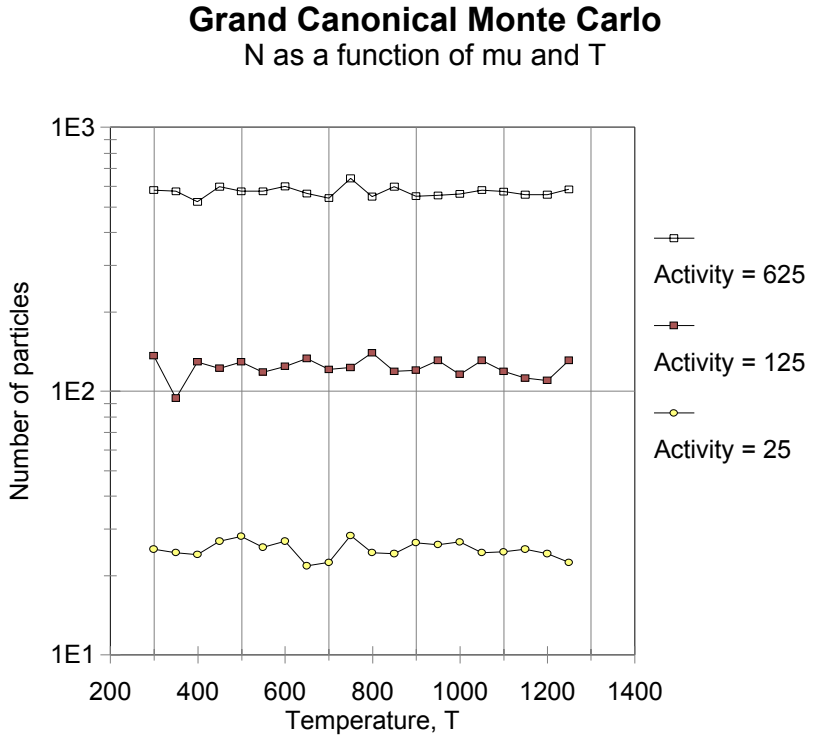
3.3 Parallel Tempering

Discovering that spatial tempering did not yield successful results upon parallelization, we decided to next implement a parallel tempering algorithm. Our primary goal was to show that the equilibration of such an algorithm could be performed as rapidly for many systems, running at a variety of temperatures, as for a single system. The test would be that the resulting energy distributions are essentially independent of the temperature, representing the overall equilibrium of the system.

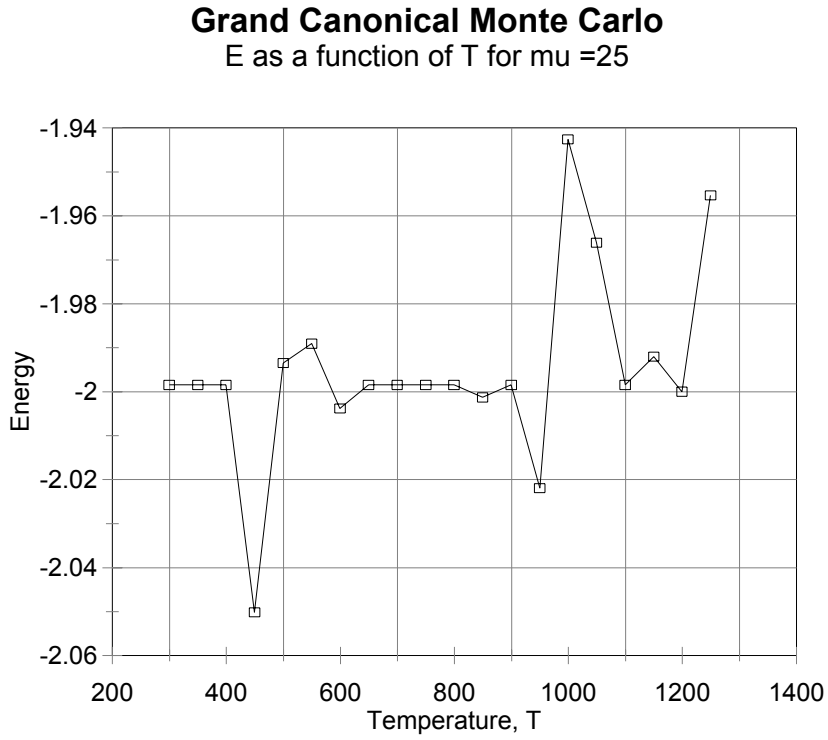
As the basis for our testing, we implemented the algorithm for 10000 cycles, each consisting of 5000 MC trial moves (particle relocation followed by particle creation or destruction attempts). The resulting graphs, at activities of $z = 25$, $z = 125$, and $z = 625$ for the temperatures $T = 300K$ through $T = 1250K$, at $50K$ intervals, are plotted in the accompanying figures. The computation times are indicated in the following graphs.

#Proc.	Running Time ($z = 25$)	#Proc.	Running Time ($z = 125$)
1	387.2s	1	1384s
8	436.8s	8	1396s
16	440.9s	16	1402s
20	447.0s	20	1512s

These results are excellent, since there is only an increase in running time of the order of 10% to 15%, while the number of systems brought to equilibrium has increased by a factor of 20. Consequently, the scaleup is a factor of approximately 17.5 for a system with 20 processors.



For the $z = 25$ system, we also show that the energy values are within a 3% band for the range of temperatures for which we have performed the simulation, which indicates that the system has come to equilibrium.



4 Conclusions and Future Work

Our work suggests that the spatial decomposition algorithm for grand canonical ensemble simulations is of practical interest only for large systems or systems with complicated interaction schemes for which the cost of communications is in scale with the savings realized by dividing the workload among multiple processors. Parallel tempering, however, represents an efficient means for bringing multiple systems to equilibrium simultaneously, as the communication costs are negligible compared to the normal costs of performing the simulation on a single system.

The natural next steps of this project would be to incorporate atoms with finite radii and hard interactions (chain molecules), instead of point particles, and to extend the system to three dimensions, as well as optimization of the MPI code for parallel tempering. Our code provides a solid basis for further thermodynamic studies of gases and liquids, and will be applicable to our thesis research.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids*. Oxford University Press, 1987.

[2] G. S. Heffelfinger and M. E. Lewitt. A comparison between two massively parallel algorithms for Monte Carlo computer simulation: An investigation in the grand canonical ensemble. *J. Comp. Chem.*, 17(2):250–265, 1996.

[3] K. Hukushima and K. Nemoto. Exchange monte carlo method and application to spin glass simulations. *J. Phys. Soc. Japan*, 65(6):1604–1608, 1996.