MONTE CARLO SIMULATION OF IDEAL POLYMER CHAIN USING PARALLEL COMPUTING TECHNIQUE

B. Karasulu and S. Uyaver

T.C. Maltepe University, Faculty of Engineering, Marmara Eğitim Koyu, 34857 Maltepe, Istanbul, Turkey
E-mail: uyaver@maltepe.edu.tr, uyaver@gmx.net
(Received 6 October 2007)

Abstract

The importance of computer simulations in pure science is well recognized today and its popularity is increasing day by day. This is because it plays an important role between theory and experiment. Since theoretical predictions are made mainly for large systems and experiments include real systems which are too large and complex, simulation works are required for larger and more complex systems. Although modern computing machines work at very high speeds, they still cannot perform simulations for such complex systems. At this point one needs to divide the job into smaller parts and run them in parallel. Polymer physics is one of the areas where the computer simulations are widely used. Generally speaking, any polymeric system is quite large and complex. The simplest system is the model called ideal polymer chain, which can be said the base to understand other more complex polymer systems. This work accomplishes Monte Carlo simulation of an ideal polymer chain using parallel computing technique. The results are compared to the theory and to the results from the serial computation. The performance analysis of the parallel computing is made.

1. Introduction

Traditionally softwares are written for serial computation. This kind of softwares is run on a single CPU (Central Processing Unit). In a serial computation instructions are executed in a sequence. Serial computation may require very high speedy computers especially when computations are a bit complex and large. In fact, any computational task which is related to a real problem, such as computational study of a biological system or a study in environmental science, can be generally said extremely large. For example, simulation of such systems can be expected to last for several months or even years. A serial computing unit which can be said to be sufficient for such computations is very rare and extremely expensive.
The idea comes that one can split the task into many pieces and make the smaller pieces of the overall computation simultaneously. In this case one can obtain the results in a shorter time or can get the results for larger systems in the same time. In this respect, this kind of parallel computing has been more powerful in many research areas.

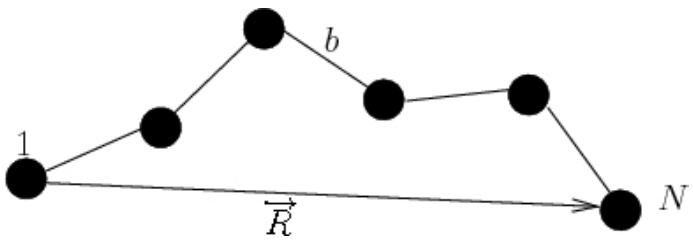


Fig. 1. Representation of a freely-jointed ideal polymer chain.

Although any real polymeric system is very complex and large, an ideal polymer chain is a very simple system [1]. The ideal polymer chain is well developed theoretically. The theoretical predictions are well observed in experiments and also confirmed by computer simulation works [2]. On the other hand, one can say ideal polymer chain can be a starting point to understand other more complex polymer systems.

2. Problem and method

In order to investigate the details of parallel computing technique, Monte Carlo (MC) simulation of an ideal polymer chain is chosen[1]. An ideal chain consists of N freely-jointed links, each of length b and able to point in any direction irrespective of each other (see Figure 1). Theoretical root-mean-square (rms) end-to-end distance R for this ideal chain is

$$R = \sqrt{\langle R^2 \rangle} \approx bN^{1/2}, \tag{1}$$

where N is the chain length. These theoretical predictions are well observed in many experiments and also the simulation works are in a good agreement [1, 2, 4].

Since any polymer system is very complex, the modeling of such systems is quite complicated. Existence of several interaction terms makes these systems very hard to be studied via computers. In this respect, we may apply the technique of parallel computing. In our problem, we have chosen the ideal polymer system and pay more attention to the parallel computing. The obtained results are confirmed by the theoretical prediction and also checked against the ones from the serial computation.

Here, in order to apply the parallel computing technique, we study two different chain lengths. One is N=480 and the other is N=960. The reason to choose two different systems is to see the importance of parallel computing for larger systems. In simulating the polymer model, standard Metropolis Monte Carlo methods [5] have been used. Changes are made to the polymer configurations using the pivot move. The pivot move makes that one monomer is picked randomly and one chain end, e.g. to the right of the picked monomer, is rotated as a rigid body at random orientation and angles (see Figure 2).

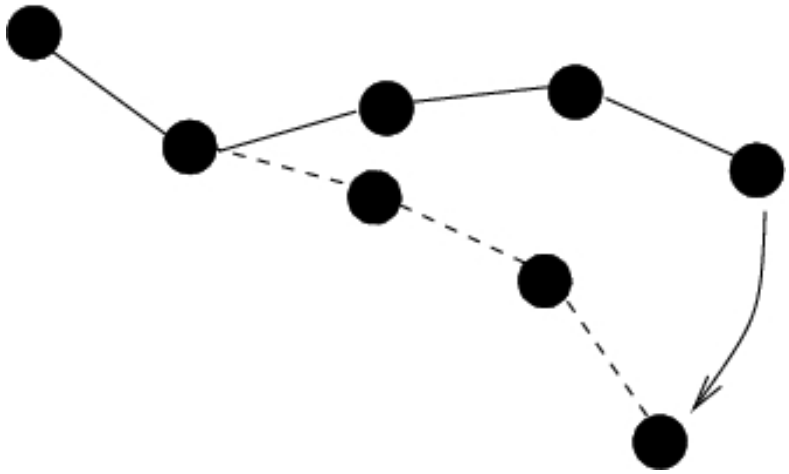


Fig. 2. Trial move used in the simulation of the chain. Right hand of the chain to the randomly picked monomer is rotated as a rigid body at random orientation and angles.

[1]Regardless of the level of complexity, the three dimensional structures of a polymer chain will determine many of its most important properties. The Monte Carlo methods have shown great promise in studying polymer conformation problems [3].

Simulations are started from arbitrary configurations. After we reach the equilibrated states, we start to measure end-to-end distances for different chain lengths using the same information over the chains 480 or 960.

3. Results

The serial and parallel computation were accomplished in the machines "Intel Pentium 4, 3.2 GHz, 512 MB RAM". In the case of parallel computation, we had the opportunity to obtain up to 25 nodes, all are connected via 10/100 Mbit/s. The codes were written in C language and MPI (Message Passing Interface) was implemented for the parallel computation.

3.1. Serial computation

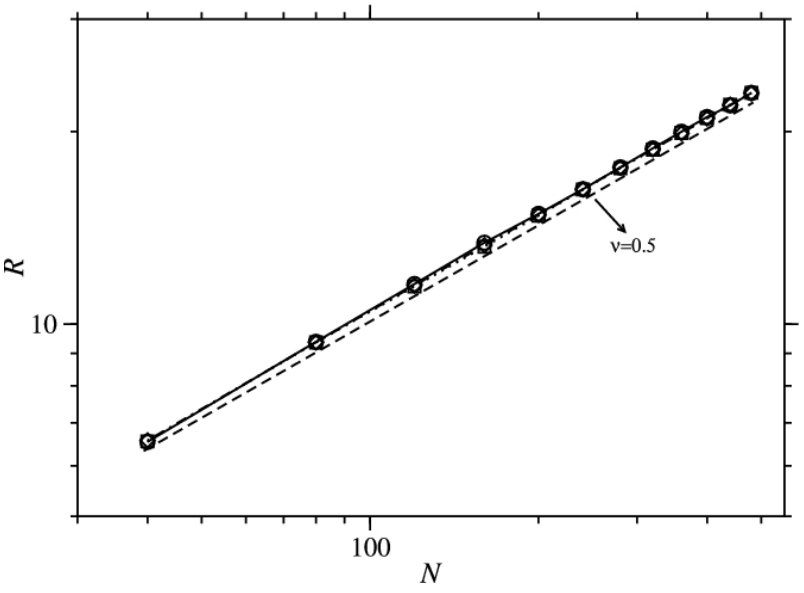
In the serial computation the instructions are executed on a single CPU. We start the simulation from an arbitrary configuration and observe end-to-end distance. When we are convinced that the system is equilibrated well, we start the main simulation part, where we measure the averaged values. The pseudo-code is given in Figure 3.

```
procedure simulate
        start with arbitrary configuration
        equilibrate the system
        make pivot move
        calculate end-to-end for different lengths
        obtain averages
end
```

Fig. 3. Pseudo-code for the serial computing.

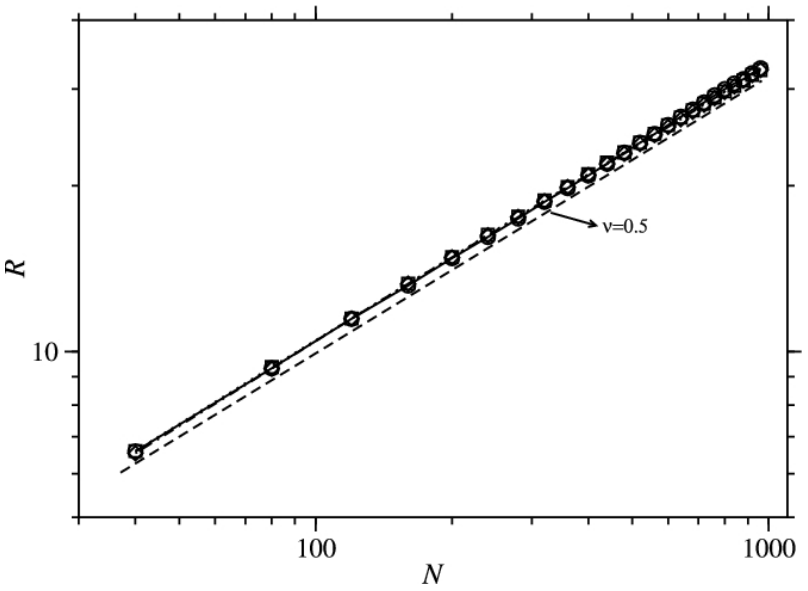
We make three different runs for the chains 480 and 960 length. They are 5000 MCM (Monte Carlo step per monomer), 15000 MCM and 25000. The corresponding end-to-end distance vs. number of monomers is shown in Figures 4(a) and 4(b).

As the theory predicted, the asymptotic behavior is seen as ν=1/2 very well.



(a) N=480

Fig. 4. End-to-end distance vs. number of monomers for an ideal polymer chain. Circles obtained from the serial



computation with MCM=5000, squares from MCM=15000, and diamonds from MCM=25000. The dashed lines are representing the asymptotic behavior, ν=1/2.

(b) N=960

3.2. Parallel computation

The serial computation is parallelized: The idea is to split the serial computation into smaller parts. Here we spit the computation as follows. The part for reaching the equilibrated states is done by the processor, say P0. The same processor starts the main simulation part. After it makes pivot and metropolis criteria, it sends the information on the system to the other processors, say Pi (i=1,2, …). The processors Pi receive the information and they calculate the required quantities. This procedure is looped up to the total MCM. Before the final, rms averages are calculated and the results are returned to the processor P0. The pseudo-code can be given as Figure 5.

```
procedure simulate
        if(P0) then
        start with arbitrary configuration
        equilibrate the system
        make pivot
        send information to Pi's
        collect averages from Pi's
        else
        receive information from P0
        calculate end-to-end distances for different lengths
        obtain averages
        send averages to P0
        endif
end
```

Fig 5. Pseudo-code for the parallel computing. P stands for processor.

The corresponding chain length dependences are shown in Figures 6(a) and 6(b). Note here that we display only the results for 480 and 960 chains simulated at 25000 MCM with 4 and 25 processors. With the other number of processors we observe no difference in the results. As obtained in the serial computation, we observe the same asymptotic behavior in a very good agreement with the theory.

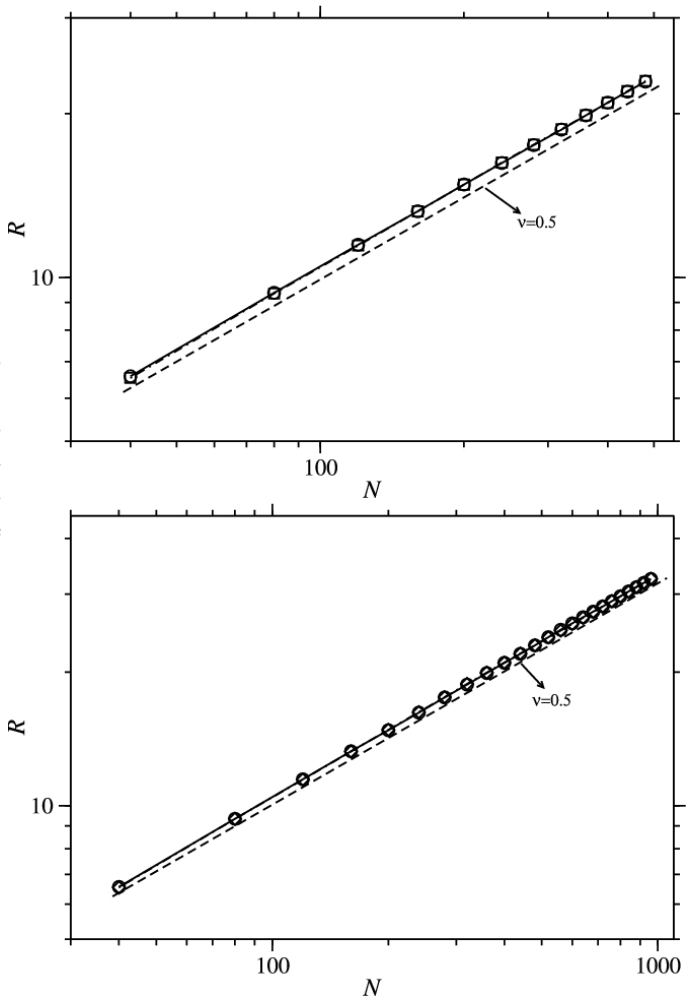


Fig. 6. End-to-end distance vs. number of monomers for an ideal polymer chain. Results shown are from the parallel computation with MCM=25000. Circles with 4 processors (n_p=4), squares with 25 processors. The dashed lines are representing the asymptotic behavior, ν=1/2.

The speed of a program is the time it takes the program to execute. This could be measured in any increment of time. Speedup is defined as the time it takes a program to execute in serial (with one processor) divided by the time it takes to execute in parallel (with many processors). The formula for speedup is

$$S_p = \frac{T1}{TN}, \tag{2}$$

where *T1* is the time it takes to execute the program when using only one processor and *TN* is the time when using *N* time [6, 7].

The speed-ups are shown in Figures 7(a) and 7(b). In the speed-up figures, we observe nonlinear dependence on number of processors. At a critical value of n_p, we have the maximum speed-up. Here it is obvious that one wishes to know the optimal number of processors in the computation. In our case, for 480 we see 4-5 processors are the optimal and for 960 we see 9-10 processors are the optimal.

In order to explore speedup more, we shall do a bit of analysis. If there are n_p workers working on a project, we may assume that they would be able to do a job in $1/n_p$ time of one worker working alone. Now, if we assume the strictly serial part of the program is performed in *B*T1* time, then the strictly parallel part is performed in $((1-B)*T1)/n_p$ time. With some substitution and number manipulation, we get the formula for speedup as

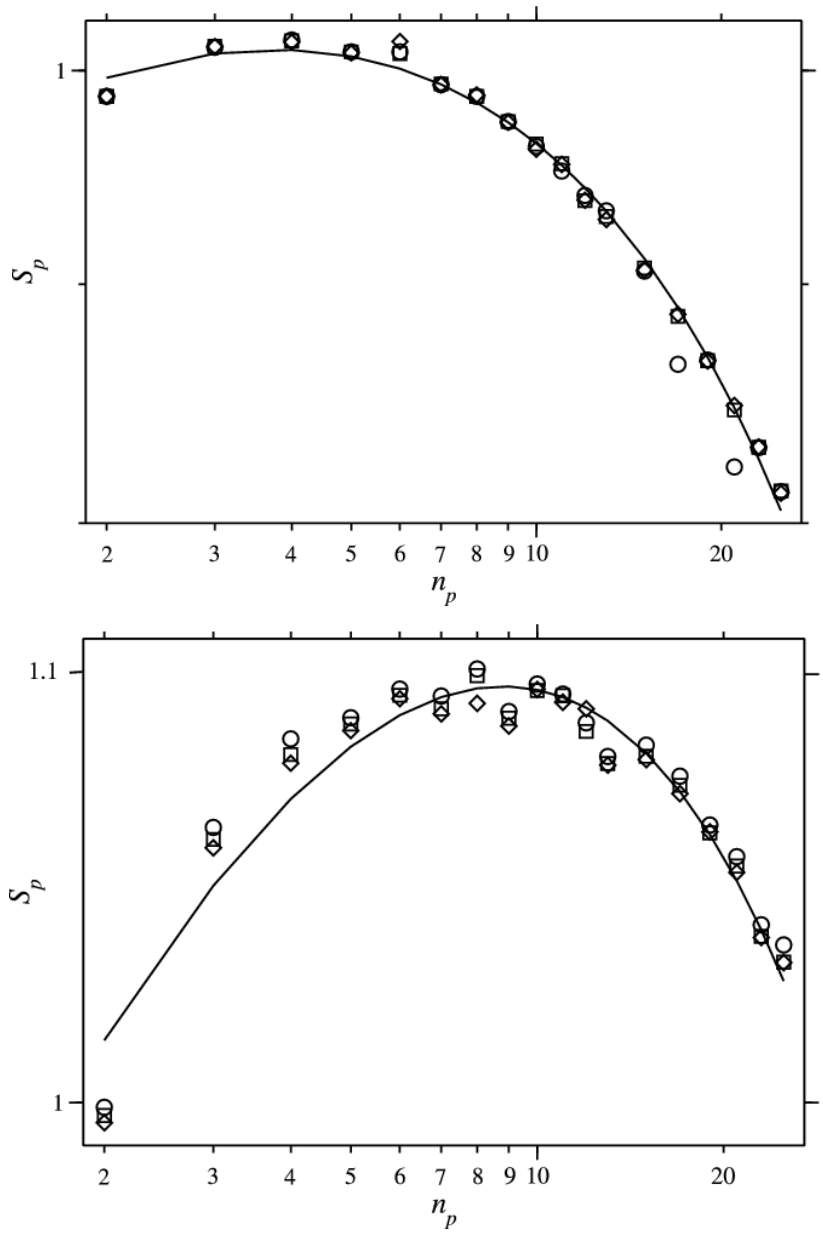


Fig. 7. Relative speed-ups for the chains *N=480* and *N=960*. Circles are for MCM=5000, squares for MCM=15000, and diamonds for MCM=25000.

$$S_p = \frac{n_p}{(B * n_p) + (1 - B)}. \tag{3}$$

For our system, we obtain the best speed-up for the chain 960 when we make the simulation the longest, which is MCM=25000 in our work. As seen in Figure 7(b), this relative speed-up is 1.10 with n_p=9. The corresponding serial part for this run is shown in Figure 8.

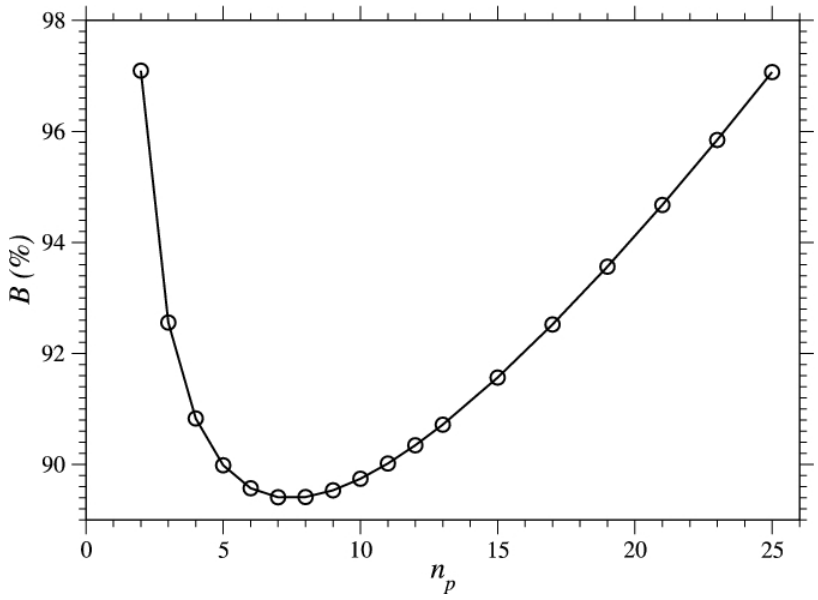


Fig. 8. The serial part (*B %*) of the computation for the run of *N=960* and MCM=25000.

Here for n_p=9, which we have the best speed-up, we see that the serial part of the computation is about 89 % the total computation. Since we are investigating a polymer chain, in which the monomers are bound to each other, the serial part of the total computation is always going to be major. We should note that for the more complex polymer systems including several interaction potentials, one can find several ways to decrease the serial part of the total computation. In order to show from our model, let us only take the part in which we calculate only the end-to-end distances for different chain lengths and obtain averages. In this situation, we certainly have very well speed-up values (see Figure 9). In this plot, we show only for *N=960*. Here we have the highest speed-ups about 3 corresponding to $B \simeq 28\%$.

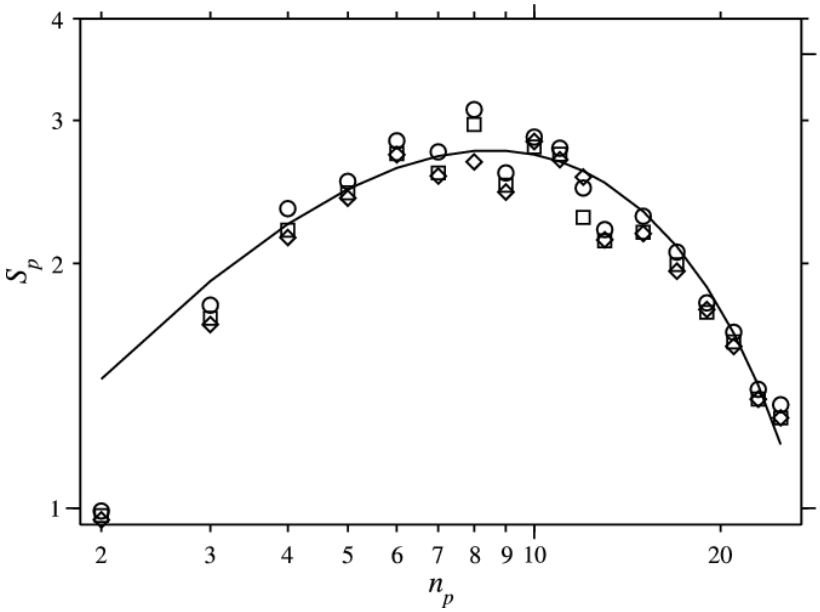


Fig. 9. Relative speed-ups for the chain *N=960* when we consider only the part in which end-to-end distances are calculated and averages are obtained. Circles are for MCM=5000, squares for MCM=15000, and diamonds for MCM=25000.

Finally, we would like to demonstrate two snapshots obtained from the parallel works. As seen in Figures 10(a) and 10(b), the coiled structures for *N=480* and *N=960* are seen well.

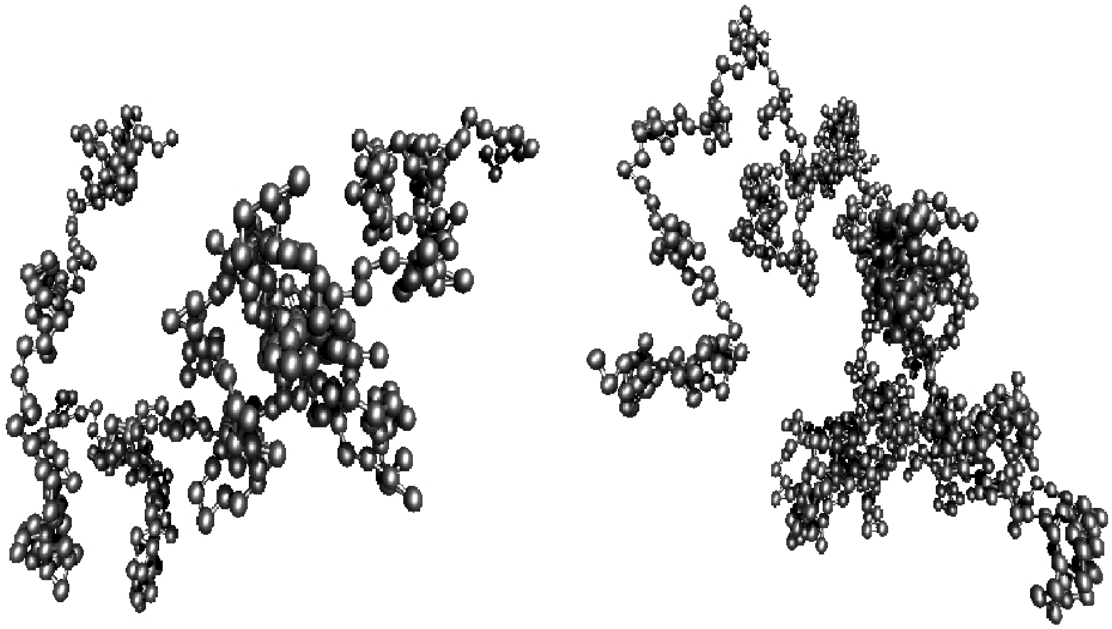
4. Conclusions

Parallel computing is obviously the technique one may require when larger and more complex systems are included in the computation. It is quite easy to implement using MPI library functions, but one has to investigate his/her system very well in detail.

We conclude that parallel Monte Carlo simulations are producing the same scientific results without any doubt. Here it seems quite important that one should consider the size and the duration of the computation. As seen in our results, for smaller systems one may not gain too much in paralel computation. In this case there is no advantage to implement the parallel computation.

On the other hand, since in the parallel computation there is always a serial part remaining, in order to achieve high speed-ups, one may like to make the serial computation part as smaller as possible. At this point, a detailed work on the problem or the system should be done. In our problem, since we have bonds between the monomers, we cannot consider the system a bulk of particles.

In our work, parallel Monte Carlo simulation of an ideal polymer chain shows once again that the ideal polymer chain obeys the scaling law, where the end-to-end distance varies with the square root of *N*, number of monomers.



(a) N=480 (b) N=960

Fig. 10. Snapshots obtained from the parallel MC simulation of the chains *N*=480 and 960.

We acknowledge T.C. Maltepe University and TUBITAK ULAKBIM High Performance Computing Center for the numerical calculations reported in this paper.

References

[1] M. Doi, Introduction to Polymer Physics, Clarendon Press, Oxford, 1996.
[2] M. Doi and S.F. Edward, The Theory of Polymer Dynamics, Clarendon Press, Oxford, 1986.
[3] K. Binder and A. Milchev, J. Comput. Aided Mater. Des., 9, 33, (2002).
[4] P.G. de Gennes, Scaling Concepts in Polymer Physics, 2nd edition, Cornell University Press, Ithaca, 1985.
[5] D.P. Landau and K. Binder, A Guide to Monte Carlo Simulations in Statistical Physics, Cambridge University Press, New York, NY, ISBN 0-521-65366-5, 2000.
[6] I. Foster, Designing and Building Parallel Programs, 1st edition, Addison Wesley, 1995.
[7] T.G. Lewis and H. El-Rewini, Introduction to Parallel Computing, Prentice-Hall, 1992.