Crystallization on a sphere using the simulated annealing algorithm implemented on H.P.C. systems

Voogd, J.M.; Sloot, P.M.A.

Publication date
1993

Published in
European simulation symposium

CRYSTALLIZATION ON A SPHERE USING THE SIMULATED ANNEALING ALGORITHM IMPLEMENTED ON H.P.C. SYSTEMS

J.M. Voogd, P.M.A. Sloot
Parallel Scientific Computing and Simulation Group,
Faculty of Mathematics and Computer Science, University of Amsterdam,
Kruislaan 403 1098 SJ Amsterdam

ABSTRACT

The research presented here is a comparison of the scalability of the simulated annealing algorithm on a vector super computer (CRAY Y-MP) with the scalability of a parallel implementation on a massively parallel transputer surface (Parsytec GCel with 512 nodes of type T805).

Some results of the annealing procedure applied to the crystallization of Lennard-Jones particles on a sphere are presented.

1 INTRODUCTION

The application that we are working on is a simulation of crystallization with spherical boundary conditions. This is implemented with a simulated annealing (S.A.) algorithm. Since this is a problem that requires an enormous amount of computing power even for modest problem sizes, we started looking for methods of speeding up the calculations.

In this work we will give the background of our research in section 2. In sections 3 and 4 we will discus the functional and implementation aspects. Sections 5 deal with the theoretical time complexities. In section 6 we give the results of the measurements of the vector implementation and the parallel implementation. Finally in section 7 we will give the conclusions and some brief results of the optimal distributions found with the S.A. engine.

This research was funded by the NCF (Stichting Nationale Computer Facitliteiten) and the FOM (Fundamenteel Onderzoek der Materie).

2 BACKGROUND

Particle dynamics simulations with spherical boundary conditions for large numbers (10^3 - 10^6) of particles (e.g. molecules) with Lennard-Jones or similar interactions at high density, provide an important testing ground for the study of closed 2D systems. Spherical boundary conditions have been used as an alternative to periodic boundary conditions to approximate bulk systems. Although the spherical topology of the boundary conditions has only a limited effect on the properties of bulk systems, they do affect the properties of the crystalline state in an essential way (for example by inducing global symmetry). Particularly the study of hierarchical clustering and the ordering of defects in a spherical matrix is a challenging ‘close packing’ problem.

The hypothesis has been put forward [van Dantzig] that in spherical bilayer vesicles (spontaneously formed from fragmented biomembrane material), hierarchically ordered arrangements of the lipid matrix may be induced in the most densely packed ('backbone') shell of the vesicles. Such patterns result from highly non-linear co-operative effects. Like in the case of viruses, these patterns may be very specific, reflecting the optimisation in enthalpy and entropy in the self-assembly process. The study of spherical crystallisation can test the hypothesis on hierarchical regular polyhedral arrangements. In particular, it may help to understand the quite remarkable observation of 'quantum' jumps in the size of biomembrane vesicles [Bont, 1978].

To study these types of phenomena we have to address the long standing problem of equilibrium arrangements of particles on a sphere under the influence of two-body forces. In the case of Coulomb interactions, the problem resembles the mathematical exercise of spreading a given number of points equally spaced over a spherical surface. This equivalence relates the physics of the problem to a geometrical principle. It may be understood by considering the geodesic connecting lines between N points, giving a spherical polyhedral net with N vertices. Below N = 20 one has perfectly regular polyhedral nets corresponding to the Platonic solids with 4, 6, 8, 12 and 20 vertices. Beyond N = 20 no perfectly regular polyhedral net have been observed.

Many problems originating from physics, chemistry and mathematics, like the experiments that we are working on, crystallisation of particles, can be formulated as optimisation problems. A vast majority of these problems involve the determination of the absolute minimum of a underlying multidimensional function. Usually optimisation of these complex systems is far from trivial since the solution must be attained from a very large irregular candidate space, containing many local extrema. As a consequence the computational effort

required for an exact solution grows more rapidly than a polynomial function of the problem size, the problem is said to be NP (non-polynomial time) complete. Because it is impossible to examine all solution candidates, even in principle, approximation methods are required.

A well established computational scheme is the Simulated Annealing (S.A.) method, a stochastic optimisation procedure that mimics the essentials of physical annealing. In physical annealing a material is heated to a high temperature and then allowed to cool slowly. At high temperature the molecules move freely with respect to one another. If the liquid is cooled slowly, thermal mobility is lost. The molecules search for the lowest energy consistent with the physical constraints.

Although S.A. guarantees the finding of the global minimum, the time required for the algorithm to converge increases rapidly with increasing number of particles and/or local minima. In the biophysical problem we are dealing with, the number of particles to investigate is typically larger than 10^5 and the number of local minima is also very large. Therefore conventional annealing implementations do not apply and more efficient methods need to be investigated. A method to lower the computational time is to use vector super computers. With the new breed of parallel machines and programming paradigms, other very fast implementations come within reach.

3 FUNCTIONAL ASPECTS

3.1 The sequential version

The S.A. algorithm for solving combinatorial optimisation problems was formulated by [Kirkpatrick et al., 1983]. It was based on a method developed by [Metropolis et al., 1953] to study the equilibrium properties of very large systems of interacting particles at finite temperature. In terms of the crystallization problem at hand the procedure works as follows. First N particles are randomly placed on a virtual supporting sphere. The annealing begins by creating a Markov chain, of given length, at a certain temperature. The Markov chain grows by randomly displacing particles and calculating the corresponding change in energy of the system. To calculate ΔE the energy of the particle before the move and after the move is required where ΔE is the energy of the new situation minus the energy of the current situation. Both energy calculations take N-1 two-body potential energy calculations. The moves are accepted with probability $P(\Delta E,T)$ at temperature T according to the following scheme :

$$P(\Delta E, T) = \exp(-\Delta E / T) \qquad \text{if } \Delta E > 0 \qquad (1)$$

$$P(\Delta E, T) = 1 \qquad \text{if } \Delta E < 0 \qquad (2)$$

Unaccepted moves are undone. This choice of $P(\Delta E,T)$ guarantees that the system evolves into the Boltzmann distribution [Metropolis et al., 1953].

After a certain number of steps the radius is perturbed. This is done by calculating the energy of the system at a randomly generated new radius and subtracting the current energy. Acceptance is also decided according to the probability scheme given above.

After a chain has ended the temperature is lowered by multiplying the temperature with the cool-rate, which is a number between 0 and 1 (typically 0.9) after which a new chain is started. This process continues until a stop criterion is met. The stop criterion in our implementation is met when the standard deviation in the final energies of the last ten chains falls below a certain value (typically 10^{-6}). The energy of the system is defined by the energy per particle. This removes the dependency of the energy on the number of particles such that the stop criterion can be fixed.

3.2 The vector version

The vectorized version is based on the sequential algorithm described above. The SIMD-Parallelism of the target machine, the CRAY Y-MP, is parallelism on the level of floating point operation. The add and multiply units are divided into components, this is the basic idea of pipelining. In a vector computer one instruction is issued to a functional unit, for example a floating-point add, for a series of operands. The functional unit will generate the results for each operand at the rate of typically one result per clock cycle.

The performance of a program on a vector computer depends on many elements of the hardware design. Among these are: the size and number of vector registers, number of concurrent paths to memory, instruction issue rate and number of duplicate functional units (multiple 'vector pipelines'). The execution time of any vector instruction consists of the start-up time and a processing time that is proportional to the vector length (number of elements of the vector).

3.3 The parallel version

The parallel version actually consists of two kinds of decompositions, the first is a decomposition of the Markov chains (systolic), the second is a functional decomposition of the energy calculation in a step in the Markov chain.

A synchronous algorithm that does not mimic sequential annealing is systolic S.A. [Aarts et al. 1986]. In systolic S.A. a Markov chain is assigned to each of the available processors. All chains have equal length. The chains are executed in parallel and during execution information is transferred from a given chain to its successor. Each Markov chain is divided into a number of sub chains equal to the number of available processors. The execution of chain k+1 is started as soon as the first sub chain of chain k is generated. Equilibrium is not yet established by then. Quasi-equilibrium of the system is preserved by adopting intermediate results of previous Markov chains.

Let P be the number of processors, L the length of the Markov chains, SL = (L/P) the length of the sub chains, T_k the temperature in Markov chain M_k, and $X_{k,m,i}$ the i-th configuration vector of sub chain $M_{k,m}$.

The first configuration of a new sub chain, $X_{k,m,1}$, m>1, is either the last configuration of the previous sub chain, $X_{k,m-1,SL}$, or the final configuration of the last generated sub chain of M_{k-1}, $X_{k-1,m,SL}$. The choice is made on the same probability scheme as given above in formulae (1) and (2) with $\Delta E = E_{k-1,m,SL} - E_{k,m-1,SL}$. The overlap enhancement [Kim et al., 1990] consists of transferring the final configuration of M_{k-P} to M_k. M_k chooses probabilistically between $X_{k-P,P,SL}$ and $X_{k-1,1,SL}$. We have used a fixed cooling schedule based on the cool-rate.

In addition to this parallel algorithm we have exploited the parallelism that can be obtained from an ordinary Monte Carlo algorithm. Here the most time consuming part of the program is the calculation of the energy difference resulting from the perturbations. Since these calculations are independent we parallelize this part of the program by functional decomposition [ter Laak et al.,1992].

4 IMPLEMENTATION ASPECTS

4.1 The vector version

The potential energy calculation turns out to be the most time consuming part of the simulation according the S.A. algorithm. The potential energy is calculated on two different occasions: for each move of a particle the potential energy for that particle has to be calculated and for perturbing the radius we need the energy of the total system. The total energy calculation is implemented as an outer loop with index 'i' running from 1 to N-1, and an inner loop 'i'+1 to N. The inner loops are done in vector mode. The implementation of the inner loops is equal to the implementation of the calculation of the one particle potential as described above.

In the inner loop the condition is imposed that the distance between two particles is smaller than the cut-off distance. The actual calculation of the Lennard-Jones potential, is only executed when the condition is true. The state of the condition is handled by means of a vector mask operation. The vector mask register elements are set to one whenever the condition is true. Only those vector elements for which the corresponding vector mask register elements are equal to one are eventually stored in memory.

4.2 The parallel version

The systolic algorithm has a simple communication pattern that can be efficiently implemented as a ring. The communication overhead is small since each processor contains a complete independent database for the optimisation problem. To interchange information about the intermediate state it only has to send and receive at the end of each sub chain.

To calculate ΔE for a particle displacement we have to do two times N-1 two-body potential energy calculations. Since the N-1 calculations are independent we can perform them in parallel. If we connect a processor farm to a master processor that generates Markov chains it can assign ΔE calculation jobs to processors in the farm. If we use a hybrid implementation, systolic S.A. with energy calculations in a farm, we need a farm attached to every processor in the systolic decomposition.

In order to cut the communication costs of the farm for the energy calculation, we use a tree configuration and let every processor hold a copy of the complete configuration. The set of N particles is divided into $\mathbf{P}_{tree}$, seven in our implementation, equally sized subsets. Each of the $\mathbf{P}_{tree}$ processors is assigned to one of the subsets. The master processor is the root of the tree. If a move is accepted an update message is sent to the slave processors together with the next move that has to be calculated. The energy calculation for a perturbation of the radius is handled analogous

5 TIME COMPLEXITIES

5.1 The sequential version

Each step in a Markov chain consists of the displacement of a particle and the calculation of the energy difference with an update if the move is accepted. The radius is not perturbed every step, in this paper we will use that the radius is perturbed once every N steps of particle moves.

The time needed for the particle move is independent of the number of particles. The energy calculation resulting from a particle move is of order N. The energy calculation

for a perturbation of the radius is of order N^2, but since this calculation is used once every N steps we can also put this contribution to the time complexity to be of order N. The complete time complexity of one step in the Markov chain can be described by :

$$\mathbf{T}_1^{seq} = \mathbf{T}_m + \mathbf{T}_e(\mathbf{N}) + \mathbf{T}_r(\mathbf{N}) = \mathbf{c}_1 + \mathbf{c}_2\mathbf{N} \qquad (3)$$

where :

$\mathbf{T}_1^{seq}$ is the time needed for one step in the Markov chain with a radius update once every N steps,

$\mathbf{T}_m$ is the time needed for displacing one particle,

$\mathbf{T}_e(\mathbf{N})$ is the time needed for calculation the energy difference of a displacement,

$\mathbf{T}_r(\mathbf{N})$ is the time needed for perturbing the radius and calculating the energy difference once every N steps.

The single step described in the time complexity analyses above has to be multiplied by the length of the chain (L) to get the time taken by producing one chain. We then have to multiply the time for one chain with the number of chains generated for reaching a stable minimum (M) to get the time complexity of the complete annealing algorithm.

L is dependent on the number of particles because the convergence slows down if the number of particles increases. In this work we will use L=50*N.

M has an almost constant value. This can be explained as follows. We adjust the radius of the sphere so that the average distance between neighbouring particles is constant with varying N. Since moving one particle mainly affects particles in the neighbourhood, because of the short range of the Lennard-Jones interaction, the average ΔE per move is independent of N. Consequently, we can keep the initial temperature fixed.

5.2 The vector version

In the vector version a limited part of the algorithm is vectorized. The total time complexity consists of the time complexity of the sequential part plus the time complexity of the vectorizable part

For the time complexity analyses we have to know which parts of the program can be done in vector mode. The perturbation of a particle which appears as a constant in the sequential time complexity has no parts that can be vectorized. The subroutines that do have parts that are vectorized are the potential energy calculation for perturbations of a particle and the radius.

The calculation of the potential energy step of one particle move consists of two potential energy calculation loops. These loops are performed in vector mode. We estimate that the number of arithmetic operations within one potential energy loop is 11 when the distance is larger than the cut-off. An additional 6 arithmetic operations are performed if the Lennard-Jones potential is calculated.

If we choose a cut-off distance of 2 (distances are scaled with the minimum of the potential energy) then on average 15 particles fall within the cut-off distance. This is the case at the density where there is a global energy minimum on a flat surface. For a large number of particles the optimal density of a sphere approximates the optimal density of a flat surface.

With these numbers and the machine parameters τ_{setup} and τ_{clock} we were able to determine the time complexity of this step [Hockney and Jesshope,1988].

To find the time complexity of the perturbation of the radius we can multiplicate the previous described time with N and an appropriate constant. This can be done since the calculation of the complete potential energy of the system involves 0.5*N*(N-1) calculations and the potential energy of a particle takes 2*(N-1) calculations.

The complete time complexity is the same as the sequential part only the two parts of one step in the Markov chain that involve potential energy calculations have to be changed :

$$\mathbf{T}_1^{vec}(\mathbf{N}) = \mathbf{T}_m + \mathbf{T}_e^{vec}(\mathbf{N}) + \mathbf{T}_r^{vec}(\mathbf{N}) \qquad (4)$$

where :

$$\mathbf{T}_e^{vec}(\mathbf{N}) = 2^{*}\left(\tau_{setup} + \left(\frac{11}{2}(\mathbf{N}-1) + \frac{90}{2}\right)\tau_{clock}\right) \qquad (5)$$

$$\mathbf{T}_r^{vec} = \frac{1}{4}(\mathbf{N}-1)\mathbf{T}_e^{vec}(\mathbf{N}) \qquad (6)$$

5.3 The parallel version

Our parallel implementation is a combination of systolic S.A. and functional decomposition of the energy calculation. First the functional decomposition is discussed.

The time for the calculation of the energy difference during a perturbation of a particle in the tree decomposition is now more than just a constant times the number of particles as in the sequential version. In fact since the work is done by more processors the time needed is devided by $\mathbf{P}_{tree}$. But a communication overhead constant has to be added. The perturbation time for the radius is handled in an analogous manner.

The time complexity of one step in the Markov chains is same as the sequential time complexity but now with the appropriate formulas for the tree decomposition.

$$T_1^{fun} = T_m + \left(c_3 \frac{N}{P_{tree}} + T_{c1} \right) + \left(c_4 \frac{N}{P_{tree}} + T_{c2} \right) \qquad (7)$$

In systolic S.A. P (the number of processors) chains are generated at the same time. This means that in the time that in the sequential version one chain is generated in the systolic version P chains are generated. Again a communication time, T_c, is associated with the implementation.

The number of chains that are generated in systolic S.A. is somewhat more than the sequential version since the algorithm has to start up the processors in the ring one by one. This means that the total number of chains is the number of chains in the sequential version and added to this is the number of processors in the ring minus one.

$$T_p^{hyb} = \left(T_1^{fun} * L/P + T_c \right) * \left(M + P - 1 \right) \qquad (8)$$

T_c can be neglected for large N because the communication contribution to the total time is much lower than that of the calculation of the sub chains.

6 MEASUREMENTS OF THE TIME COMPLEXITIES

6.1 the vector version

To find the actual time taken by the implementations we have done experiments in order to find the constants appearing in the descriptions of the previous section.

We found for the vector implementation on the Cray that :

$$T_e^{vec}(N) = 53.4 + 0.154N \ \mu s$$
$$T_r^{vec}(N) = (N-1)(26.7 + 0.039N) \ \mu s$$
$$T_m = 125.0 \ \mu s$$

The complete time complexity for the vectorized version is now given by :

$$T^{vec}(N) = (205 + 0.193 * N) * L * M \ \mu s$$

6.2 the parallel version

From the parallel implementation on the T805 transputer platform we found for the tree decomposition that :

$$T_e^{tree} = 4.4 * N + 960 \ \mu s$$
$$T_r^{tree} = 2.0 * N + \frac{960}{N} \ \mu s$$
$$T_m = 1400 \ \mu s$$

For the communication time associated with the systolic S.A. implementation we found $T_c = 75 * 10^3 \mu s$.

The complete time complexity for the hybrid implementation, systolic S.A. and the tree decomposition, is given by :

$$T_1^{tree} = \left(2.4 * 10^3 + \frac{9.6 * 10^2}{N} + 6.4 * N \right)$$
$$T^{hyb}(N) = \left(T_1^{tree} * \frac{L(N,P)}{P} + 75 * 10^3 \right) * \left(M + P - 1 \right)$$

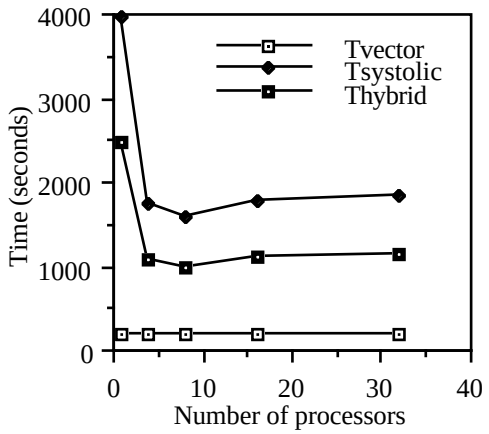
The S.A. algorithm is an inherently sequential scheme. Parallelizing it by the systolic S.A. method we introduce a functional difference to the sequential version. This parallelization has consequences for the accuracy of the iterative processes. For the vectorized version the values of M and L are the same as the sequential version and also if we use the tree decomposition nothing changes. But the systolic algorithm is functionally different. We notice that if we increase the number of processors in the ring that the number of chains that have to be generated increases. This is because the smaller the sub chains, the more chains have to be generated before a stable minimum is found. This leads to a decrease in efficiency. Together with the increase of the number of chains the found solution gets worse. This means that we actually have the dependency L(N,P) if we want to keep the same quality of the solutions. If we determine what the chain length L(N,P) should be we find that the execution times for these new chain lengths are such that there is a minimum at a certain number of processors. If more processors are used the execution time will only increase.

7 CONCLUSIONS AND SOME RESULTS

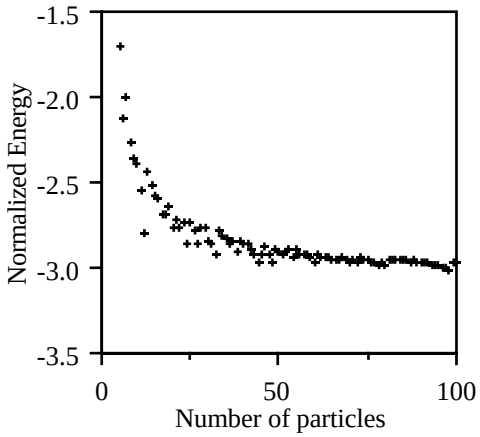
From the section about the vector time complexity it turns out that vector computing does not have the ability to be adapted to a specific problem but its power lies in the fact that you can easily adapt the sequential code. With a minimum of trouble, you can achieve high speedups.

A massively parallel machine allows various implementations. Therefore different implementations can be tested to find the one which performs best. It turns out

that the systolic implementation in the present form is not capable of using large numbers of processors. In the following figure we plot the measured times for the vector and the systolic implementation without the tree decomposition :



With small number of processors the systolic S.A. is not faster than the vectorized version. This is due partly to the fact that the clock speed of the Cray is much higher than on the T805. In a short while a T9000 based transputer platform will be installed at this institute. The clock speed of those transputers is higher that the T805 of the Parsytec GCel. Also the communication will be completely handled by hardware. Preliminary results [Bader et al,1993] show that the T9000 will be about a factor 5 to 8 faster. With those processors we will be able to outrun the Cray.



The S.A. program gives the configuration, the energy and the radius after the annealing procedure. Other output consists of the number of chains that had to be generated and the time taken by the program.

After a number of runs with different numbers of particles we can make the previous figure of the potential energy of the system versus the number of particles. It is clear that at some N there are configurations of lower potential energy than surrounding values of N.

REFERENCES

R. van Dantzig, P.M.A. Sloot and W.S. Bont, Stability and size of spherical bilayer vesicles. Manuscript submitted

W.S. Bont (1978), The Diameters of Membrane Vesicles Fit in Geometric Series, J. of Theor. Biology 74, pp. 361-375

S. Kirkpatrick, C.D. Gelatt, Jr., M.P. Vechi, (1983, May) Optimization by Simulated Annealing, Science 220, number 4598 , pp. 671-680

N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller and E. Teller (1953), Equation of State Calculations by Fast Computing Machines, J. of chem. physics, Volume 21, number 6, pp. 1087-1092.

E.H.L. Aarts, F.M.J. de Bont, E.H.A. Habers and P.J.M van Laarhoven (1986), Parallel implementations of the Statistical Cooling algorithm, North Holland Integration, the VLSI journal 4, pp. 209-238

A. ter Laak, L.O. Hertzberger, P.M.A. Sloot (1992), NonConvex Continuous Optimization Experiments on a Transputer System, Transputer Systems - Ongoing Research, ed. A.R. Allen (IOS Press, Amsterdam) p.251

Hockney, Yesshope (1988), Parallel Computers 2, Adam Hilger imprint by IOP Publishing Ltd. ISBN 0-85274-812-4

G. Bader, B. Przywara (1993, May 5), T9000 - A Preliminary Evaluation of Arithmetic Performance, Interdisziplinäres Zentrum für Wissenschaftliches Rechnen, Universität Heidelberg, Im Neuenheimer Feld 368, D-6900 Heidelberg, t9@iwrl.iwr.Uni-Heidelberg.de

Y. Kim, M. Kim (1990, june), A step-wise-overlapped parallel annealing algorithm on a message-passing multiprocessor system, vor.2(2), pp123-148