Brazilian Journal of Development

Comparing parallel algorithms for van der waals energy with cell-list technique for protein structure prediction

Comparando algoritmos paralelos para energia de van der waals com técnica de lista de células para predição de estrutura de proteína

DOI:10.34117/bjdv5n7-001

Recebimento dos originais: 10/05/2019
Aceitação para publicação: 05/06/2019

Daniel R. F. Bonetti
Doutorado em Ciência da Computação pelo Instituto de Ciências Matemática e de Computação da Universidade de São Paulo em São Carlos
Instituição: Universidade De São Paulo - Instituto De Ciências Matemáticas E De Computação
Endereço: Avenida Trabalhador São-carlense, 400 - Centro - CEP: 13566-590 - São Carlos - SP
Email: dbonetti@icmc.usp.br

Gesiel Rios Lopes
Doutorando em Ciência da Computação pelo Instituto de Ciências Matemática e de Computação da Universidade de São Paulo em São Carlos
Instituição: Universidade De São Paulo - Instituto De Ciências Matemáticas E De Computação
Endereço: Avenida Trabalhador São-carlense, 400 - Centro - CEP: 13566-590 - São Carlos - SP
Email: gesielrios@usp.br

Alexandre C. B. Delbem
Doutorado em Engenharia Elétrica pela Universidade de São Paulo em São Carlos
Instituição: Universidade De São Paulo - Instituto De Ciências Matemáticas E De Computação
Endereço: Avenida Trabalhador São-carlense, 400 - Centro - CEP: 13566-590 - São Carlos - SP
Email: acbd@icmc.usp.br

Paulo S. L. Souza
Doutorado em Física Aplicada Opção Em Física Computacional pelo Instituto de Física de São Carlos da Universidade de São Paulo
Instituição: Universidade De São Paulo - Instituto De Ciências Matemáticas E De Computação
Endereço: Avenida Trabalhador São-carlense, 400 - Centro - CEP: 13566-590 - São Carlos - SP
Email: pssouza@icmc.usp.br

Kalinka C. Branco
Doutorado em Ciência da Computação pelo Instituto de Ciências Matemática e de Computação da Universidade de São Paulo em São Carlos

Brazilian Journal of Development

Instituição: Universidade De São Paulo - Instituto De Ciências Matemáticas E De Computação
Endereço: Avenida Trabalhador São-carlense, 400 - Centro - CEP: 13566-590 - São Carlos - SP
Email: kalinka@icmc.usp.br

Gonzalo Travieso
Doutorado em Física pelo Instituto de Física de São Carlos da Universidade de São Paulo
Instituição: Universidade de São Paulo, Instituto de Física de São Carlos, Departamento de Física e Ciência dos Materiais.
Endereço: Av. Trabalhador São-carlense, 400 - Centro - CEP: 13566590 - São Carlos, SP - Brasil - Caixa-postal: 369
Email: gonzalo@ifsc.usp.br

ABSTRACT

The discovery of the structure of a protein is a difficult and expensive task, because it requires minimizing different energies related to them. The van der Waals energy hás the most expensive evaluation in this context, and computational methods have been developed in this way, such as Genetic Algorithm (GA) and cell-list technique, which reduces its the complexity from O(n2) to O(n). Even with the support of GA and cell lists, the van der Waals energy evaluation still requires a long computing time, even for a small protein. Parallel Computing is capable to reduce the runtime to predict the structure of proteins. Parallel algorithms in such context are usually specific for one programming model and computer architecture, resulting in limited speedups. This paper compares the runtime of three distinct parallel algorithms for the evaluation of an ab initio and full-atom approach based on GA and cell-list technique, in order to minimize the van der Waals energy. The three parallel algorithms are in C and use one of these programming models: MPI, OpenMP or hybrid (MPI+Open MP). Our results show that van der Waals Energy are executed faster and with better speedups when using hybrid and more flexible parallel algorithms to predict the structure of larger proteins. We also show that for small proteins the communication of MPI imposes a high overhead for the parallel execution and, thus the Open MP presents a better relation cost x benefit in such cases

Keywords: Parallel computing, Genetic Algorithms, Protein Structure Prediction, ab initio, van der Waals energy.

RESUMO

A descoberta da estrutura de uma proteína é uma tarefa difícil e dispendiosa, porque requer a minimização de diferentes energias relacionadas a elas. A energia de van der Waals tem a avaliação mais cara neste contexto, e os métodos computacionais foram desenvolvidos desta forma, como o Algoritmo Genético (GA) e a técnica da lista de células, que reduz a complexidade de O (n2) para O (n). Mesmo com o apoio do GA e das listas de células, a avaliação energética de van der Waals ainda requer um longo tempo de computação, mesmo para uma pequena proteína. Computação Paralela é capaz de reduzir o tempo de execução para prever a estrutura das proteínas. Algoritmos paralelos em tal contexto são geralmente específicos para um modelo de programação e arquitetura de computador, resultando em acelerações limitadas. Este artigo compara o tempo de execução de três algoritmos paralelos distintos para a avaliação de uma abordagem ab initio e full-atom baseada na GA e na

Brazilian Journal of Development

técnica de lista de células, a fim de minimizar a energia de van der Waals. Os três algoritmos paralelos estão em C e usam um desses modelos de programação: MPI, OpenMP ou híbrido (MPI + Open MP). Nossos resultados mostram que a van der Waals Energy é executada mais rapidamente e com melhores acelerações ao usar algoritmos paralelos híbridos e mais flexíveis para prever a estrutura de proteínas maiores. Mostramos também que para pequenas proteínas a comunicação do MPI impõe uma alta sobrecarga para a execução paralela e, assim, o MP Open apresenta uma melhor relação custo x benefício em tais casos.

Palavras-chave: Computação Paralela, Algoritmos Genéticos, Predição de Estrutura de Proteína, ab initio, energia de van der Waals

1 INTRODUCTION

The discovery of a protein structure is a difficult and expensive task, even with nowadays optimization algorithms attempting to approximate acceptable results for its structure. Genetic Algorithms (GAs) use locals and global minimal of potential energies in search spaces to estimate such structure in cyclic steps. In this context, even small proteins present avast number of possible structures in native state that a chain of amino acid can assume [1], [2], [3], [4].

The potential energy of a protein can be classified in covalent and non-covalent. Covalent bonding energies among atoms are calculated for each molecules atom and the atoms in an enclosed neighbourhood of two to four atoms. These en- ergies may be: improper energy, long energy, bonding energy, torsional energy and Urey-Bradley [5]. Energies for atoms that do not have a covalent bond are calculated for all combinations of atomic pairs of the molecule and, therefore, require more computational time. They are: van der Waals, electrostatic, solvation and hydrogen bonding [6].

Studies suggest that the Lennard-Jones potential is capable to model the interaction between pairs of atoms [7]. The van der Waals energy uses the Lennard-Jones potential to compute the configuration energy. The algorithm for van der Waals energy applies such potential to every combination of pairs of atoms in the molecule, with a O(n2) complexity (n is the number of atoms). In practice, the van der Waals energy evaluation for one configuration is fast, but the high number of evaluations performed by an optimization algorithm leads to time-consuming algorithms [8].

Aiming to minimize the runtime of such algorithms, there are different parallel solutions available in the literature [9], [10], [11]. Parallel algorithms in such context have usually distinct and limited performance, mainly because they are spe-cific for one programming model and computer architecture. In [10] and [11], instead of just parallelizing van der Waals energy from its O(n2) algorithm, the authors first improved the efficiency of

the energy using the cell-list algorithm, enabling the complexity reduction to O(n). They also reported results of van der Waals calculations by cell-list with MPI in [10] and OpenMP in [11].

This paper compares the performance (runtime), of three specific parallel algorithms for the evaluation of an ab initio and full-atom approach based on GA and cell-list technique, to minimize the van der Waals energy. The three parallel algorithms are in C and use one of these programming models: MPI, OpenMP or hybrid (MPI+OpenMP). We show, in our experiments, the importance to develop adaptive algorithms to explorer the benefits of different molecules, geometry, architectures and programming paradigms. Indeed, our results show that van der Waals Energy is executed faster when using hybrid parallel algorithms for larger proteins. For small proteins, the communication of MPI imposes a high overhead for the parallel execution and, thus, in these cases, the Open MP presents better results.

The remaining of this paper is structured as follows. Section II introduces concepts related to the van der Waals calculation. Section III shows the cell-list procedure and the van der Waals calculation using cell-list as well as its parallel version. The experimental analysis is presented in Section IV. Finally, Section V concludes this paper.

2 VAN DER WAALS ENERGY

Van der Waals energy frequently describes the energy of a molecule. The Lennard-Jones potential (also known as Lennard-Jones 12-6) allows to calculate the van der Waals energy of a molecule [12]. The van der Waals energy varies according to the distance of the pair of atoms and the type of atoms (hydrogen, carbon, nitrogen, oxygen, etc.), as shown in Equation 1, where rij is the relative distance. It is calculated by the Euclidian distance di,j between atoms i and j and the van der Waals radii constant R of atoms i and j:

$$r_{ij} = \frac{d_{i,j}}{R_i + R_j}. \qquad (1)$$

The major contributor to the energy is the distance between the pair of atoms. Van der Waals energy is highly repulsive when the atoms are close to each other since their electron clouds start to overlap, i.e., in such distances, the energy increases quickly and tends to

infinite. The minimum point of energy between a pair of atoms is known as van der Waals contact, where there is neither repulsion nor attraction. It is stabilized by the concomitant action of repulsion and attrac-tion. Distances below the van der Waals contact will produce repulsion and distances above it will produce attraction. At large distances, the pairs of atoms cannot interact with each other. Therefore the energy tends to zero (Figure 1). To avoid that, a cutoff of 8 A is often used to prevent unnecessary ˚computation. On the other hand, a tapering-off avoids the energy from assuming large numbers. If rij is smaller than 0.8 A, then the potential will assume a constant ˚C. The Lennard-Jones potential used in Protein Structure Prediction (PSP) is shown in Equation 2:

$$f_{LJ}(r_{ij}) = \begin{cases} Ar_{ij}^{-12} - Br_{ij}^{-6} \text{ if } r_{ij} > 0.8, \\ C \text{ if } r_{ij} \leq 0.8, \end{cases} \qquad (2)$$

Where A and B are constants experimentally determined based on characteristics of the environment, and C is given by Ar−12 ij − Br−6 ij with rij = 0.8.

The van der Waals energy of a molecule is given by the sum of the interaction of all pairs of atoms. It results in n $\frac{n^2-n}{2}$ interactions, where n is the number of atoms of the molecule, showed in Equation 3:

$$E_{vdw} = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} f_{LJ}\left(r_{ij}\right). \qquad (3)$$

The van der Waals energy is widely used in dynamics and molecular modeling software because, from this energy, we have a model of the behavior of the interaction between pairs of atoms of a molecule. In this way, it is possible to simulate the behavior of molecules under temperatures and

Brazilian Journal of Development

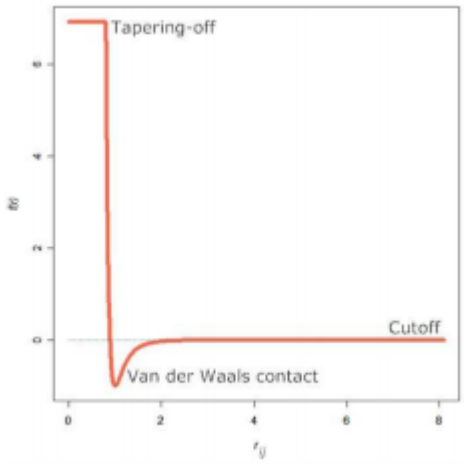

Fig. 1. Cutoff and tapering-off limits modified in Lennard-Jones (fLJ) potential.

Pressures, verifying if it can maintain its stable structure. On the other hand, the use of van der Waals energy in GAs aims to determine the quality of individuals differentiating good solutions (which have the structure similar to molecules studied in nature) and bad solutions (which are not in the most stable form of energy, presenting structures that are difficult to find in nature). Thus, we used the energy of van der Waals in the function of GA evaluation, enabling GA to find structures of proteins close to those found in nature [10], [11], [13], [14], [15], [16], [17].

3 METHOD OF CELL-LISTS

Cell-list is a general technique that enhances the efficiency of algorithms that calculate pairs of particles separated by a cutoff [18]. It creates cells of at least the cutoff length enabling the interaction of only atoms inside the cell and neighboring cells. In this study, the cell-list technique is adapted to the van der Waals calculation, in which the particles are represented by the atoms in the molecule configuration. From this point, we will use Cell-list Algorithm (CA) to describe the sequential cell-list algorithm developed, as well as Cell-list Parallel Al-gorithm (CPA) to describe the parallel version. CPA is divided into CPA with OpenMP, CPA with MPI and hybrid CPA.

CA has three major steps: i) cell-list construction; ii) allocation of the atoms to their correct cells and iii) traversal through the cells. In the first step, a vector of atoms a is taken from the molecule and x, y, z are found for the lower limits

and x, y, z for the upper limits. The set of cells is called grid and the size of the grid is denoted by the triple (Lx, Ly, Lz). By knowing the grid dimension and the cutoff it is

possible to find the number of cells in each axis. To ensure that the size of the cells is at least the length of the cutoff in each axis, the size of the cells, represented by the triple (Nx, Ny, Nz), is calculated as Nx = bLx/rcc, Ny = bLy/rcc, Nz = bLz/rcc, where rc is the cutoff. Every cell has equal dimensions and is represented by the triple (Cx, Cy, Cz), where Cx = Lx/Nx, Cy = Ly/Ny and Cz = Lz/Nz. (see Figure 2).

The atoms are then ready to be inserted in their correct cells (step ii). Considering that a is the array of all-atom coordinates in a three-dimensional space, we decompose a to represent each axis of the Cartesian space: ax, ay, az. From the atom coordinate and the length of the cells (Cx, Cy, Cz) it is possible to find to which cell each atom belongs, as shown in Figure 3. The index of each cell must be computed in x, y, z axis given by indexes i, j, k, respectively. A cell index (cijk) is computed for each atom l in the molecule, where l ranges from 1 to the number of atoms in molecule (|a|). For each atom inserted in a cell (cijk), the number of atoms of that cell (|cijk|) is increased.

Finally, instead of atoms, cells are traversed (Figure 4), and the interaction between cells is computed. When a cell is visited (also called reference cell) the Lennard-Jones potential is performed for each pair of atoms inside the cell and all its neighboring cells (step iii). Considering that Lennard- Jones potential is symmetric (fLJ , Equation 2), once the interaction between two cells is computed, both cells will not be available for further calculation. Therefore, the interactions between cells occur only in one direction. For example, at a certain point, cell cijk will interact with its neighboring cell (cijk+1). However, when cell (cijk+1) is the reference cell, the interactions between both cells will be unnecessary.

Figure 5 shows the grid of cells of size 1 × 3 × 3. The green ball in the green cell (cijk, the actual reference cell) has only three valid neighboring balls (blue balls). However, no interaction will occur with the very left ball, since it had already occurred when the left cell was the reference cell.

The only valid neighboring cells for a given cell cijk are cijk+1, cij+1k−1, cij+1k, cij+1k+1, ci+1j−1k−1, ci+1j−1k, ci+1j−1k+1, ci+1jk−1, ci+1jk, ci+1jk+1, ci+1j+1k−1, ci+1j+1k and ci+1j+1k+1. They are not valid only when the reference cell is in some point of boundary of the grid, which will occur when i = Nx or j = 0 or j = Ny or k = 0 or k = Nz (see Figure 5). To reduce the number of conditional verifications, we increase the number of cells for each axis Nx, Ny, and Nz calculated by Algorithm 1 by two, creating a boundary of

empty cells, like a skin. The cells are then indexed from 0 to N + 1 (N = [Nx, Ny, Nz]) 1. The sum of the partial energies produced by the interaction of cells will produce the van der Waals energy, as similarly produced by Equation 3.

Data: Atoms vector a: a_x, a_y, a_z
Result: Size of cell (C_x, C_y, C_z); cells per dimension (N_x, N_y, N_z); reference point $(\underline{x}, \underline{y}, \underline{z})$.

1 $\overline{x} = \max_{1 \le l \le |a|} a_x(l)$; $\underline{x} = \min_{1 \le l \le |a|} a_x(l)$;
2 $\overline{y} = \max_{1 \le l \le |a|} a_y(l)$; $\underline{y} = \min_{1 \le l \le |a|} a_y(l)$;
3 $\overline{z} = \max_{1 \le l \le |a|} a_z(l)$; $\underline{z} = \min_{1 \le l \le |a|} a_z(l)$;
4 $L_x = \overline{x} - \underline{x}$; $N_x = \lfloor L_x / r_c \rfloor$; $C_x = L_x / N_x$;
5 $L_y = \overline{y} - \underline{y}$; $N_y = \lfloor L_y / r_c \rfloor$; $C_y = L_y / N_y$;
6 $L_z = \overline{z} - \underline{z}$; $N_z = \lfloor L_z / r_c \rfloor$; $C_z = L_z / N_z$;

Fig. 2. Pseudocode of CA - Part 1: It constructs the 3D grid.

3.1 CELL-LIST IN PROTEIN STRUCTURE PREDICTION

Several Molecular Dynamic (MD) softwares [Haile 1992] use the cell-list technique to speed up the energy calculation, such as GROMACS [Spoel et al. 2004], CHARMM [Stote et al. 1999], Amber [Case et al. 2004] and LAMMPS [Sandia 2003]). The energy values of MD software and GA have different purposes. For GA, the energy is calculated so that the quality of a configuration (how much the molecule appears to be a real protein) can be evaluated. For MD, the energy describes the behavior of molecules.

MD softwares usually start from a protein already determined by experimental methods. Then, small modifications (the simulation process) occur to find the equilibrium state. The simulation starts by assigning a temperature of the system, i.e., each atom will

Data: Quantity of atoms $|a|$; atom positions a_x, a_y, a_z; cell size (C_x, C_y, C_z); reference point $(\underline{x}, \underline{y}, \underline{z})$
Result: Cells c_{ijk} filled with their corresponding atoms

1 **for** $l \leftarrow 1$ **to** $|a|$ **do**
2 $i = |a_x(l) - \underline{x}| / C_x$;
3 $j = |a_y(l) - \underline{y}| / C_y$;
4 $k = |a_z(l) - \underline{z}| / C_z$;
5 Add atom $a(l)$ to c_{ijk};
6 **end**

Figure 3. Pseudocode of CA - Part 2: It places the atoms in their correct cells.

```
Data: Cells c_{i,j,k}; cells number per dimension (N_x, N_y, N_z)
Result: Van der Waals energy E_vdw of the molecule
E_vdw = 0;
for i ← 1 to N_x do
    for j ← 1 to N_y do
        for k ← 1 to N_z do
            foreach atom pair (l, m) from c_{i,j,k}, l ≠ m do
                Compute f_LJ(r_lm);
                Add f_LJ(r_lm) to E_vdw;
            foreach atom pair (l, m) with l from c_{i,j,k} and m from one of
              neighbors N_{i,j,k} do
                Compute f_LJ(r_lm);
                Add f_LJ(r_lm) to E_vdw;
```

Figure 4. Pseudocode of CA - Part 3: It traverses the cells and computes the interactions.

Figure 5. Example of a cell-list in a 3D grid, where $N_x \times N_y \times N_z = 1 \times 3 \times 3$.

have a velocity. After these parameters have been configured, the simulation is ready to begin and will stop when a temperature equilibrium or another stop criterion has been reached. At the end of the simulation, a practitioner analyzes the data produced.

The cell grid is created for a specific molecule before the simulation begins, enabling only one construction and allocation of the cell grid for the whole simulation process. At each simulation step, only the traversal is performed during the simulation, since the velocity of the atoms can produce small changes in their positions. At some point of the simulation, the atom can jump to a neighboring cell. In that case, the atoms must be in the correct cells, otherwise wrong results will be produced.

MD software has different strategies to avoid the reallocation of atoms into cells. Some MD softwares create a cell that is a little larger than the cutoff. Therefore it is more difficult for an atom to escape from its original cell. Other MD softwares update the celllist after a fixed number of simulation steps, while the rest of them updates the cell grid only when an atom escapes from the original cell. These strategies avoid the time spent on the construction and allocation of atoms to the cells.

Brazilian Journal of Development

On the other hand, GA produces huge changes in the molecule's configuration, making the construction of a new cell grid and allocation necessary for each new molecule configuration generated. The whole process of CA must be applied to every molecule configuration before the evaluation of the solution. As a GA requires hundreds of thousands of evaluations to find a minimum sub-optimum, the CA will also be performed for the same number of evaluations. However, the repetition of steps one and two used to evaluate an individual in a GA is insignificant.

3.2. PARALLEL CELL-LIST FOR PSP

The cell-list technique reduces the configuration evaluations in a GA from $O(n^1)$ to $O(n)$. Thus, the time spent to evaluate the configurations grows linearly according to the number of atoms in the molecule. For small proteins, i.e., up to 1,000 atoms, the time spent to compute the van der Waals energy is short, even using the van der Waals with the celllist technique. However, this time will be significant if the hundreds of thousands of evaluations of the GA are considered.

A simple way to parallelize and reduce time is to take advantage of the GA evaluation to perform as many evaluations as possible at once. In each GA generation, all new individuals could be evaluated at the same time mapping the evaluation procedure for the available processors (Figure 6-(a)). Such a parallelization of a GA is a common technique present in many Evolutionary Algorithms and requires no knowledge of how the evaluation function works [Alba et al. 2002]. However, computational time can be wasted if the evaluation function is not optimized. Such an approach should ideally be used after the optimization of the evaluation function, which is not our focus here. Before we can apply such a parallelization technique, we need to investigate it at the lower level, i.e., using fine-grain parallelization. Figure 6 illustrates two different ways to speed up a GA for PSP. First, as [Alba et al. 2002] proposed[2] is the evaluation of individuals in parallel. The second, (Figure 6-(b)) as used in this paper, is to evaluate a single solution using parallel techniques, enabling its use for other metaheuristics, as Differential Evolution [Storn and Price 1997] and Estimation of Distribution Algorithms [Larranaga and Lozano 2002].

The finest-grain way to parallelize such a calculation using Lennard-Jones potential is not a good way to parallelize as well. The Lennard-Jones potential has few calculations that could be performed in parallel. However, the overhead produced in the creation of a new

[1] It works for any GA, not only GAs for PSP.

thread will be more significant than the Lennard-Jones computation itself. We decided to split the computation of traversing the cells into several processors so that Lennard-Jones can be performed in a parallel way with load balancing. Given the whole cell grid of size $N_x \times N_y \times N_z$, a simple and efficient way to distribute the task is to create N_x tasks. Therefore, each task will have a $1 \times N_y \times N_z$ size.

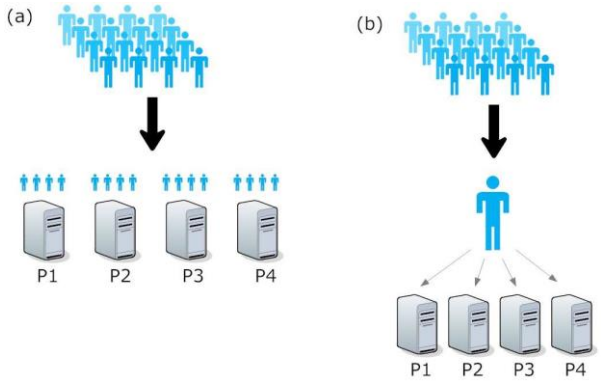

Figure 6. Different ways to speedup a GA: (a) coarse-grain, in which all individuals are evaluated at the same time on available processors; (b) fine-grain, one individual is evaluated using all processors available. That is, an individual is partioned into the number of available processors. The partial energy is computed at each processor and summed to compose the energy for an individual.

3.2.1. Parallel cell-list with OpenMP

As GAs require the evaluation of different solutions at each generation, all-atom coordinates for each individual evaluation must be known. For the OpenMP paradigm, in which all atoms are in the same memory as the processor that will compute the van de Waals energy, the communication is insignificant.

We use the *pragma* directive before the first loop as shown in Figure 4 and make all indices of loops as private. E_{vdw} is the only variable characterized as shared and receives the sum reduction operation. The default number of threads used is equal to the number of available cores [Bonetti et al. 2013].

3.2.2. Parallel cell-list with MPI

The MPI paradigm is more difficult to deal with since it has several overheads associated with the communication. If we parallelized the van der Waals calculation (without being applied to GAs), it would not be necessary to send the atom coordinates to the processors; atom coordinates could be read from the disk, and the calculation can be performed. However, GAs produce new configurations, changing the values of the atom positions. For the MPI implementation used in GA, we considered the time spent to send the

atom coordinates to the processors. This overhead is not present in OpenMP since new protein configurations generated by GA are visible for all processors.

Two different methods of communication were developed: (i) constructing celllist only on master and sending parameters N_x,N_y,N_z,L_x,L_y,L_z and the reference atom (the atom of lowest value) to slaves (Figure 7-(a)) and (ii) sending all atoms to all slaves and performing the entire cell-list construction process for all slaves (Figure 7-(b)). The latter method can waste computation time since it repeats the same process of cell-list construction for all slaves. On the other hand, method (i) does not require the replication of cell-list construction, but slave processors must be idle while the master is constructing the cell-list and determining which atoms must be sent to which slave. Only when the slaves have received all parameters and their specific atoms, are they ready to perform the calculation. The advantage of method (i) is that it requires less communication than method (ii). Indeed, method (ii) produces a little more communication. However, such time is shorter than the idle time required in method (i) [Bonetti et al. 2010b].

Therefore, we decided to use the method (ii) in this paper. Each slave will perform the whole process of CA. The difference is that each processor traverses its specific cells and composes a sub-total of van der Waals energy. At the end of the traversal, the processors send the sub-total energy to the master so that the total van der Waals energy of the molecule (see Figure 7-(b)) can be composed.

We took advantage of MPI to create our data structure to store the atom coordinates using *MPI Datatype*. This data structure can store the *x*,*y*,*z* of the atom coordinates. Only one communication is necessary to send atom coordinates to the processors, which decreases the overhead. The communication method used is the *MPI Bcast*.

We used a round robin schedule mapping of tasks for the processors to keep the load balancing throughout the processors, i.e., processor i will compute tasks $t_i = \{i, i + n, i < N_x\}$.

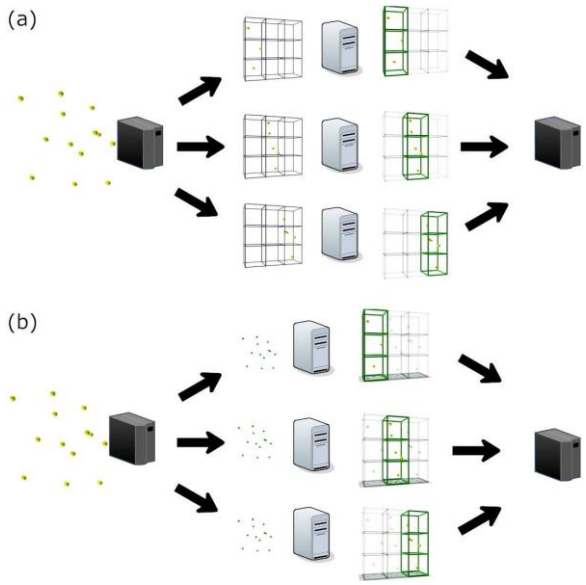

Figure 7. Two communication type for the MPI version of CA: (a) the master processor reads the atom, compute the cell grid and send to slaves the cell grid configurations and the atoms that each slave will use; (b) the master send all atoms coordinates to slaves. Each slave executes the CA algorithm, but will only traversal through the specific cells of the cell grid.

3.2.3. Combining OpenMP and MPI (Hybrid parallel cell-list)

OpenMP and MPI can be combined to create a hybrid paradigm. For each process created in a node using MPI, the process is partitioned into threads corresponding to the number of available cores of the processor. This way of mapping task to processors avoids several repeated messages since the round robin mapping of MPI sends atoms to repeated nodes more than once. In fact, the MPI version has one communication per task. On the other hand, for the hybrid-paradigm, the MPI portion performs only one communication for each node, since the threads are consequently mapped through OpenMP.

4 EXPERIMENTAL ANALYSIS

We used the cluster stored in the LCR[2] to evaluate the runtime of the proposed approaches. The cluster has 14 nodes and is divided into two groups according to its characteristics. The first group has 10 nodes with AMD Dual-Core 64 bits 2.8 GHz processors and 4 GB of RAM. The second group has 4 nodes with Intel Core i7 64 bits 2.67 GHz processors and 12 GB of RAM. The operation system is GNU/Linux Ubuntu with kernel 2.6.26-2. All nodes have two network adapters: one for the file system and another for messages in MPI, both connected to two independent 3Com Gigabit Ethernet switches.

[2] Reconfigurable Computer Laboratory, group of research of Embedded and Evolutionary and Systems at Institute of Mathematics and Computer Sciences at Sao Paulo University, Brazil.

The inputs of the algorithms are based on proteins that differ in structure and size. Eight different sizes of proteins used were chosen from PDB. Protein 1A11 was selected to be the lower bound, with only 390 atoms. Protein 1HTO was selected to represent the upper bound of experiments, with 147,900 atoms. Other proteins were selected to cover the range of proteins to evaluate the scalability of the proposed techniques. Also, accuracy has not been considered in our experiments, but it remains unchanged.

4.0.1. Statistical analysis

Each experiment was repeated 100 times so that a correct sample could be obtained in the measurements of the time. A mean was computed from the sample for the production of the graphics. The times were measured using the same function as used in IOZone Filesystem Benchmark [IOzone 2009].

All results shown in the next Section were calculated using software R. The methods were statistically compared using the p-value of the Welch Two Sample t-test with 95% confidence interval. The comparisons made were: CA with Quadratic Algorithm (QA) in Equation 1; CPA OpenMP with 8 and 16 processors with CA; CPA MPI, processors ranging from 2 to 18 with CA; CPA hybrid, processors ranging from 8 to 32 with CA; CPA hybrid, processors ranging from 8 to 32 with CPA MPI; and finally, CPA hybrid, processors ranging from 8 to 16 with CPA OpenMP. All tests were performed for all 8 proteins, rendering in 176 tests. The highest p-value obtained was 0.002 and occurred between techniques CPA hybrid with CPA OpenMP, both with 8 processors. The detailed values can be seen in Table 1.

Here we present the results achieved for CA and CPA. First, we evaluated the van der Waals calculation using the CA. Next, we showed the results for CPA using the three different approaches. Lastly, we show a comparison between our proposed techniques and the traditional van der Waals calculation applied in GAs for PSP.

Table 1. P-value's of the Welch Two Sample t-test. The highest value is highlighted.

Procs.	Sample 1	Sample 2	1A11	1BFI	1AI0	4HHB	1W1L	1RUZ	2BGN	1HTO
1	CA	QA	1.9 E-114	0.0E+00	7.2 E-300	0.0E+00	1.0E-179	2.3E-019	3.3E-005	6.4 E-008
8	CPA	CA	2.8	0.0E+0	1.0	7.5E-	9.6E-	1.2E-	2.2E-	2.5

	OMP		E-068	00	E-222	149	252	246	111	E-172
16	CPA OMP	CA	4.1 E-006	2.8E-269	9.3 E-202	1.4E-216	2.9E-247	3.1E-190	1.7E-080	5.6 E-111
2	CPA MPI	CA	6.4 E-169	1.7E-157	4.5 E-075	1.0E-047	1.8E-062	2.1E-067	3.3E-129	1.7 E-175
4	CPA MPI	CA	1.1 E-072	5.7E-100	9.7 E-027	8.7E-077	1.8E-004	1.0E-013	3.7E-162	4.0 E-187
6	CPA MPI	CA	9.3 E-012	5.3E-104	2.5 E-089	8.0E-085	1.2E-069	5.1E-072	7.7E-276	9.3 E-216
8	CPA MPI	CA	1.9 E-006	7.4E-097	1.8 E-081	3.5E-015	1.7E-071	1.8E-080	1.4E-135	1.7 E-173
10	CPA MPI	CA	2.3 E-047	4.8E-160	4.4 E-099	1.7E-012	4.0E-007	2.7E-075	8.8E-091	1.4 E-117
12	CPA MPI	CA	7.2 E-063	3.6E-170	2.2 E-117	1.9E-004	2.4E-013	4.7E-088	1.4E-160	1.9 E-109
14	CPA MPI	CA	5.3 E-063	1.4E-167	1.8 E-101	1.2E-016	2.2E-006	1.3E-076	7.7E-089	2.8 E-087
16	CPA MPI	CA	2.1 E-062	1.8E-166	6.4 E-117	1.7E-019	2.4E-010	2.2E-073	6.5E-087	5.3 E-089
18	CPA MPI	CA	1.4 E-	1.4E-177	2.2 E-	1.0E-009	5.6E-005	5.9E-075	3.4E-084	5.5 E-

			065		004					083
8	CPA Hybrid	CA	4.5 E-019	0.0E+000	2.0 E-207	6.8E-295	0.0E+000	3.1E-183	4.2E-143	1.3 E-157
16	CPA Hybrid	CA	5.3 E-173	0.0E+000	7.6 E-300	9.8E-300	0.0E+000	0.0E+000	6.7E-233	1.9 E-198
24	CPA Hybrid	CA	1.1 E-128	4.6E-233	4.1 E-300	2.6E-248	0.0E+000	0.0E+000	1.0E-300	2.0 E-058
32	CPA Hybrid	CA	2.1 E-144	0.0E+000	5.0 E-302	1.6E-266	0.0E+000	0.0E+000	0.0E+000	3.2 E-261
8	CPA Hybrid	CPA MPI	1.1 E-015	2.5E-066	5.9 E-106	1.4E-038	9.7E-279	1.1E-100	2.5E-029	1.5 E-056
16	CPA Hybrid	CPA MPI	3.7 E-132	2.4E-039	3.2 E-145	2.2E-102	0.0E+000	3.6E-050	2.0E-061	5.3 E-053
24	CPA Hybrid	CPA MPI	7.3 E-118	1.2E-036	4.8 E-134	1.7E-062	6.3E-080	4.1E-053	1.1E-070	6.3 E-076
32	CPA Hybrid	CPA MPI	3.0 E-136	1.9E-014	2.6 E-144	3.8E-086	1.2E-087	4.1E-077	9.4E-082	6.6 E-087
8	CPA Hybrid	CPA OMP	2.0 E-003	2.8E-043	4.9 E-297	2.2E-052	1.3E-134	5.3E-132	5.2E-028	4.1 E-061
16	CPA Hybri	CPA	7.1 E-	3.1E-051	3.6 E-	1.3E-143	8.5E-204	7.4E-235	2.1E-042	1.1 E-

d	OMP	004		162					062

4.1. SPEEDUP OF CELL-LIST ALGORITHM

The base algorithm (or the reference algorithm) is the QA implementation of the van der Waals calculations as showed in Equation 1 using neither cell-list nor parallelization.

Figure 8 shows the speedup achieved using the proposed CA in comparison to QA. Points represent the experimental data, and the line represents the predicted linear model. Indeed, the CA reduced the complexity from $O(n^2)$ to $O(n)$. The speedup line predicted linear increases according to the size of the protein. Even for small proteins, the speedup is significant. For protein 1AI0 with 4,728 atoms, the speedup is 5. Larger proteins did produce speedups more impressive, as 1HTO, which resulted in a speedup of 127. The improvement relies on the size of the cell grid (which also depends on the number of atoms). The larger the number of atoms, the larger will be the cell grid. Table 2 shows the size of the proteins and the cell grid sizes. Based on the results, we can state that CA is the best sequential algorithm achieved for the van de Waals energy calculation.

For instance, the QA time of the van der Waals energy for the protein 1AI0 is 161 ms, while the same energy can be obtained using CA in 32 ms. That is, a GA could take up to 1 hour to predict such a protein. Thus, in the next Sections, we will show the results achieved for CPA.

Table 2. Proteins selected from PDB with the number of atoms and corresponding dimensions of the cell grids used.

Protein	**Num. of Atoms**	**Cell Grid Size** $(N_x \times N_y \times N_z)$
1A11	390	$3 \times 2 \times 3$
1BFI	1,753	$4 \times 5 \times 4$
1AI0	4,728	$6 \times 6 \times 5$
4HHB	8,804	$8 \times 6 \times 7$
1W1I	11,671	$9 \times 8 \times 8$
1RUZ	22,380	$13 \times 14 \times 14$
2BGN	69,448	$14 \times 14 \times 32$
1HTO	147,980	$27 \times 18 \times 16$

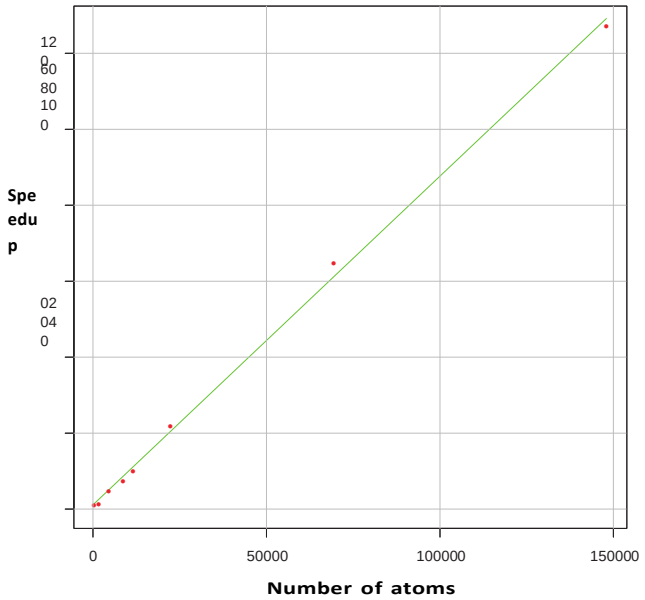

Figure 8. Speedup achieved with CA in relation to QA.

4.2. PARALLEL CELL-LIST WITH OPENMP

Although the simplest parallel implementation of the van der Waals uses OpenMP, it can produce good results. Figure 9 shows the speedup achieved for CPA using OpenMP about CA. The experiment was performed in a node containing an i7 processor with 4 physical cores. The speedup is close to 4, indicating that the tasks were properly distributed among processors and the computational time of the van der Waals calculation was proportionally reduced by the number of cores.

In biology, globular proteins are known to be more compact, i.e., that more atoms interact with each other [Berg et al. 2002]. The high number of atoms inside a single cell could be a small disadvantage of cell-list since it will have to compute more interactions, always performed in QA. Points four and five of Figure 9 show a depression in the speedup since these proteins are more globular than the others used. However, this speedup is still good.

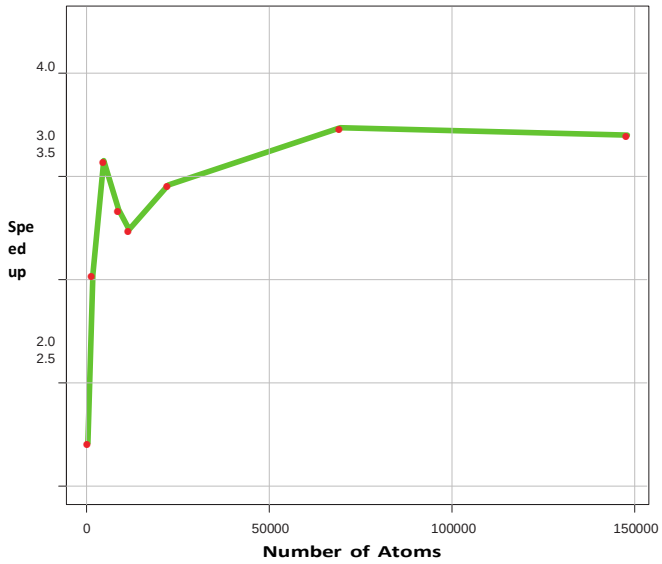

Figure 9. Speedup achieved with CPA with OpenMP in relation to CA.

4.3. PARALLEL CELL-LIST WITH MPI

Figure 10 shows the speedup achieved when the van der Waals energy was computed using the CPA with MPI. The experiments were performed in 9 nodes of AMD processors. For small proteins, such as 1BFI (1,753 atoms), the speedup was not significant, due to inter-process communication. The cell-list procedure is so fast that the communication time strongly influences the total computational time, and the parallelization for small proteins is not viable. On the other hand, for proteins above 4,728 atoms, the speedup is significant for a small number of processors.

However, even for large proteins, the tradeoff between communication and computation is not good. The speedup is low when MPI is used with more than 4 processors since the communication time increases as the protein size increases.

4.4. PARALLEL CELL-LIST WITH OPENMP AND MPI (HYBRID)

A expected good way to take advantage of both paradigms, OpenMP and MPI, is to use a hybrid paradigm, in which we can explore features of i7 processors with OpenMP. Besides, it can be used on several nodes with MPI. Figure 10 shows that for above 5 processors the efficiency is frozen. We ran the hybrid in 4 nodes of the cluster that contain the i7 processor, performing only four communication calls. After receiving the atoms by MPI, each node computes the specific region of the cell grid, splitting the tasks through OpenMP, i.e., each node could compute several tasks (8 in the i7 processor) using only one communication.

Brazilian Journal of Development

Figures 11, 12, 13 and 14 show the speedup achieved by the proposed methods (CPAs). For small proteins such as 1A11, the OpenMP paradigm is more adequate (Figure 11). That happens since the number of cores of one node is higher than the number of tasks.

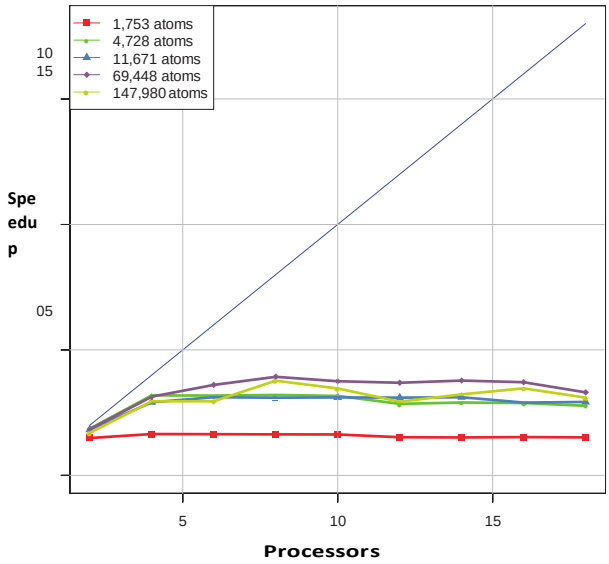

Figure 10. Speedup achieved for CPA with MPI in relation to CA.

For protein 1AI0 (Figure 12), the number of processors is significant when considering the hybrid and the OpenMP alone, since, for above 15 processors, the hybrid approach is fastest. In Figure 13, the hybrid is the fastest in all cases. The use of more processors than tasks will again produce the same speedup. Figure 14 shows the increase in the speedup for the hybrid approach. In all four cases, the use of MPI isolated is not a good approach. However, when combined with OpenMP, it produces better results.

Brazilian Journal of Development

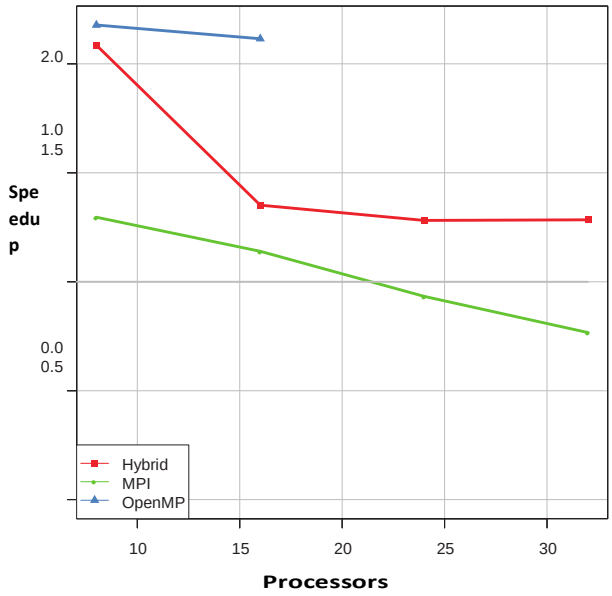

Figure 11. Speedup achieved with the three proposed algorithms for protein 1A11 (390 atoms).

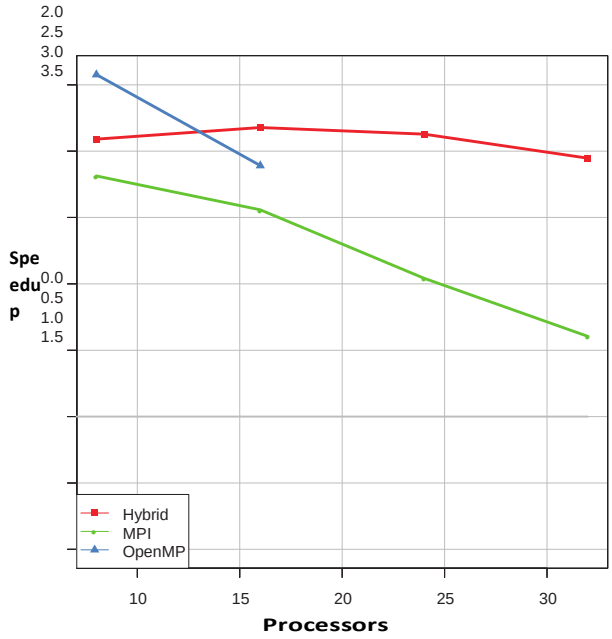

Figure 12. Speedup achieved with the three proposed algorithms for protein 1AI0 (4,728 atoms).

Brazilian Journal of Development

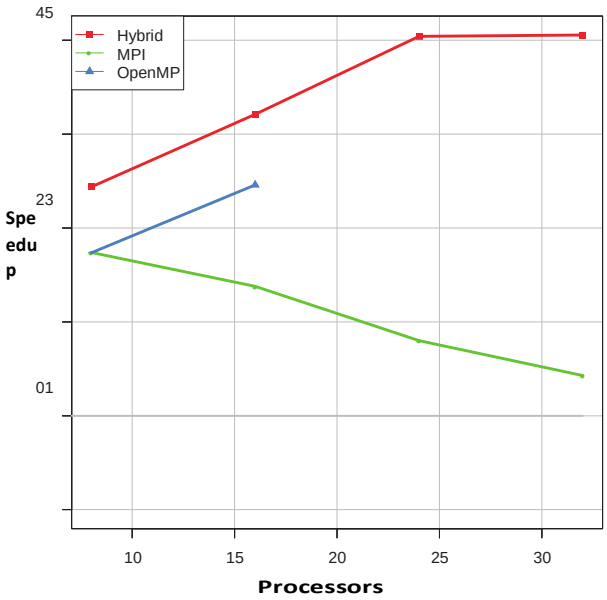

Figure 13. Speedup achieved with the three proposed algorithms for protein 1RUZ (22,380 atoms).

4.5. COMPARISON BETWEEN TECHNIQUES

Figure 15 shows a comparison between the techniques addressed in this paper, including the hybrid proposed. For very small proteins, the speedup of the sequential cell-list is good. MPI worked well for non-small proteins. OpenMP has shown the best speedup for proteins like 1A11. However, the hybrid paradigm is, indeed, the fastest algorithm for almost all classes of proteins.

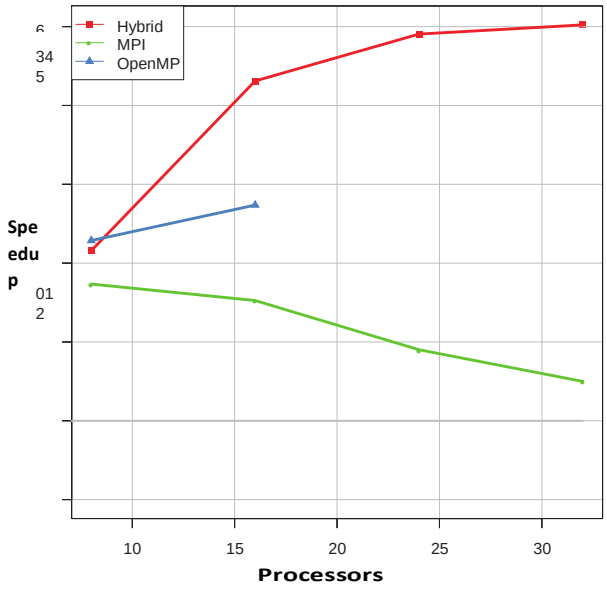

Figure 14. Speedup achieved with the three proposed algorithms for protein 2BGN (69,448 atoms).

In practice, OpenMP is the most efficient (considering the speedup and the number of processors) of all parallel implementations of cell-list, since it can reduce the runtime near to

the hybrid paradigm, i.e., the hybrid paradigm uses 24 processors over OpenMP to produce a slightly different speedup. Therefore, OpenMP is more suitable to be used in a GA for PSP.

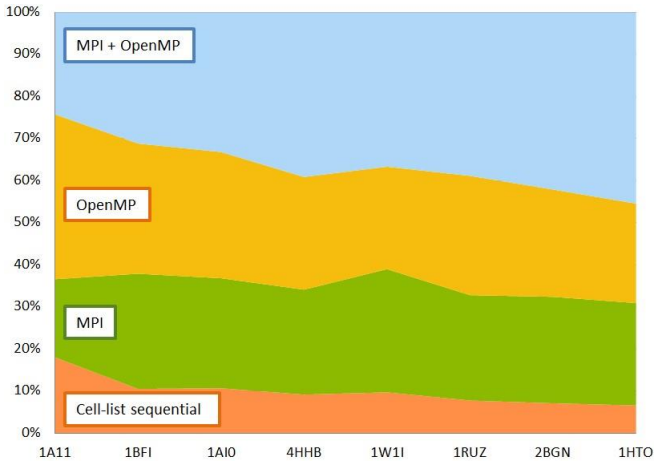

Figure 15. Comparison of performance of the proposed techniques to speedup the van der Waals.

4.6. PARALLEL CELL-LIST IN GA FOR PSP

This experiment with ProtPred was exclusively performed in one node of Sun Blade x6250 dual processor Intel Xeon E5440 at Laboratorio Nacional de Computac¸´ ao˜ Cient´ıfica (LNCC). To validate the approach in a real-world application, we converted the traditional ProtPred into an algorithm capable of evaluating the van der Waals energy using cell-list and OpenMP. The parallel algorithm did not affect the quality of prediction. We considered only the runtime of the GA with 250,000 evaluation functions calls, and we set the population size and number of generations to 500.

Figure 16 shows the results of the implementation. Indeed, the runtime of a GA for PSP can be reduced by the use of more efficient and parallel techniques. For protein 4HHB, the runtime of a GA was reduced from 33 hours to approximately 3.5 hours, i.e., a speedup of 9.5 and a reduction time of 90%.

Brazilian Journal of Development

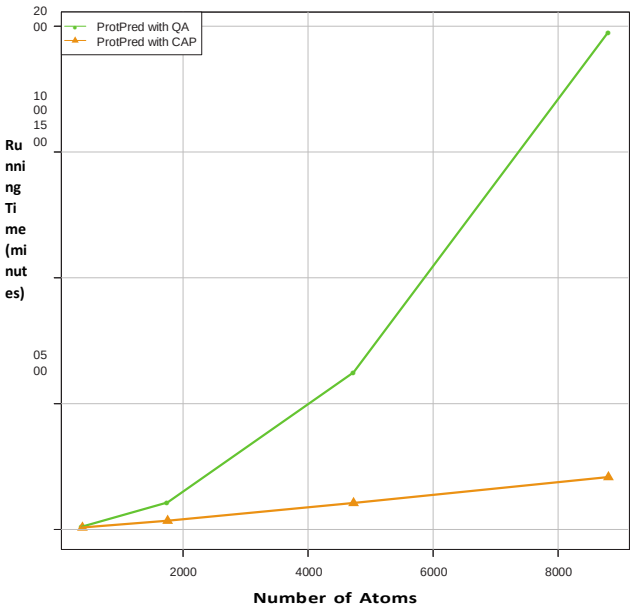

Figure 16. Comparison of performance of the CPA with OpenMP and the QA used in a real-world application.

5 CONCLUSIONS

The van der Waals energy used to evaluate the quality of proteins in GA can be efficiently computed using the cell-list technique, which evaluates the protein using linear complexity as accurately as the quadratic algorithm.

Moreover, flexible parallel techniques applied to cell-list can be used to reduce the run time of the GA. By flexible, we mean the use of distinct parallel programming paradigms in a same algorithm, which, together, can explorer diverse benefits available in parallel platforms.

This paper compares the results of three parallel paradigms of the celllist: OpenMP version developed in [Bonetti et al. 2013], MPI developed in [Bonetti et al. 2010b] and the new hybrid parallel implementation of cell-list using MPI and OpenMP concomitantly. All parallel versions reduced the running time of the van der Waals energy calculation, when compared to their sequential version. The hybrid version, however, shows significant speedups independently from the size of the protein.

The trade-off between communication and computation times of the MPI algorithm was not good and its speedup is limited. OpenMP implementation is more suitable for small proteins, since the communication time is very short when compared to networks in a cluster. Therefore, the hybrid paradigm, which uses MPI and OpenMP has enough flexibility to speedup both small and large sizes of proteins. Our results show that, for these experiments, the hybrid approach presented speedups, regarding the running time and maintain the same

accuracy of the results. However, for smaller proteins, as expected, OpenMP using only one node with four cores can achieve speedups near to the speedups of the hybrid solution, offering, in such cases, a better relation cost x benefit.

The validation algorithm used, i.e., the GA called ProtPred, received the parallel CA implementations for the van der Waals evaluation. Such implementations in ProtPred contributes for the prediction of non-small protein viability and reduces the running time for small proteins.

Despite its efficiency with the evaluation of the van der Waals energy, ProtPred can achieve good predictions only for small proteins, due to necessity of determine other energies as well. For non-small proteins, energies that are more refined must be added to the evaluation function by electrostatic, solvation and hydrogen bond energies. Such energies have the same structure as the van der Waals energy and must be suitable to receive the cell-list technique for the complexity reduction and the same parallel algorithms. The combination of all energies may produce a more accurate and efficient GA for PSP. Furthermore, the GPU-based implementations of cell-list can be useful for the achievement of better speedups.

ACKNOWLEDGMENTS

The authors would like to acknowledge FAPESP, CAPES and CNPq (Brazilian research foundations) for their financial support. The authors would like also to acknowledge LNCC (Laboratorio Nacional de Computação Científica, Brazil) for the kindly use of the cluster SunHPC in experiments with ProtPred.

REFERENCES

Alba, E., Nebro, A. J., and Troya, J. M. (2002). Heterogeneous computing and parallel genetic algorithms. *Journal of Parallel and Distributed Computing*, 62:1362–1385. Malaga 29071, Spain accepted January 29, 2002.´

Allen, M. P. and Tildesley, D. J. (1987). *Computer Simulation of Liquids*. Oxford University Press.

Anfinsen, C. B. (1972). Studies on the principles that govern the folding of protein chains. *Nobel Lecture*, pages 103–119.

Becker, O. M., Jr., A. D. M., Roux, B., and Watanabe, M. (2001). *Computational Biochemistry and Biophysics*. CRC.

Ben´ıtez, C. M. V. and Lopes, H. S. (2010). Protein structure prediction with the 3dhp side-chain model using a master–slave parallel genetic algorithm. *Journal of the Brazilian Computer Society*, 16(1):69–78.

Berg, J., Tymoczko, J., and Stryer, L. (2002). *Biochemistry 5th ed*. W. H. Freeman.

Bonetti, D. R., Delbem, A. C., Travieso, G., and Souza, P. S. L. (2013). Enhanced van der waals calculations in genetic algorithms for protein structure prediction. *Concurrency and Computation: Practice and Experience*, 25(15):2170–2186.

Bonetti, D. R. F. (2010). Aumento da eficiencia do calculo da energia de van der waals em algoritmos geneticos para predicao de estruturas de proteinas. Master's thesis, Institute of Mathematics Sciences and Computation - University of Sao
Paulo. http://www.teses.usp.br/teses/disponiveis/55/55134/ tde-20052010-110415/pt-br.php.

Bonetti, D. R. F., DELBEM, A. C. B., and ANDRADE, F. B. (2010a). Improving the efficiency of the van der waals energy function used in the genetic algorithms for protein structure prediction. Biomat 2010 10h International Symposium on Mathematical and Computational Biology.

Bonetti, D. R. F., Delbem, A. C. B., Travieso, G., and de Souza, P. S. L. (2010b). Optimizing van der waals calculi using cell-lists and mpi. In *Evolutionary Computation (CEC), 2010 IEEE Congress on*, pages 1 –7.

Brasil, C. R. S. and Delbem, A. C. B. (2011). Ivestigating relevants aspects moeas protein structure prediction. *GECCO 2011*, pages 1–8. Manuscript sumbitted for publication.

Brasil, C. R. S., Lima, T. W. D., Gabriel, P. H. R., and Delbem, A. C. B. (2008). Moprotpred: A multiobjective evolutionary algorithm to protein structure prediction with area accessibility. *8th International Symposium on Mathematical and Computational Biology. Proceedings of BIOMAT 2008*, pages 1–2. Campos do Jordao.

Case, D., Darden, T., Cheatham, T., III, Simmerling, C., Wang, J., Duke, R., Luo, R., Merz, K., Wang, B., Pearlman, D., Crowley, M., Brozell, S., Tsui, V., Gohlke, H., Mongan, J., Hornak, V., Cui, G., Beroza, P., Schafmeister, C., Caldwell, J., Ross, W., and Kollman, P. (2004). *AMBER 8*. http://ambermd.org/. University of California, San Francisco.

Das, S., Abraham, A., Chakraborty, U., and Konar, A. (2009). Differential evolution using a neighborhood-based mutation operator. *Evolutionary Computation, IEEE Transactions on*, 13(3):526–553.

de Lima Correa, L., Borguesan, B., Krause, M. J., and Dorn, M. (2018).ˆ Threedimensional protein structure prediction based on memetic algorithms. *Computers Operations Research*, 91:160 – 177.

Dorn, M., e Silva, M. B., Buriol, L. S., and Lamb, L. C. (2014). Three-dimensional protein structure prediction: Methods and computational strategies. *Computational Biology and Chemistry*, 53:251 – 276.

Friesner, R. A. (2002). *Computational Methods for Protein Folding: Advances in Chemical Physics*. Wiley.

Haile, J. (1992). *Molecular dynamics simulation: elementary methods*. John Wiley & Sons, Inc. New York, NY, USA.

Handl, J., Lovell, S. C., and Knowles, J. (2008). Investigations into the effect of multiobjectivization in protein structure prediction. In Rudolph, G., Jansen, T., Beume, N., Lucas, S., and Poloni, C., editors, *Parallel Problem Solving from Nature – PPSN X*, pages 702–711, Berlin, Heidelberg. Springer Berlin Heidelberg.

IOzone (2009). Iozone filesystem benchmark.

Jones, J. E. (1924). On the determination of molecular fields. ii. from the equation of state of a gas. *Proceedings of the Royal Society of London. Series A*, 106(738):463–477.

Larranaga, P. and Lozano, J. A. (2002). *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation (Genetic Algorithms and Evolutionary Computation)*. Kluwer Academic Publishers Group.

Nelson, D. L. and Cox, M. M. (2004). *Lehninger Principles Of Biochemistry Fourth Edition*. w. H. Freeman.

Sandia, C. (2003). *LAMMPS Users Manual*. http://lammps.sandia.gov/. Last access: December, 2010.

Spoel, D. V. D., Lindahl, E., Hess, B., van Buuren, A. R., Apol, E., Meulenhoff, P. J., Tieleman, D. P., Sijbers, A. L. T. M., Feenstra, K. A., van Drunen, R., and Berendsen, H. J. C. (2004). *Gromacs User Manual Version 3.2*. www.gromacs.org. 19-21.

Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341– 359. 10.1023/A:1008202821328.

Stote, R., Dejaegere, A., Kuznetsov, D., and Falquet, L. (1999). Tutorial charmm.

Webster, D. M. (2000). *Protein Structure Prediction: Methods and Protocols*. Humana Press.