Paper title:

Rigid-CLL: Avoiding Constant-distance Computations in Cell Linked-Lists Algorithms

Corresponding author:

V. Ruiz de Angulo. E-mail: ruiz@iri.upc.edu

Authors and affiliations:

V. Ruiz de Angulo[†], J. Cortés[‡], and J. M. Porta[†]

[†] Institut de Robòtica i Informàtica Industrial, CSIC-UPC
Llorens Artigas 4-6, 2a planta
08028 Barcelona, Spain
E-mails: {ruiz,porta}@iri.upc.edu
Fax: +34 93 401 57 50

[‡] Laboratoire d'Analyse et d'Architecture des Systèmes, CNRS
7, avenue du Colonel Roche
31077 Toulouse, France
E-mail: jcortes@laas.fr
Fax: +33 5 61 33 63 45

Keywords:

Non-bonded interactions, Steric contacts, Cell linked-lists, Monte Carlo simulations, Molecular Dynamics simulations, Mechanical models, Robotic approaches to molecular modeling.

Rigid-CLL: Avoiding Constant-distance Computations in Cell Linked-Lists Algorithms

V. Ruiz de Angulo[†], J. Cortés[‡,*], and J. M. Porta[†]

[†] Institut de Robòtica i Informàtica Industrial, UPC-CSIC

Llorens Artigas 4-6, 08028 Barcelona (Spain)

[‡] CNRS, LAAS, 7 avenue du colonel Roche, F-31077 Toulouse Cedex 4, France

[*] Université de Toulouse; UPS, INSA, INP, ISAE; UT1, UTM, LAAS; F-31077 Toulouse Cedex 4, France

Abstract

Many of the existing molecular simulation tools require the efficient identification of the set of non-bonded interacting atoms. This is necessary, for instance, to compute the energy values or the steric contacts between atoms. Cell linked-lists can be used to determine the pairs of atoms closer than a given cutoff distance in asymptotically optimal time. Despite this long-term optimality, many spurious distances are anyway computed with this method. Therefore, several improvements have been proposed, most of them aiming to refine the volume of influence for each atom. Here, we suggest a different improvement strategy based on avoiding to fill cells with those atoms that are always at a constant distance of a given atom. This technique is particularly effective when large groups of the particles in the simulation behave as rigid bodies as it is the case in simplified models considering only few of the degrees of freedom of the molecule. In these cases, the proposed technique can reduce the number of distance computations by more than one order of magnitude, as compared with the standard cell linked-list technique. The benefits of this technique are obtained without incurring in additional computation costs, because it carries out the same operations as the standard cell linked-list algorithm, although in a different order. Since the focus of the technique is the order of the operations, it might be combined with existing improvements based on bounding the volume of influence for each atom.

Keywords: Non-bonded interactions, Steric contacts, Cell linked-lists, Monte Carlo simulations, Molecular Dynamics simulations, Mechanical models, Robotic approaches to molecular modeling.

1 Introduction

Molecular simulations tools such as Molecular Dynamics and Monte Carlo algorithms [11] or recent methods originating from Robotics [2, 6, 7, 15, 27, 32] rely on the efficient determination of interactions between non-bonded atoms. This is fundamental, for instance, to speed up energy computations, or to filter high-energy conformations by detecting steric contacts. In practice, only atoms closer than a given cutoff distance are considered when computing the non-bonded interactions. However, unless specialized hardware is used, the determination of the close-enough pairs of atoms can take from 64% to 89% of the simulation time depending on the cutoff used [9].

The Cell Linked-List (CLL) algorithm [17] is the method of choice to determine pairs of close atoms since it outperforms alternative methods such as Verlet lists [10]. In this algorithm, the simulation box is divided into cubic cells with a side equal to the cutoff distance, d. In a first phase, each particle is inserted in a cell taking into account its coordinates. In a second phase, the particles in a given cell are queried for interaction with particles in the same or in adjacent cells. Since the assignment of the particles to cells scales with the number of particles, n, and the number of neighboring cells is bounded, the overall cost of the algorithm is linear with n, which is asymptotically optimal. However, the constant factors in this algorithm play a central role to obtain an efficient implementation. Actually, the 27 cells defining the neighborhood of a given atom are just a rough approximation of the ball of radius d centered at this atom. Less than a 16% of the volume covered by the cells is actually inside the ball and this implies that a large number of spurious interactions are considered, with the consequent decrease in efficiency. Many variants of the CLL algorithm have been proposed to improve its performance. One possible approximation is to use smaller cells [23, 26]. The problems with this approach are the large memory requirements and the increase of the number of neighboring cells to be checked for each atom, which can lead to computational inefficiencies, particularly due to the high percentage of empty cells. The efficiency can also be increased by sorting the data to detect too far away particles [14] or to reduce the cache misses [24, 33]. In these cases, though, any gain obtained comes at the extra cost of the sorting process. The CLL algorithm can also be sped up by parallelizing the nearest-neighbor detection [16, 22, 26, 30], sometimes resorting to special hardware which can be either not generally available [9] or too specialized, avoiding the use of optimal CLL algorithms [12]. Finally, another approach to reduce the cost of the CLL algorithm is to group closely connected particles [25]. This yields a CLL approach at the level of clusters of particles, so that many of the distance computations can be resolved by operations between a small number of cluster centers. However, this is at the expense of increasing the size of the cells and, thus, of increasing the number of tests for particles in different clusters.

In this paper, we also target systems of clustered particles, but for the particular case where clusters form rigid groups, and without resorting to larger cells. Rigid clusters of particles appear, for instance, in simulations operating at the level of molecules, and in general in environments where some distances or angles are constrained to a fixed value during the whole simulation. Interesting examples of this are Monte Carlo and Molecular Dynamics simulations operating in internal coordinates [29] considering the rigid geometry assumption [13]. Larger rigid clusters appear when running rigid-body molecular dynamics simulations [8, 18], or when considering simplified models tailored to simulate large-scale conformational transitions in biological macromolecules [7, 20]. Moreover, there is a growing interest in Monte Carlo algorithms applied to protein structure prediction and protein design that, at every step, replace some protein segments with one selected from a database [4, 31]. In this case, a rigid subset is replaced by another and, thus, large portions of the molecule are considered as rigid bodies. In all these applications, the terms of the potential energy involving pairs of atoms in the same rigid group are constant for all conformations and, therefore, conformations can be compared considering only the energy terms from interacting atoms in different rigid groups, which can be a very small fraction of the number of non-bonded interactions. The same applies to steric contacts that can only appear between atoms in different rigid groups. The idea exploited in this paper arises from the observation that the usual independent insert and query phases of the CLL algorithm can be actually interleaved. In this way, distances between atoms in the same rigid cluster are avoided simply because those non-interacting particles are not yet included in the grid of cells when the query is performed.

The following section presents the proposed variant of CLL, that we call Rigid-CLL. To facilitate its understanding, we describe intermediate stages leading to it from the basic CLL algorithm. Next, the performance of the method is analyzed and compared to existing CLL algorithms.

Algorithm 1: The CLL algorithm.

```
   Input: A set of atoms, $M$, a grid, $G$, and a cutoff distance, $d$.
   Output: The list, $P$, of atom pairs closer than $d$.
   // Insert phase
1  foreach $a \in M$ do
2      $c \leftarrow$ Cell$(a, G)$
3      Insert$(a, c)$
   // Query phase
4  $P \leftarrow \emptyset$
5  foreach $c \in G$ do
6      foreach $c' \in$ Neighbors$(c, G)$ do
7          foreach $a \in c$ do
8              foreach $a' \in c'$ do
9                  if $a < a'$ and $\|a - a'\| \leq d$ then
10                     $P \leftarrow P \cup \{(a, a')\}$
11 Return$(P)$
```

2 The Rigid-CLL Algorithm

Algorithm 1 describes the CLL algorithm in its more common form. This algorithm takes as input a set of atoms, M, possibly containing more than one molecule, a grid, G, discretizing the environment where the atoms move, and a cutoff distance, d, used to determine which atoms interact. Herein, the cells forming the grid are assumed to be cubes of side d. In this way, only atoms in 27 cells have to be checked when determining the interactions for the atoms in each cell. As shown in Algorithm 1, the CLL algorithm has two phases: a insert phase (lines 1-3) and a query phase (lines 4-10). In the insert phase, each atom in M is assigned to a cell in the grid according to its coordinates. Function Cell (line 2) computes the mapping from coordinates to cells and function Insert associates the atom to the corresponding cell. In the query phase, all the pairs of neighboring cells are scanned and the distance between atoms in those cells is checked. Function Neighbors(c, G) returns the list of 27 cells neighbors of cell c, possibly considering periodicity conditions. Atoms closer than the cutoff distance d (line 9) are added to the output list (line 10). Note that atoms can be sorted taking into account its coordinates or an index associated with them and that the condition $a < a'$ (line 9) is added to avoid computing the same distance twice. Several minor variations of this algorithm exist such as, for instance, avoiding repeated distance computations at the level of cells (i.e., using $c \leq c'$), but the overall cost of the algorithm, remain almost the same. In an ideal algorithm, all computed distances between a pair of atoms will lead to a new item in the output list (line 10). In the plain CLL algorithm, though, in average less than a 16% of the considered pairs of atoms are actually closer than the given cut off distance. Nevertheless, the whole algorithm is asymptotically optimal since it scales linearly with n, the number of atoms in M. This is more evident in Algorithm 2 where some of the loops of the query phase are re-ordered without changing the overall behavior of the algorithm. In this form it is clear that since the number of neighboring cells for each atom and the number of atom per cell are bounded, the algorithm scales with n. Since this cost is asymptotically optimal, in general, no other algorithm can perform better by more than a constant factor. Our aim is to increase this constant factor not for the general case, but

Algorithm 2: The CLL algorithm with loop reordering.

Input: A set of atoms, M, a grid, G, and a cutoff distance, d.
Output: The list, P, of atom pairs closer than d.

```
   // Insert phase
1  foreach $a \in M$ do
2      $c \leftarrow$ CELL($a, G$)
3      INSERT($a, c$)
   // Query phase
4  $P \leftarrow \emptyset$
5  foreach $a \in M$ do
6      $c \leftarrow$ CELL($a, G$)
7      foreach $c' \in$ NEIGHBORS($c, G$) do
8          foreach $a' \in c'$ do
9              if $a < a'$ and $\|a - a'\| \leq d$ then
10                 $P \leftarrow P \cup \{(a, a')\}$
11 RETURN($P$)
```

Algorithm 3: The CLL algorithm merging the Insert and Query phases.

Input: A set of atoms, M, a grid, G, and a cutoff distance, d.
Output: The list, P, of atom pairs closer than d.

```
   // Combined Insert-Query
1  $P \leftarrow \emptyset$
2  foreach $a \in M$ do
       // Query phase for atom $a$
3      $c \leftarrow$ CELL($a, G$)
4      foreach $c' \in$ NEIGHBORS($c, G$) do
5          foreach $a' \in c'$ do
6              if $\|a - a'\| \leq d$ then
7                  $P \leftarrow P \cup \{(a, a')\}$
       // Insert phase for atom $a$
8      INSERT($a, c$)
9  RETURN($P$)
```

Algorithm 4: The Rigid-CLL algorithm.

```
Input: A set of atoms, $M$, a grid, $G$, and a cutoff distance, $d$.
Output: The list, $P$, of atom pairs in different rigid groups closer than $d$.
// Combined Insert-Query
1  $P \leftarrow \emptyset$
2  foreach $r \in$ RigidGroups($M$) do
       // Query phase for rigid group $r$
3      foreach $a \in r$ do
4          $c \leftarrow$ Cell($a, G$)
5          foreach $c' \in$ Neighbors($c, G$) do
6              foreach $a' \in c'$ do
7                  if $\|a - a'\| \leq d$ then
8                      $P \leftarrow P \cup \{(a, a')\}$
       // Insert phase for rigid group $r$
9      foreach $a \in r$ do
10         $c \leftarrow$ Cell($a, G$)
11         Insert($a, c$)
12 Return($P$)
```

for the case where the simulation involves rigid groups of atoms. Note that a trivial approach to deal with the presence of rigid groups in the molecule would be to check whether atoms a and a' are in the same group before actually computing the distance between them in line 9 of Algorithm 2. This saves computation of distances, but it does not change the overall structure of the algorithm and, thus, it does not reduce the cost of looping over all pairs of atoms, even if they are in different rigid groups.

The loop rearrangement taking from Algorithm 1 to Algorithm 2 can be further applied merging the loops from the insert and the query phases, leading to Algorithm 3. Note that in this algorithm the insert and query operation for a given atom are inverted to avoid checking the possible interaction of an atom with itself. This algorithm is equivalent to the original CLL algorithm, but simpler since only preceding atoms a, with $a < a'$, are in the grid when a given atom a' enters in the query phase. This makes unnecessary the test $a < a'$ and, produces shorter (half on average) lists of atoms in each cell, i.e., it reduces the number of iterations over the loop at line 5 of the algorithm. However, this is only a marginal gain in efficiency since no distance computation is saved.

The Rigid-CLL algorithm proposed in this paper saves distance computations modifying the structure of Algorithm 3 to process atoms by rigid groups (see Algorithm 4). When a rigid group, r, enters the query phase (lines 3 to 8) the interactions of the atoms in r with those in the preceding rigid groups are examined. Afterward, all atoms in r are inserted in the grid (lines 9 to 11). Thus, the interactions among atoms within the same rigid group are not considered, independently of the distance separating them. As a consequence, the output of Rigid-CLL only includes pairs of atoms in different clusters. As mentioned, this output is enough to compute the energy offset that will enable comparison between different conformations.

In Algorithm 4, like in Algorithm 3, atoms causing repeated interactions are not present in the lists of the most internal loop in the new algorithm (line 6). However, in this case, those lists are particularly short since, at the query point for a given atom, a, atoms belonging to the rigid group including a, which include most of its neighbors, are not yet in the grid. Consequently, a large number of distance computations are saved with respect to the original CLL algorithm.

As a final optimization, it is clear that the query phase of the atoms of the first rigid group is useless because no atom has still been introduced in the grid. The same can be said from the insert phase of the atoms in last rigid group, since it will not be a subsequent query phase. Since the order of treatment of the rigid groups is arbitrary, we can select as first and last groups the two largest ones. In this way, although no distance computation is saved, we can skip the more expensive query and the second more expensive insert phases.

	Number of atoms	Number of rigid groups	Number of DOFs
POU	1399	170	169
Hemoglobin A	4336	4	24
Hemoglobin A (flex. side chains)	4336	892	915

Table 1: The three systems used in the experiments with the corresponding number of atoms, rigid groups, and degrees of freedom.

3 Experiments

The Rigid-CLL algorithm has been implemented in C++ and integrated in the Psf-Amc molecular modeling framework [1]. In this framework, we evaluated the advantaged of using the rigid-aware algorithm with respect to two existing algorithms. The first one is the standard CLL algorithm implemented as in Algorithm 2. The second algorithm used in the comparison is a recent improvement of the CLL algorithm [14] where particles are sorted along the axis connecting cells in order to avoid the computation of distances between particles that are too far away along that axis. This improved CLL algorithm will be called S-CLL herein. The algorithms have been tested first on three systems of different sizes and considering a variable number of degrees of freedom (DOFs), as summarized in Table 1. The first system is the POU domain of N-Oct-3 transcription factor. The structure of the molecule involves two distinct sub-domains, termed POUs and POUh, connected by a flexible linker [21]. Therefore, in our model, these two sub-domains were considered as rigid atom groups. Flexibility was allocated to the 36-residue linker, which involves 169 DOFs (108 in the backbone, corresponding to 36 consecutive residues, 61 in the side chains). The two other systems correspond to hemoglobin A [5], which is a tetrameric protein. We considered two instances, one where each monomer is a rigid cluster, and the other with rigid backbones, but flexible side-chains. These models involve 24 and 915 DOFs, respectively. All experiments have been carried out by sampling the neighborhood of the native conformations of molecules with small movements and with two types of settings, corresponding to the two main types of applications of CLL-like algorithms. In the first set of tests the algorithms were applied to perform steric clash detection, typically used in molecular simulations to filter out high-energy configurations. To detect steric clashes, the side of the cells is set to the diameter of the largest atom in the molecule and the test $\|a - a'\| \leq d$ is replaced by $\|a - a'\| \leq R + R'$ with R and R' the radii of atoms a and a', respectively. The second set of experiments, were done with a constant cutoff distance $d = 10$ Å, which is typically used for energy evaluation of protein conformations within molecular dynamics or Monte Carlo simulations.

Table 2 shows the number of distance computations carried out by each algorithm. This

		R-CLL	S-CLL	CLL	R-CLL/S-CLL	R-CLL/CLL
Steric	POU	2811	4554	15885	0.62	0.17
	Hemoglobin A	2418	15807	60248	0.15	0.04
	Hemoglobin A (flex. side chains)	33289	15804	60211	2.11	0.55
$d = 10$ Å	POU	56391	92821	218004	0.61	0.25
	Hemoglobin A	441248	436940	1468147	1.01	0.3
	Hemoglobin A (flex. side chains)	1143634	436938	1468323	2.62	0.78

Table 2: Number of computed distances and relative ratios in Rigid-CLL, S-CLL, and the standard CLL for steric clash detection and with a cutoff $d = 10$ Å.

		R-CLL	S-CLL	CLL	R-CLL/S-CLL	R-CLL/CLL
Steric	POU	0.23	1.3	0.79	0.18	0.29
	Hemoglobin A	0.83	3.76	2.65	0.31	0.40
	Hemoglobin A (flex. side chains)	1.53	3.79	2.82	0.40	0.54
$d = 10$ Å	POU	0.44	1.6	2.50	0.27	0.18
	Hemoglobin A	3.29	6.52	16.58	0.51	0.20
	Hemoglobin A (flex. side chains)	8.71	6.51	16.84	1.34	0.52

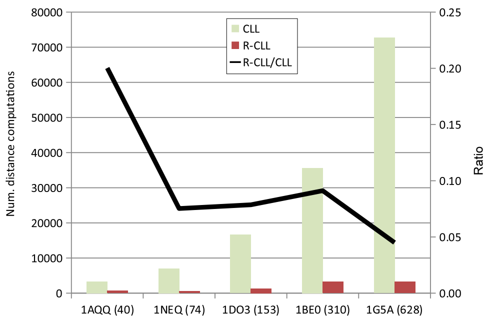
Table 3: Execution time (in ms) and relative ratios in Rigid-CLL, S-CLL, and the standard CLL for steric clash detection and with a cutoff $d = 10$ Å.

measure is independent of the actual implementation of the algorithms and is typically used in previous works to show their potential efficiency. For instance, the strategy of partitioning the domain using cells of edge length to half the cutoff [23, 26] computes a 58% of the number of distances computed by the plain CLL, and S-CLL reduces this ratio to a 26%. As shown in the table, as far as the size of the rigid clusters is large, the Rigid-CLL algorithm performs remarkably better than the standard CLL algorithm reducing the number of distances actually computed to a ratio similar or even lower than those of previous works. Rigid-CLL is also better than the S-CLL in several cases, specially on those where the rigid clusters are large. Note that, under the rigid geometry assumption, rigid groups containing several atoms naturally arise even when the molecule is considered fully flexible. For instance in the case of the hemoglobin with flexible side chains, the average number of atoms per rigid group is less than 5. In such an extreme case, however, the advantage of using the Rigid-CLL algorithm is clearly minor. A possibility to deal with these situations would be to combine Rigid-CLL with existing approaches such as S-CLL. Since these algorithms are based on a different principle, their combination appears to be feasible.

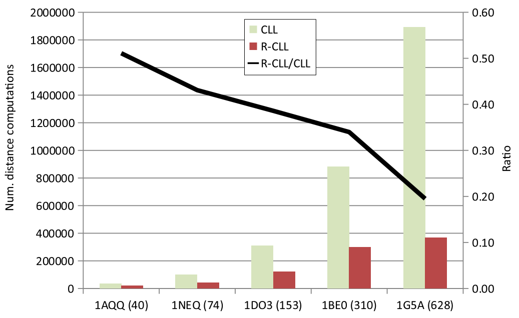
However, the advantage of S-CLL in terms of computed distances must be counterbalanced with the additional cost of projecting and sorting the particles in neighboring cells. In our approach nothing is paid for the distance computations saved. As a consequence, and as shown in Table 3, the actual execution times of both the standard CLL and S-CLL are higher than the ones for the Rigid-CLL, except for the extreme case of the hemoglobin with flexible side chains with $d = 10$ Å, where S-CLL is faster. These computational times correspond to the execution on a Intel Xeon at 2.66 Ghz and they do not only account for the cost of distance

	R-CLL	CLL & S-CLL	R-CLL/CLL
POU	641	4108	0.16
Hemoglobin A	23	14460	0.002
Hemoglobin A (flex. side chains)	4165	14462	0.29

Table 4: Size of the output list with Rigid-CLL and with the standard CLL with cutoff $d = 10$ Å.



(a)



(b)

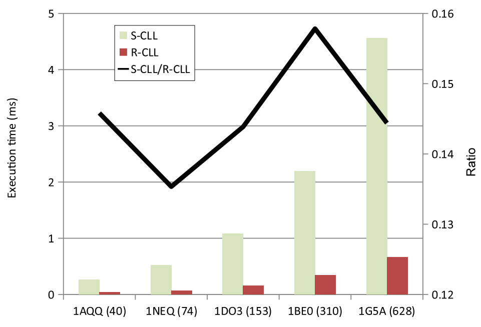
Figure 1: Comparison between Rigid-CLL and the standard CLL algorithm with increasingly large proteins in terms of computed distances. (a) Testing for steric clashes. (b) Using a cutoff $d = 10$ Å.

computations, but also for other aspects such as the memory access patterns that, according to [10], can form a significant bottleneck for particle-based simulations. Therefore, these results validate the advantage of Rigid-CLL in practical situations.

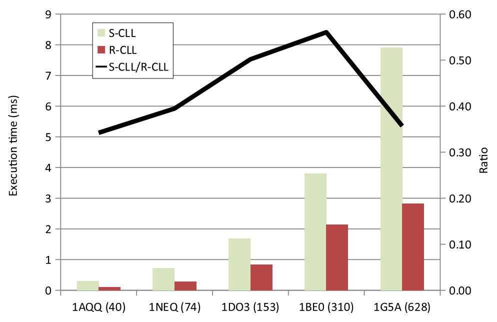
The same set of tests was carried out to evaluate the trivial modification of CLL that checks membership to a same rigid. In this case, the computing time ratios with respect to standard CLL are at best 0.75 and 0.65 for steric clash and cutoff $d = 10$ Å, respectively. These ratios are notoriously larger than the ratios obtained with Rigid-CLL. Moreover, this trivial modification is slower than CLL for a highly articulated model such as Hemoglobin with flexible side chains, because in this example the majority of the checks added to the algorithm are useless. In contrast, by design, Rigid-CLL never entails an extra cost with respect to CLL.

Note that Rigid-CLL returns a set of interacting atoms that is different and always smaller than that of non rigid-aware algorithms, as shown in Table 4. In general, the smaller the output set the faster the subsequent processes such as, for instance, the computation of the energy to compare different conformations. The reduction of interactions can be dramatic when the system consist only in large rigid groups, as it is the case of hemoglobin A.

Figure 1 shows how the number of distance computations in Rigid-CLL scales with the size of the molecules while keeping fixed the number of DOFs. We tested the algorithm with a set of increasingly large proteins. The number of amino acid residues and PDB identifier are: 40 (1AQQ), 74(INEQ), 153 (1DO3), 310 (1BE0), and 628 (1G5A). In all cases the protein was modeled with 3 DOFs in the backbone corresponding to the ω, ϕ and ψ bond torsions of the middle residue. The results show that overall the advantage of the Rigid-CLL algorithm increases with the size of the protein.

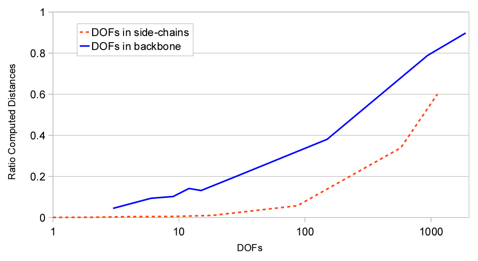


(a)

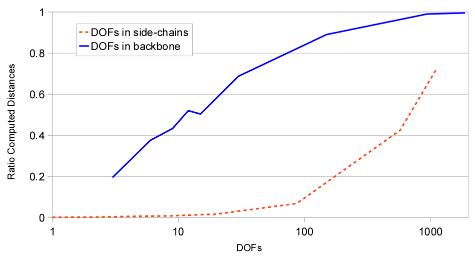


(b)

Figure 2: Comparison between Rigid-CLL and S-CLL with increasingly large proteins in terms of execution time in milliseconds. (a) Testing for steric clashes. (b) Using a cutoff $d = 10$ Å.



(a)



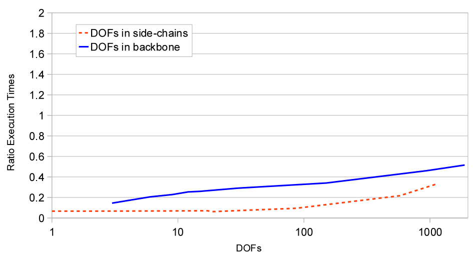
(b)

Figure 3: Ratios of computed distances between Rigid-CLL and the standard CLL algorithm with increasing number of DOFs either in the backbone or in side-chains. (a) Testing for steric clashes. (b) Using a cutoff $d = 10$ Å.

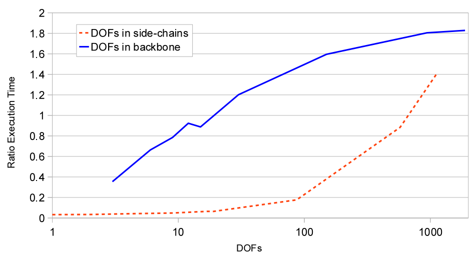
Since the comparison between Rigid-CLL and S-CLL can not be only based on the number of distance computations, those algorithms are compared in terms of the actual execution time to take into account the overall cost of all the algorithms in real conditions. The result shown in Fig. 2 indicate that Rigid-CLL is at least twice as fast as S-CLL in all tested cases, being significantly faster, in particular for steric clash tests.

Figure 3 shows the influence of the number of DOFs in the performance of the Rigid-CLL algorithm as compared with the standard CLL, while keeping the molecule size. We tested the algorithm for the 1G5A protein with an increasing number of DOFs uniformly distributed either in the backbone or in the side chains. The ratio of computed distances is below one indicating that the Rigid-CLL algorithm is advantageous with respect to the standard CLL, specially while the number of DOFs is limited, i.e. when there is a small number of rigid atom groups. For example, when 50 side-chains are flexible (the model involves 88 rigid clusters), the number of computed distances is reduced by almost two orders of magnitude, for both the steric clash test and when using a cutoff distance of 10 Å.

Finally Fig. 4 show the ratio of execution times between Rigid-CLL and S-CLL also when increasing the number of DOFs in the 1G5A protein. The results show that Rigid-CLL is again



(a)



(b)

Figure 4: Ratios of execution times in milliseconds between Rigid-CLL and S-CLL with increasing number of DOFs either in the backbone or in side-chains. (a) Testing for steric clashes. (b) Using a cutoff $d = 10$ Å.

faster except when using a large number of DOFs and a large cutoff. As mentioned, it would be possible to combine Rigid-CLL with a method reducing the field of influence for each atom, such as S-CLL or using smaller cells [23, 26].

4 Conclusions

This paper introduced Rigid-CLL, a method particularly suited to detect range-limited interactions between sets of particles including rigid clusters. Rigid-CLL can be used in any kind of molecular simulation tool, even when the DOF and the rigid clusters change at each iteration [28]. Moreover, the proposed algorithm is compatible with any previously proposed improvements for the CLL algorithm, such as reducing the cell size, or ordering the distances among neighboring atoms. Although applicable in general, the method is particularly attractive when the molecules are modeled with a moderate number of DOFs or when using a small cutoff distance, such as the one required in steric clash detection. This is a typically setup for robotics-based algorithms to compute the motions of simplified protein models.

An open point not discussed in this paper is how to determine rigid clusters from protein models. This is a relevant problem that has been addressed for instance using rigidity theory approaches [19], or by the analysis of conformational ensembles produced by molecular dynamics simulations [3]. Such methods could be used in combination with our algorithm. Note, however, that further discussion on this point is out of the scope of this paper.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Science and Innovation under project DPI2010-18449.

References

[1] Psf-Amc web page. http://softs.laas.fr/Psf-Amc.

[2] M. S. Apaydin, D. L. Brutlag, C. Guestrin, D. Hsu, J. C. Latombe, and C. Varma. Stochastic roadmap simulation: an efficient representation and algorithm for analyzing molecular motion. *Journal of Computational Biology*, 10(3-4):257–281, 2003.

[3] S. Bernhard and F. Noé. Optimal identification of semi-rigid domains in macromolecules from molecular dynamics simulation. *PLoS One*, 5(5):e10491, 2010.

[4] R. Bonneau, C. E. Strauss, C. A. Rohl, D. Chivian, P. Bradley, L. Malmström, T. Robertson, and D. Baker. De novo prediction of three-dimensional structures for major protein families. *Journal of Molecular Biology*, 1(322):65–78, 2002.

[5] N. L. Chan, P. H. Rogers, and A. Arnone. Crystal structure of the S-nitroso form of liganded human hemoglobin. *Biochemistry*, 37(47):16459–16464, 1998.

[6] J. Cortés, T. Siméon, V. Ruiz, D. Guieysse, M. Remaud, and V. Tran. A path planning approach for computing large-amplitude motions of flexible molecules. *Bioinformatics*, 21:i116–i125, 2005.

[7] J. Cortés, D.T. Le, R. Iehl, and T. Siméon. Simulating ligand-induced conformational changes in proteins using a mechanical disassembly method. *Physical Chemistry Chemical Physics*, 12:8268–8276, 2010.

[8] A. Dullweber, B. Leimkuhler, and R. McLachlan. Symplectic splitting methods for rigid body molecular dynamics. *Journal of Chemical Physics*, 107(5):5840–5851, 1997.

[9] D. E. Shaw *et al.* Millisecond-scale molecular dynamics simulations on Anton. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, pages 1–11, 2009.

[10] E. S. Fomin. Consideration of data load time on modern processors for the Verlet table and linked-cell algorithms. *Journal of Computational Chemistry*, 32(7):1386–1399, 2011.

[11] D. Frenkel and B. Smit. *Understanding Molecular Simulations: From Algorihtms to Applications.* Academic Press, 2002.

[12] M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande. Accelerating molecular dynamic simulation on graphics processing units. *Journal of Computational Chemistry*, 30(6):864–872, 2009.

[13] K. D. Gibson and H. A. Scheraga. Energy minimization of rigid-geometry polypeptides with exactly closed disulfide loops. *Journal of Computational Chemistry*, 18(3):403–415, 1997.

[14] P. Gonnet. A simple algorithm to accelerate the computation of non-bonded interactions in cell-based molecular dynamics simulations. *Journal of Computational Chemistry*, 28(2):570–573, 2007.

[15] N. Haspel, M. Moll, M. L. Baker, W. Chiu, and L. E. Kavraki. Tracing conformational changes in proteins. *BMC Structural Biology*, 10(Suppl. 1):S1, 2010.

[16] T. H. Heinz and P. H. Hünenberger. A fast pairlist-construction algorithm for molecular simulations under periodic boundary conditions. *Journal of Computational Chemistry*, 25(12):1474–1486, 2004.

[17] R. W. Hockney and J. W. Eastwood. *Computer simulation using particles.* Taylor & Francis, Inc., Bristol, PA, USA, 1988.

[18] M. Ikeguchi. Partial rigid-body dynamics in NPT, NPAT and NPgammaT ensembles for proteins and membranes. *Journal of Computational Chemistry*, 25(4):529–541, 2004.

[19] D. J. Jacobs, A. J. Rader, M. F. Thorpe, and L. A. Kuhn. Protein flexibilty predictions using graph theory. *Proteins: Structure, Function, and Bioinformatics*, 44:150–165, 2001.

[20] M. K. Kim, R. L. Jernigan, and G. S. Chirikjian. Rigid-cluster models of conformational transitions in macromolecular machines and assemblies. *Biophysical Journal*, 89:43–55, 2005.

[21] D. S. Latchman. POU family transcription factors in the nervous system. *Journal of Cellular Physiology*, 179:126–133, 1999.

[22] G. B. MacPherson and J. M. Reesea. Molecular dynamics in arbitrary geometries: Parallel evaluation of pair forces. *Molecular Simulation*, 34(1):97–115, 2008.

[23] W. Mattson and B. M. Rice. Near-neighbor calculations using a modified cell-linked list method. *Computer Physics Communications*, 119(2-3):135–148, 1999.

[24] S. Meloni, M. Rosati, and L. Colombo. Efficient particle labeling in atomistic simulations. *The Journal of Chemical Physics*, 126(12):121102, 2007.

[25] R. J. Petrella, I. Andricioaei, Brooks B. R, and M. Karplus. An improved method for nonbonded list generation: Rapid determination of near-neighbor pairs. *Journal of Computational Chemistry*, 24(2):222–231, 2003.

[26] M. Pütz and A. Kolb. Optimization techniques for parallel molecular dynamics using domain decomposition. *Computer Physics Communications*, 113(2-3):145–167, 1998.

[27] B. Raveh, A. Enosh, O. Schueler-Furman, and D. Halperin. Rapid sampling of molecular motions with prior information constraints. *PLoS Comput. Biol.*, 5(2):e1000295, 2009.

[28] V. Ruiz de Angulo. Monte Carlo simulations with few DOF's in large molecules. Technical report, Institut de Robòtica i Informàtica Industrial, 2011.

[29] C. D. Schwieters and G. M. Clore. Internal coordinates for molecular dynamics and minimization in structure determination and refinement. *Journal of Magnetic Resonance*, 152(2):288 – 302, 2001.

[30] D. Shaw. A fast, scalable method for the parallel evaluation of distance-limited pairwise particle interactions. *Journal of Computational Chemistry*, 26(16):1803–1803, 2005.

[31] A. Shehu and B. Olson. Guiding the search for native-like protein conformations with an ab-initio tree-based exploration. *International Journal of Robotics Research*, 29(8):1106–1127, 2010.

[32] X. Tang, B. Kirkpatrick, S. Thomas, G. Song, and N. M. Amato. Using motion planning to study RNA folding kinetics. *Journal of Computational Biology*, 12:862–881, 2005.

[33] Z. Yao, J.-S. Wang, G.-R. Liu, and M. Cheng. Improved neighbor list algorithm in molecular simulations using cell decomposition and data sorting method. *Computer Physics Communications*, 16(1-2):27–35, 2004.