Spatial Clustering of Molecular Dynamic Trajectories in Protein Unfolding Simulations

Pedro Gabriel Ferreira[1], Cândida G. Silva[2,3], Paulo J. Azevedo[4], and Rui M. M. Brito[2,3]

[1] Genome Bioinformatics Laboratory, Center for Genomic Regulation, Barcelona, Spain
pedro.ferreira@crg.es
[2] Chemistry Department, Faculty of Science and Technology, University of Coimbra, Coimbra, Portugal
[3] Center for Neuroscience and Cell Biology, University of Coimbra, Coimbra, Portugal
[4] Department of Informatics, CCTC, University of Minho, Braga, Portugal

Abstract. Molecular dynamics simulations is a valuable tool to study protein unfolding *in silico*. Analyzing the relative spatial position of the residues during the simulation may indicate which residues are essential in determining the protein structure. We present a method, inspired by a popular data mining technique called Frequent Itemset Mining, that clusters sets of amino acid residues with a synchronized trajectory during the unfolding process. The proposed approach has several advantages over traditional hierarchical clustering.

1 Introduction

Protein folding is the process by which the protein acquires its three-dimensional (3D) native structure. The 3D structure of a protein, ultimately determined by its linear sequence of amino acids, is essential for protein function. Recently, several human and animal pathologies, such as cystic fibrosis, Alzheimer's and mad cow disease, among others, have been identified as protein folding or unfolding disorders. Over the years, many experimental and computational approaches have been used to study these processes. The analysis of data obtained from molecular dynamics (MD) simulations of induced-unfolding processes may be an important tool to explore and understand the protein unfolding mechanisms.

Analyzing the behavior and the relationship among the residues during the simulation may provide important insights on the unfolding process. In particular, identifying groups of residues that show a synchronized behavior during the simulation can provide important clues on which residues play a critical role on protein folding and protein structure. Such residues may be involved in the formation of a core or nucleus that is preserved within the partially folded structures. In this work, by *synchronization* or *synchrony* between two or more

residues we mean that they conserve their relative distance during the simulation. Traditional clustering techniques are not flexible enough to allow additional restrictions nor that an element (residue) may appear in more than one cluster. Therefore, they are not adequate to capture the behavior of residues that appear synchronized with more than one set of other residues.

Here, we propose a method to cluster sets of amino acid residues (AARs), that have a synchronized trajectory during the unfolding process of the protein Transthyretin (TTR) [2, 3, 7]. Since the clustering is determined by the spatial location of the alpha-carbon (C_α) atoms of each residue, we call it *Trajectory Spatial Clustering*. The method is inspired in a popular data mining technique called Frequent Itemset Mining (FIM) [1, 5]. The method is devised in three main steps. First, all the pairs of AARs that during the simulation have a small distance variation are determined. These pairs will form the seed for larger clusters. In the second step, these clusters are successively extended with other AARs. If the elements of a cluster exceed a certain distance variation threshold (provided by the user), it means that the cluster is no longer synchronized and the extension process is stopped. Clusters that are contained in other clusters are considered redundant. In the last step, after all clusters have been determined, redundant cluster can be rejected and similar clusters merged.

2 Method

We start this section by formalizing the problem and then presenting our approach by describing the proposed algorithm. The goal of this work is to design a methodology that discovers sets of residues – clusters – that have a synchronized trajectory during the simulation. The residues in a cluster follow their own trajectory but conserve among each other their relative distance in 3D space. In order to measure the distance variation between two residues, we introduce a measure called *coefficient of distance variation*, denoted as *cdv*. Two AARs are considered to form a synchronized cluster if its *cdv* does not exceed a user defined threshold value, called cdv_{max}. An expected outcome is that this value increases with the cluster extension. To deal with this, we introduce a second parameter, cdv_{inc}, representing an increment in the cdv_{max} threshold for each new extension.

Residues that are contiguous in the linear sequence are expected to have a highly conserved movement due to physical restrictions. Thus, to avoid reporting such trivial relations between residues, a third parameter called *minimum sequence distance* (*msd*) is introduced. Imposing such restriction means that for residues to be considered in the same cluster, their distance in the linear sequence should be greater or equal to *msd*. The problem can be stated as follows:

Given the 3D coordinates of the C_α atom of each residue at each instant of the simulation, a maximum coefficient of distances variation (cdv_{max}), a cdv_{inc} increment for each new extension and a minimum sequence distance (msd) be-

tween residues, find all the clusters of residues that show a synchronized behavior.

The overall proposed approach can be summarized in the following three-step algorithm:

Step I - Rigid Link Determination Find all pairs of residues with distance variation below a pre-defined threshold (cdv_{max}) and a sequence distance greater than msd. These pairs will form the seeds for the clusters.

Step II - Cluster Extension In an agglomerative way, each cluster is successively extended with new residues. A residue is considered for extension if it forms a rigid link with one of the residues in the cluster. Extension stops when the cluster exceeds a given variation threshold ($cdv_{max} + cdv_{inc}$).

Step III - Cluster Filtering This step is optional and consists in removing redundant clusters and merging similar ones. Clusters are then ranked according to their cdv value.

2.1 Rigid Link Determination

In the first step of the method, we find all the pairs of residues that in the 3D space have a coefficient of distance variation less than the initial cdv_{max} and that in the linear sequence are apart at least msd residues. These pairs of residues are called *rigid links* and correspond to clusters of size 2. The distance between the residues A and B, at a given instant i of the simulation, is given by the Euclidean distance between their C_α atoms, expressed by Equation 1.

$$dist_i(A,B) = \sqrt{(A_i.x - B_i.x)^2 + (A_i.y - B_i.y)^2 + (A_i.z - B_i.z)^2} \quad (1)$$

For a simulation of N time points, the cdv of two residues A and B correspond to the root mean square deviation of the Euclidean distance between A and B and is given by Equation 2.

$$cdv(A,B) = \sqrt{\frac{\sum_{i=1}^{N} dist_i(A,B)}{N}} \quad (2)$$

By performing a pairwise comparison of all the AARs the rigid links are found.

2.2 Cluster Extension

In step two, the initial set of clusters are extended with new residues. Transitivity is the basic idea behind the extension process and it can be described as follows: if a and b form a rigid link denoted as $rl(a,b)$, and b and c another rigid link $rl(b,c)$, then $\{a,b,c\}$ will potentially form a cluster with synchronized behavior.

However, for clusters with size greater than two, a way to measure the overall cluster synchrony is needed. Several approaches can be taken. One such approach

would be to measure the variation of the cluster centroid. For each time frame the cluster centroid (central point) is calculated. Then, it is verified if during the simulation period the centroid positions vary significantly. This can be measured through the root mean square deviation of the Euclidean distance between two consecutive time points. If this variation is below a pre-defined threshold, the cluster is considered synchronized. We observed that this approach is incorrect since increasing the cluster size will always result in a decrease of centroid variation. Therefore, this measure is not able to capture correctly the cluster variation. A different approach to measure the overall cluster variation was considered. The global variation of a cluster C is given by the average value of cdv between all pairs of residues in cluster C. Algorithm 1 provides the way to calculate the global variation of the cluster. With this calculation each cluster can now be extended given that its global variation remains below a certain threshold.

input : *cluster* C (*cluster to extend*); msd
for $i = 1$ **to** $|C| - 1$ **do**
  for $j = i + 1$ **to** $|C|$ **do**
    `/* Compare pairwise all AARs in C */`
    $p = C[i]$;
    $q = C[j]$;
    $dSum = 0$;
    if $abs(p - q) \geq mindist$ **then**
      $dist = CDV(p, q)$;
      $dSum = dSum + dist$;
      $pairs = pairs + 1$;
    end
  end
end
return $gvar = dSum/pairs$;

Algorithm 1: GVar: Calculation of global variation of a cluster C.

The cluster extension step relies on a bottom-up procedure, where larger clusters result from the extension of smaller ones. Given a synchronyzed cluster $C = \{c_1, \cdots, c_n\}$, $C \cup \{x\}$ is a candidate cluster if $\exists\, rl(c_n, x)$. This strategy, which is based in one of the most efficient strategies for FIM [4], allows to find all valid candidate sets while maintaining a reduced number of redundant candidates. This approach relies on the monotonic property, also called downward closure, largely used by itemset mining algorithms [5] to discover sets of items that co-occur frequently. Frequent itemsets appear in the database a number of times greater than a given threshold value called minimum support. From this property results that any subset of a frequent itemset must also be frequent. Analogously, we have that all sub-clusters of a synchronized cluster are also synchronized.

Extending the cluster with new residues will increase its variation. At a certain point the initial variation threshold will become too restrictive to allow

for new extensions. Thus, the method has to account for a certain variation tolerance to each new extended cluster. This is expressed through the parameter cdv_{inc}. For each new extension of the cluster the value of cdv_{max} is increased by cdv_{inc}. From this results that a cluster is always tested for synchrony with the threshold relative to its size. Algorithm 2 describes the extension procedure (DFSExtend): the cluster C is extended with all rigid links (line 1) where the first AAR of the rigid link is equal to the last AAR in C (line 3). C is extended with the second AAR in rl if the global variation of the new cluster is below the allowed threshold (line 6). In this case $newC$ is considered a synchronized cluster. If $newC$ cannot be further extended (verified by an outcome of 0 in DFSExtend - line 7) then it is added to the list of synchronized clusters $LSynC$. This test saves work in step III since it already eliminates some of the redundant clusters. If the option to report all synchronized clusters is activated, then all new clusters that pass the test in line 6 are added to list $LSynC$.

```
   input : cluster C; Lrl(list rigid links); cdv_max; cdv_inc
 1 foreach rl in Lrl do
       /* AAR in rigid link rl                                          */
 2     a ↔ b = rl;
 3     if pop(C) = a then
           /* extend C with b                                          */
 4         newC = push(C, b);
 5         newCvar = GVar(newC);
 6         if newCvar ≤ (cdv_max + cdv_inc) then
               /* extend recursively new cluster                       */
 7             ex = DFSExtend(newC, Lrl, cdv_max + cdv_inc, cdv_inc);
 8             if ex == 0 then
 9                 push(LSynC, newC);
10             end
11         else
12             return 0;
13         end
14     end
15 end
```

Algorithm 2: DFSExtend: Recursive function for cluster extension.

2.3 Cluster Filtering

Performing cluster search through the combination of an exhaustive enumeration and residue at a time extension results that some of the reported clusters are found to be sub-clusters of larger clusters, which leads to the notion of maximal cluster. A cluster is said to be *maximal* if it is not contained in any other cluster. Therefore non-maximal clusters can be rejected (not reported) since they provide redundant information. Cluster extension is performed through a depth

first search. Reporting only the larger clusters of a branch of the search space tree eliminates many of the redundant clusters. For instance, in the following extension path $\{a, b\} \rightarrow \{a, b, c\} \rightarrow \{a, b, c, d\}$ only $\{a, b, c, d\}$ needs to be reported. Since some of the non-maximal clusters can be found at the end of a path, an additional test to verify cluster maximality is required. This consists in verifying if a synchronized cluster is contained or contains other synchronized cluster. Since highly synchronized redudant clusters can also be of interest, the user may choose the option of reporting all synchronized clusters. Additionally, clusters with similar composition can also be reported. For instance, $\{R10, R21, R50\}$ and $\{R10, R22, R50\}$ can be found as synchronized clusters. To make easier the interpretation of the results, a post-processing step can be applied to merge such clusters. The previous clusters can be merged as $\{R10, R(21-22), R50\}$.

3 Application

In this section, we make use of the proposed algorithm to assist in the study of the unfolding mechanisms of the protein Transthyretin (TTR) (Figure 1), a human plasma protein involved in amyloid diseases such as Familial Amyloid Polyneuropathy, Senil Systemic Amyloidosis and Familial Amyloid Cardiomyopathy [2].

Molecular dynamics protein unfolding simulations may be analyzed through the variation of chemical or geometric properties [2, 3, 7]. We focused on a particular dataset which consists of 127 time series, each representing the 3D coordinates of the C_α atom of each AAR along one MD unfolding simulation. Each time series is a collection of 8000 data frames, with the 3D coordinates of the C_α atom of each AAR per picosecond (ps) of simulation, for a total of 8 nanoseconds (ns) of simulation.

A prototype in C++ that implements the algorithm described in section 2 was developed. Different combinations of values of the parameters cdv_{max}, cdv_{inc} and msd were tested with computation times around 2 seconds. Here, we show and analyze the results obtained with the following parameter values: $cdv_{max} = 8$; $cdv_{inc} = 0.6$ and $msd = 2$. 148 rigid links are found and 30 synchronized clusters are reported. With $cdv_{inc} = 0$, only 15 clusters are found; with $cdv_{max} = 20$, $cdv_{inc} = 0.0$ and $msd = 2$, we obtain 3775 rigid links and 41277 clusters. Table 1 presents the 30 clusters ordered by the average cdv value. Note that these clusters were reported before step III and can be further tested for maximality and merged.

From the analysis of Table 1, three sets of AARs appear highly represented: 29 to 35, 67 to 73, 79 to 83 and 92, 94 and 96. In Figure 2, it is graphically represented the AARs preponderance on the reported synchronized clusters. Each point depicts the position of the C_α of each AAR in the native structure of the protein. Lines represent the rigid links in the cluster. From this figure we can see that only a small fraction of the AARs in the protein are covered by the clusters. When represented in the secondary structure of the protein (Figure 1), the highly represented AARs are placed in the following structures: β-strand B

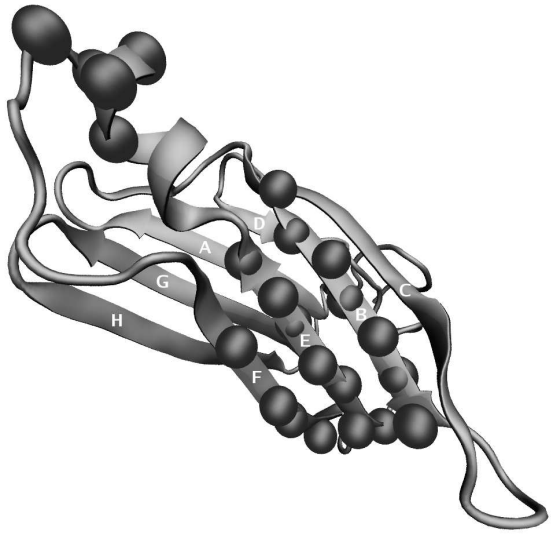

Fig. 1. Secondary structure ribbon representation of the WT-TTR monomer (PDB entry 1TTA). β-strands, α-helices, and turns and random structure are represented by arrows, coils and tubes, respectively. The eight β-strands are named A (residues 2-18), B (residues 28-36), C (residues 40-49), D (residues 54-55), E (residues 67-73), F (residues 91-82), G (residues 104-112) and H (residues 115-123). The residues identified in Table 1 are represented as beads.

Cluster	Avg *cdv*	Cluster	Avg *cdv*	Cluster	Avg *cdv*
34 69 94	6.79	33 70 92	7.85	31 71 92	8.15
77 80 83	7.05	35 68 94	7.88	73 76 79	8.18
33 69 94	7.08	75 78 81	7.92	19 112 115	8.24
76 79 82	7.24	34 67 96	7.98	32 45 58	8.29
76 80 83	7.54	32 72 92	8.01	29 73 76	8.40
32 69 94	7.57	10 13 105	8.07	75 79 82	8.44
32 71 92	7.59	32 70 92	8.08	33 72 92	8.45
35 69 94	7.60	30 72 92	8.11	30 71 92	8.52
34 68 94	7.74	35 67 96	8.13	79 85 88	8.57
31 72 92	7.83	33 71 92	8.14	30 73 76 79	9.21

Table 1. Output of the spatial clustering algorithm sorted by increasing value of average value of *cdv* for the cluster. The settings used for this analysis were: cdv_{max}=8, $cdv_{inc} = 0.6$ and $msd = 2$.

(AARs 28 to 36), β-strand E (AARs 67 to 73), α-helix (AARs 75 to 82) and β-strand F (AARs 91 to 97). More interestingly, we observed that the clusters tend to relate AARs in β-strands B, E and F mixed together, while the residues from the α-helix seem to relate more with each other. Another fact worth noticing is that among the residues from β-strands B, E and F two large "super-clusters" can be identified: (i) AARs 32 to 35 with AARs 69 and 72, and (ii) AAR 30 to 33 with AAR 71 or AAR 72, and AAR 92. In the experimental work on TTR by Liu and colleagues [6], it is suggested that the disruption of the structure of β-strands B, C, E and F is a cooperative process. By analyzing the trajectory of the different AARs during an unfolding simulation and their relation with each other, our results seem to point in the same direction. Thus, the method proposed here seems to prove useful in the analysis and comprehension of MD unfolding simulation data.

We selected the merged cluster $\{(32, 33, 34, 35)\ 69\ 94\}$ to visualize the synchrony of the elements in a cluster during the simulation. The 3D coordinates of the AARs are plotted separately in Figure 3. We can clearly see that the six AARs in this cluster have an almost perfect synchronization during the entire simulation.

In order to contrast the method proposed with a more traditional approach, Figure 4 presents the dendrogram obtained from the hierarchical clustering of the AARs based on the *cdv*. To obtain these results we first built a similarity matrix based on the *cdv* values for all the pairs of AARs. Then, a hierarchical clustering algorithm using the "average" agglomerative method was applied. As

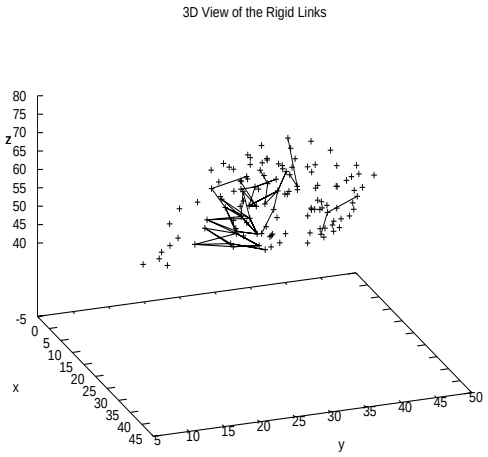

Fig. 2. Rigid Links between the c_α of the AARs of the clusters in Table 1 mapped into the native state of the protein.

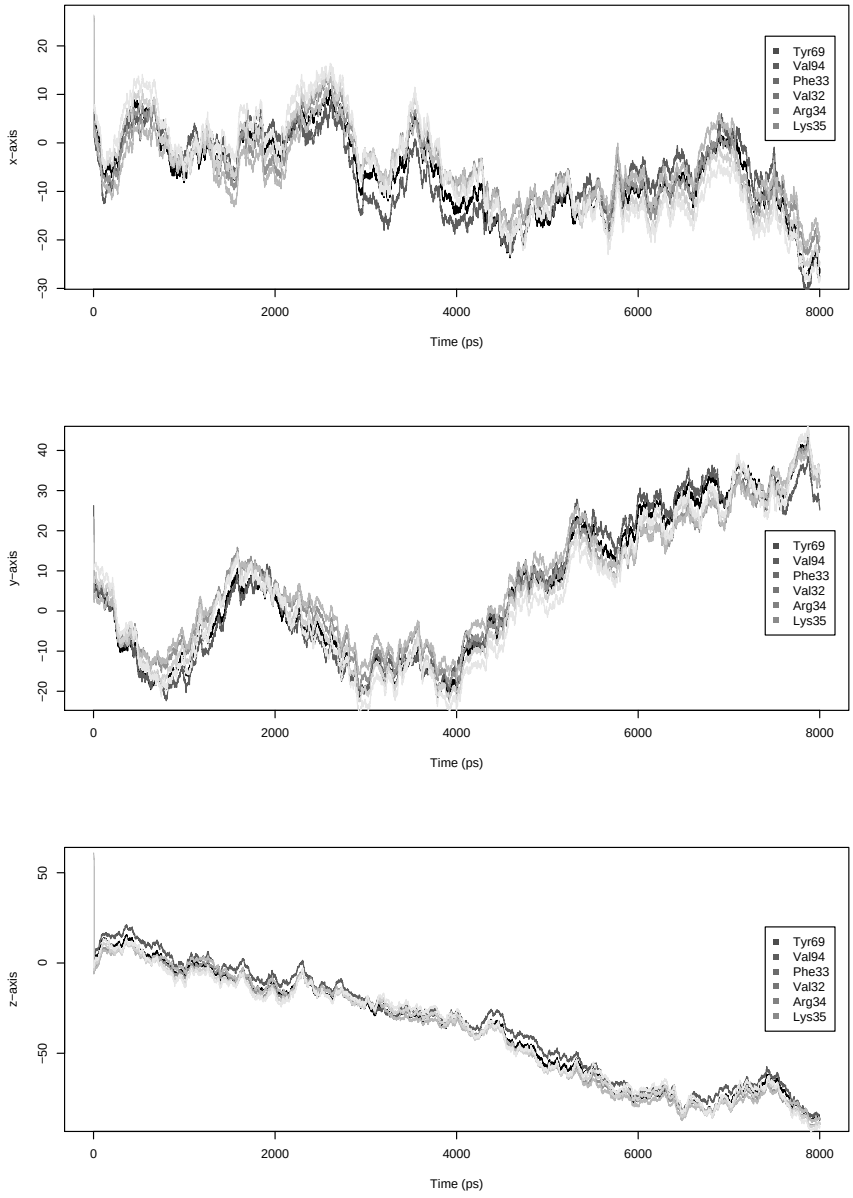

Fig. 3. 3D coordinates of the AARs in the cluster $\{(32, 33, 34, 35)\ 69\ 94\}$.

can be seen from Figure 4, the hierarchical clustering approach is mainly able to detect trivial relations, i.e relations between AARs that are contiguous in the linear sequence and therefore expected to have high synchrony. Nevertheless, a few surprising relations appear (for instance between residues 39 and 83) that may be worthwhile to look at in detail.

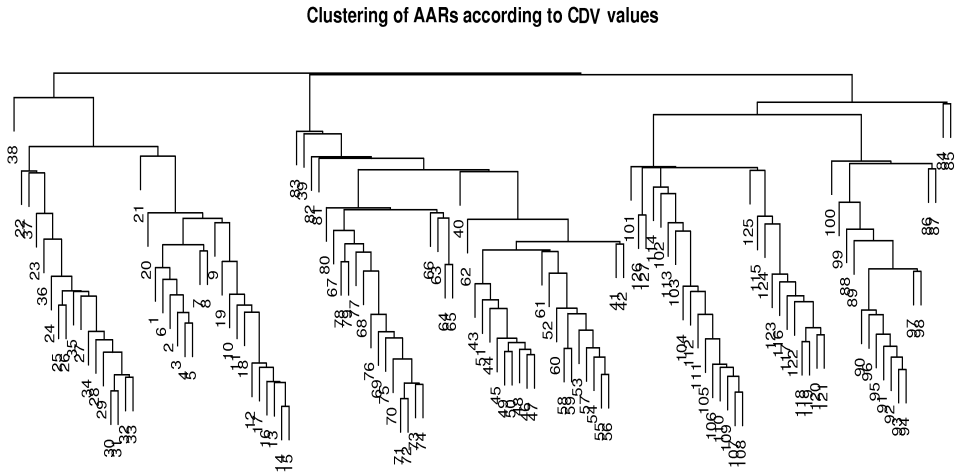

Fig. 4. Hierarchical Clustering of the AARs using the *cdv* as the similarity measure.

4 Discussion

In this paper, we propose a method for the analysis of data from MD simulations of the unfolding process of the protein Transthyretin (TTR). This method is intended to find groups of amino acid residues (AARs) that show a synchronized behavior during the entire simulation. Such AARs may be essential for the protein structure and will help to better understand the unfolding process. The applied strategy for the discovery of the synchronized clusters closely resembles the strategy applied by FIM algorithms, with the difference that the criterium to report the sets of items is not their frequency but the variation of their spatial location. Although it uses a simple approach, this method has revealed to be quite efficient regarding the characteristics of the target data. When compared to the hierarchical clustering approach, our method has the following advantages: (i) by imposing a minimum distance in the linear sequence for the AARs in the cluster, it avoids reporting trivial matches between contiguous (in linear sequence) AARs; (ii) it allows that an AAR may be present in more than one cluster and that they are located further apart in the linear sequence and in the 3D space; (iii) separating the AARs into clusters has the drawback of requiring

a threshold value, but the advantage of providing a more intuitive interpretation of the results and allowing the extension of the analysis to multiple simulations.

Regarding this last aspect, it is worth mentioning that, as in many other data mining applications, the definition of the correct parameters value is an iterative and interactive process. An apriori indication for the range of values where the most relevant solutions are expected to be found can be provided, but the exact values will always depend on the characteristics of the data.

If data from several simulations is available, this approach can be easily extended to find clusters that are conserved across several simulations. This can be done by adapting a frequent itemset mining algorithm [5] to discover itemsets that are conserved in different simulations.

Acknowledgments

The authors acknowledge the support of the "Fundação para a Ciência e Tecnologia", Portugal, and the program FEDER, through grant PTDC/BIA-PRO/72838/2006 (to PJA and RMMB) and the Fellowships SFRH/BPD/42003/2007 (to PGF) and SFRH/BD/16888/2004 (to CGS). We thank the Center for Computational Physics, Departamento de Física, Universidade de Coimbra, for the computer resources provided for the MD simulations.

References

1. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings of 20th International Conference Very Large Data Bases*, pages 487–499, 12–15 1994.
2. R.M.M. Brito, A.M. Damas, and M.J.S. Saraiva. Amyloid formation by transthyretin: from protein stability to protein aggregation. *Current Medicinal Chemistry - Immun. Endoc. and Metab Agents*, (3):349–360, 2003.
3. R.M.M. Brito, W. Dubitzky, and J.R. Rodrigues. Protein folding and unfolding simulations: A new challenge for data mining. *OMICS: A Journal of Integrative Biology*, (8):153–166, 2004.
4. B. Goethals and M. Zaki. Fimi'03: Workshop on frequent itemset mining implementations. In *Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, pages 1–13, 2003.
5. J. Han and M. Kambler. *Data Mining, Concepts and Techniques*. Morgan Kaufmann, second edition, 2006.
6. K. Liu, H.S. Cho, H.A. Lashuel, J.W. Kelly, and D.E. Wemmer. A glimpse of a possible amyloidogenic intermediate of transthyretin. *Nature Structural Bilogy*, 7(9):754–757, 2000.
7. J.R. Rodrigues and R.M.M. Brito. *How important is the role of compact denatured states on amyloid formation by Transthyretin?*, chapter Amyloid and Amyloidosis, pages 323–325. CRC Press, 2004.