RAIRO-Oper. Res. 50 (2016) 401–411
DOI: 10.1051/ro/2015043

**RAIRO Operations Research**
www.rairo-ro.org

# PETRI NETS FORMALISM FACILITATES ANALYSIS OF COMPLEX BIOMOLECULAR STRUCTURAL DATA

Anna Gogolinska[1,2], Rafal Jakubowski[1] and Wieslaw Nowak[1]

**Abstract.** Molecular dynamics (MD) simulation is a popular method of protein and nucleic acids research. Current MD output trajectories are huge files and therefore they are hard to analyze. Petri nets (PNs) is a mathematical modeling language that allows for concise, graphical representation of complex data. We have developed a few algorithms for PNs generation from such large MD trajectories. One of them, called the One Place One Conformation (OPOC) algorithm, is presented in a greater detail. In the OPOC algorithm one biomolecular conformation corresponds to one PN place and a transition occurring in PN graph is related to a change between biomolecules conformations. As case studies three simulations are analyzed: an enforced steered MD (SMD) dissociation of a transthyretin protein tetramer into dimers, the SMD dissociation of an antibody-antigen complex and a classical MD simulation of transthyretin. We show that PNs reproduce events hidden in MD trajectories and enable observations of the conformational space features hard-to-see by the other clustering methods. Thus, a fundamental process of biomolecular data classification may be optimized using the PN approach.

**Mathematics Subject Classification.** 90B10, 68W99, 92-08.

Received September 8, 2015. Accepted September 21, 2015.

## 1. Introduction

Molecular dynamics (MD) simulation is a well-established method [17] used to generate trajectories of atomic complexes, such as proteins and/or nucleic acids, in their conformational spaces. In the MD approach numerical algorithms are used to solve sets of classical, differential equations of motion. Positions of every atom from the system are calculated in consecutive time points during a typical MD simulation. A collection of coordinates of all atoms determined in one time-step is called a frame. Usually a Cartesian reference frame is used. The most important outcome of the MD run is an ordered sequence of frames called a trajectory. The computer-generated trajectory is encoded in GB-size files. Information useful for biologists, chemists or physicists must be somehow extracted from such massive data. The process involves classification based on predefined criteria. In this study we will focus our discussion on proteins. However, the algorithms described can be used for nucleic acids and other biopolymers as well. The analysis of the MD trajectory allows for estimates of proteins stability, mechanical properties, motions and eventually its biological function. The MD approach is used in fundamental research as

*Keywords.* Data mining, Petri net, molecular dynamic simulations, clustering, conformational space, graphs.

[1] Institute of Physics, Faculty of Physics, Astronomy and Informatics, Nicolaus Copernicus University, Grudziadzka 5, 87-100 Torun, Poland

[2] Faculty of Mathematics and Computer Science, Nicolaus Copernicus University ul. Chopina 12/18, 87-100 Torun, Poland. wiesiek@phys.uni.torun.pl

Article published by EDP Sciences

well as in drug design, biotechnology or in medicine [4,5,15]. Obviously, applications of the MD simulations are very important in the computational biology [7]. There are numerous variants of the MD simulations, recently steered MD method (SMD) [8], in which some atoms are fixed and others are pulled by a virtual force, is becoming more and more popular.

The MD simulation is performed using special software [3, 14, 19]; however, in analysis of the final MD trajectory a human interaction is a must. Some standard methods such as calculations of RMSD [6], contact maps [9], clustering [26] facilitate this analysis, but it is always a difficult task. The protein trajectory may be displayed by graphical software [11] and can be watched like a movie, but at temperature of 300 K all the atoms are in constant motion. Therefore it is very difficult to separate interesting conformational changes or functionally important motions from random atomic movements. Post-MD operational methods, such as essential dynamics [1], help to grasp physically important parts of a molecule dynamics and facilitate verification of scientific hypotheses. We expect that systematic methods aimed at reduced representation of a given MD trajectory (or a set of trajectories) are in high demand, since they might help to extract at least some features of the system studied in a form suitable for effective analysis, clear discussion and comparison with other data.

In order to facilitate the MD trajectory analysis process we have used Petri nets (PNs) [16,18,21]. So far we have developed four algorithms of PN generation from MD trajectories. The mapping from structural data to objects on PN graphs is different in each case. For example, in the One Place One Aminoacid (OPOA) algorithm one PN place corresponds to one amino acid of the protein modeled. In the One Place One Conformation (OPOC) scheme, which will be described here in detail, a place is linked to the whole protein conformation. The preliminary version of the OPOC algorithm has been already presented in our earlier paper [12], however, since that publication the alignment part of the OPOC algorithm has been improved. In our previous paper [12] only some general features of the OPOC algorithm were described, they are fully presented in this work.

We believe that the application of PNs powerful graph technique to the popular field of computer modeling of biopolymers, as suggested here, will bring new ideas for data mining in such complex sets as numerical MD trajectories.

## 2. The mathematical formulation

The idea of Petri net was coined by Carl Adam Petri in 1939 [18]. PNs are a mathematical modeling language designed for the description of distributed systems. A network has a form of a directed bipartite graph.

**Definition 2.1.** A Petri net graph is a 4-tuple $(P, T, F, W)$, where:

- P is a finite set of places.
- T is a finite set of transitions (or actions), such that $P \cap T = \emptyset$.
- F is a set of directed arcs, satisfying: $F \cap (P \times P) = F \cap (T \times T) = \emptyset$ (a place may be connected with the transition or the transition with the place; two places or two transitions cannot be connected).
- $W : F \to \{1, 2, 3, \ldots\}$ is a weight function assigned to arcs. As a default the weight of one is assigned to an arc.

In the PN plot places are represented as circles, transitions as squares, arcs as arrows. Arcs are annotated by weights. The default weight is 1 and it is usually omitted in the plot. An example of a Petri net plot is presented in Figure 1.

We do not have any actions in PN created according to Definition 2.1 – it is only a steady framework. In order to have actions we need to insert tokens into the PN. A distribution of tokens over the places is called marking.

**Definition 2.2.** A marking $M$ of a net N is a mapping $M : P \to \{0, 1, 2, 3, \ldots\}$.

**Definition 2.3.** Petri net is a set $(P, T, F, W, M_0)$ where $M_0$ is an initial marking, $P, T, F, W$ like in Definition 2.1.

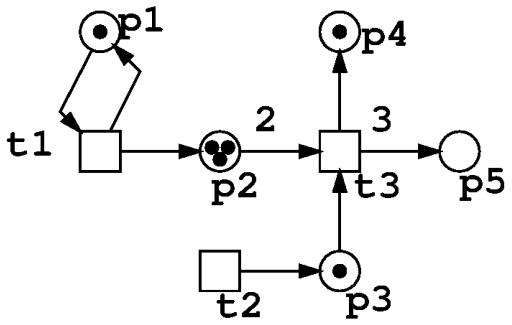


FIGURE 1. A Petri net plot.

**Definition 2.4.** For each element $t \in T$ we can define a set of input places $\bullet t = \{p \in P; (p,t) \in P\}$ − the set of places from which arcs run to the transition $t$ and a set of output places $t\bullet = \{p \in P; (t,p) \in P\}$ − the set of places to which arcs run from the transition $t$.

**Definition 2.5.** A transition $t$ may fire (such transition is called enabled) in a marking $M$ if the number of tokens in every input place p of the transition $t$ is equal or greater than the weight $W(p)$ assigned to an arc between the place $p$ and the transition $t$ in the marking $M$.

The transition $t$ consumes tokens from the input places and puts them into the output places − the number of tokens transferred is determined by the weights of arcs. The number of tokens is not constant, some transitions may put more or less tokens into their output places than they consume. In the OPOC algorithm every transition has only one input and one output place and every arc has the weight value of 1. One place can be at the same time an input and output place for one, two or more transitions.

Firing of transitions is a concurrent process. Transitions having common input places may compete for the tokens, and firing one transition may cause another transition to be inactive. The decision which transition should be fired is usually taken randomly.

## 3. The Petri net approach to MD data

In the OPOC algorithm the main idea is to link one conformation of the molecule with one place in the PN. There are numerous definitions of molecular conformations. Here a conformation will be a set of positions (coordinates) of C$\alpha$ atoms in $\mathbb{R}^3$ space. So, in the OPOC algorithm one place corresponds to positions of all C$\alpha$ atoms of the protein, and we assume that transitions represent changes between conformations induced by equations of motions adopted to run the simulation. A token marks the current conformation of the system studied. Because the analyzed molecule is at a given moment in one definite conformation, only one token is present in our PN.

Below we describe algorithms (in pseudo-code) leading to PN representations based on MD trajectories generated earlier. This "operational" analysis is just a simple post-processing of the existing MD files, but one can also "plug-in" these algorithms into his/her MD code.

The general idea of the OPOC algorithm is rather simple, however few critical issues have to be considered during the PN generation step, *e.g.*, one has to know whether the currently classified conformation has been already obtained in the previous steps or if there exists a transition between the current (time $t_i$) and the previous conformation (time $t_{i+1}$).

**Algorithm 3.1.** *One place one conformation algorithm (OPOC)*

*Input:* Files in Protein Data Bank (PDB) format containing the MD trajectories, a parameter threshold, given by the user, which sets a resolution necessary to distinguish two conformations.
*Output:* Petri net described by lists of places, transitions and arcs. Additionally files with description of places and transitions names are generated.

*Steps:*

```
1:  for (every PDB file) do
2:      frame ← readFrame()
3:      if (!places.find (frame, threshold)) then
4:          place ← createPlace(frame)
5:          places.add(place)
6:          transition ← createTransition(places.get(previousFrame), place)
7:          transition.add(transition)
8:          createArcs(places.get(previousFrame), transition, place)
9:      else
10:         place ← places.get(frame)
11:         if (!transition.contains(places.get(previousFrame), place) then
12:             transition ← createTransition(places.get(previousFrame), place)
13:             transitions.add(transition)
14:             createArcs(places.get(previousFrame), transition, place)
15:         end if
16:     end if
17: end for
```

The algorithm reads data frame by frame (lines 1-2) from every input file. During reading the new frame is checked if a structure coded in it was previously obtained (line 3). To compare the current conformation (from the current frame) and all the previous conformations (from the previous frames), which have been represented by already created places, an alignment algorithm is used. The alignment algorithm used in the present implementation is described as Algorithm 3.2. If a conformation from the current frame had not been previously obtained (lines 3-8) the new place is created (lines 4-5). Of course, if the place does not exist in the PN also a transition connected to this place is missing in the PN and such transition must be created (lines 6-7). After that, the place which represents that previous frame is connected as the input place to a newly created transition and the new place, which represents the conformation from the current frame is connected as the output place (line 8). If the current conformation has been already obtained (lines 9-16) it must be checked whether there is also a transition from that previous place/frame to the place which corresponds to the current frame (lines 11). If it does not exist (lines 12−14), the new transition is created (lines 12-13) and appropriate connections, like in the first part of the algorithm, are created (line 14).

The strong advantage of the PN approach is that during the PN graph generation additional values may be associated with places (or transitions) and saved. For example, an index of the frame when a transition occurs for the first time, the name of the input trajectory in which places or transitions occur, a counter showing how many times a place or a transition was invoked during generation. Such annotations can be very useful and enrich information coded in the Petri net.

The OPOC algorithm uses a predefined threshold to decide whether the two conformations should be classified into the same cluster (a place) or into two clusters. The threshold is defined in terms of root-mean-square distance (RMSD) between conformations v and w (in Å): $RMSD(v,w) = \sqrt{\frac{1}{n}\sum_{i=1}^{10} \|v_i - w_i\|^2}$, where the index $i$ denotes atoms and $n$ is a number of atoms in a studied structure.

The size of the generated PN depends on the value of the threshold parameter. For the increasing threshold an increasing number of places and transitions in a PN is observed, so the network will be more detailed. The

higher number of places will result in longer execution time, this is related to the search performed in line 5 of the pseudo-code. For very small threshold values (for example of 0.5 Å) almost every frame will generate a separate place (no clustering), for bigger values similar structures will be assigned to one place (clustering). In our opinion the best value of the threshold for an effective analysis of MD trajectories is around 1−1.5 Å, this is a value necessary to distinguish conformations. For bigger values the level of clustering is higher, the number of places is smaller and the more general observations are possible.

To decide whether the two conformations should be classified into the same cluster (a place) or into two clusters, a RMSD should be calculated. This RMSD depends on a transformation used to superimpose both structures. Performing a good structural alignment is a difficult problem [23]. In the implementation of the Algorithm 3.1 we have used our own superposition algorithm which is a compromise between execution time and the precision of the results. In the Algorithm 3.1 two structures are compared. For every pair of the corresponding amino acids two steps are performed − the first is a superposition of C$\alpha$ atoms based of a translation only, the second step is a rotation by a small angle to reduce the RMS distance between the structures.

**Algorithm 3.2.** *Structural alignment with rotations*

*Input:* Two structures A and B represented by lists of positions of C$\alpha$ atoms - the number of C$\alpha$ atoms must be the same in A and B, it is assumed that both structures represents the same protein, but being in different conformations. A threshold for the maximal rotation angle, the initial angle and the step of the rotation.
*Output:* RMSD calculated between structures.

*Steps:*

1: **for** (every CA atom) **do**
2: center(structureA, CA)
3: center(structureB, CA)
4: $RMSD \leftarrow calculateRMSD(structureA, structureB)$
5: $ang \leftarrow initialAngle$
6: **while** $((calculateRMSD(structureA, structureB) < RMSD) AND (ang < threshold))$ **do**
7: rotate(structureB, ang)
8: $ang \leftarrow ang + step$
9: **end while**
10: **end for**

The main part of the Algorithm 3.2 is a loop which iterates over every C$\alpha$ atom (lines 1-10). Firstly, both structures are translated in such way that corresponding C$\alpha$ atoms from the current loop are moved to the center of the coordinate system (lines 2-3). Then, the first RMSD between the transformed structures is calculated (line 4). The structure B is rotated by a small angle *ang* (line 5) in the next step. The rotation is continued until the accumulated rotation angle is larger than the desired threshold or the RMSD calculated after this rotation is bigger than that already calculated (lines 6-9). It is important that *rotate*() function represents a rotation over every coordinates' system axis and their combinations.

The Algorithm 3.2 is approximate and it can be used only for overlaps of structures (frames) representing the same protein and which are not too much different. In our OPOC algorithm two structures taken from the consecutive frames of the same trajectory are compared so it is expected that they will be quite similar. The general idea of Algorithm 3.2 is simpler than in well-known algorithms. We have performed tests (for the same inputs and in the same conditions) for the Algorithm 3.2, the Combinatorial Extension Algorithm (CEA) [27] and the FATCAT algorithm [29] implemented in BioJava suite [10]. The average difference in RMSD between Algorithm 3.2 and CA was 0.126 and between the Algorithm 3.2 and FATCAT was 0.096. However the average time of the execution of the Algorithm 3.2 was 306 ms, the CA algorithm was 1052 ms and FATCAT was 5398 ms. The structural alignment is usually performed (at least) many thousands of times for just one MD trajectory, thus its short execution time is desirable. Therefore, we claim that the RMSD error at 0.1 Å level is quite acceptable for us, especially if crude representations of bimolecular conformational spaces in PNs are

sought. In our software the Algorithm 3.1 may be also run with a structural alignment limited to translations only [12]. Although with such a setting it is much faster than Algorithm 3.2, the RMSD differences in structures are substantially bigger. Therefore, the alignment limited to translations was not used in the present paper.

## 4. Case studies

All MD trajectories analyzed in this work have been calculated using the NAMD package [19] with the CHARMM27 force field [2]. For visualization and generation of mutants the VMD code was applied [11]. Plots of Petri nets have been created in the Snoopy program [22]. Other calculations and PNs generations have been performed by our own software, available upon request from the authors.

The advantage of the OPOC algorithm is that it can be used to one trajectory and/or to many trajectories of the same protein as well. From a quite large pool of trajectories generated in our lab three examples are presented to illustrate the algorithm and its utility. It is important to show that a PN generated by OPOC contains (represents) major features and events from the trajectory used to create this particular network. Only then conclusions drawn from PNs will be reliable, especially when those conclusions refer to features which are difficult to pick up by a human inspection of the trajectory.

The first example is the PN of transthyretin (TTR, wild type): a protein tetramer stretched by SMD computer simulations into two dimers (ab+cd, see Fig. 2). The TTR initial structure has been obtained from the PDB record 1ICT [28]. The SMD pulling directions were determined by vectors connecting the centers of mass of the dimers. All C$\alpha$ atoms of TTRC-TTRD (see Fig. 2) segments were fixed and the C$\alpha$ atoms of the TTRA-TTRB dimer were pulled. The SMD forces were attached to all C$\alpha$ atoms of the pulled chains. The analyzed trajectory was 10 ns long. In a typical SMD simulation a structure of a protein is progressively deformed to examine its mechanical properties and to observe possible H-bond breaks between fragments. During the simulation the pulled and fixed segments separate from each other and the RMSD of the whole system rises in time. Knowing that in the OPOC algorithm an amendment of new elements to the PN depends on a value of the RMS distance between the consecutive structures (MD trajectory frames), one can predict how the generated PN should look like. Due to systematic deformations of the protein new places are created when the RMSD factor between

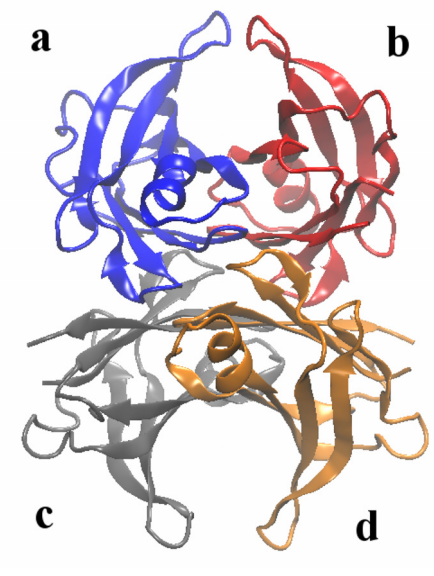


Figure 2. Structure of TTR with marked segments (a, b, c, d).

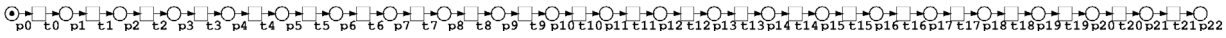


FIGURE 3. A Petri net generated for TTR SMD simulation with the threshold of 1.3 Å.

the current frame and places created previously is bigger than the assumed threshold. Thus, if SMD proceeds as expected, the network should have a trivial form of a one long chain composed of consecutive places and transitions. Indeed, such PNs have been obtained and their sizes were depend on the threshold only. In Figure 3 a PN generated for the threshold of 1.3 Å is presented. Nets for different thresholds were also generated and they have the same form (data not presented). It is worth noting that in this example we treat the TTR tetramer as a single structure.

One should note that if during the SMD stretching any unusual conformational transition happens indeed (for example, if some stable intermediate states are formed) this regular graph pattern will be perturbed. Thus, a simple inspection of such an easy to generate PN graph allows us to check in one glimpse "what is going on" in the corresponding SMD trajectory.

The next example is a PN generated for the SMD trajectory of forced dissociation of chemokine MCP-1 from its antibody. Structures of both proteins are present in the PDB database with an id of 2BDN [20]. In the paper [25] many SMD pulling directions were examined, for the current PN study a trajectory for just one vector was chosen randomly (V3 in [25]). In the simulation all C$\alpha$ atoms of the antibody were fixed and the pulling force was added to all C$\alpha$ atoms of the MCP-1. Similarly to the previous example, also here the structure of the complex is progressively deformed and the RMSD factor rises in the course of the simulation. The PN generated for the whole complex is expected to be similar to the one presented in Figure 3. However, the input for the OPOC algorithm does not have to be a list of all C$\alpha$ atoms from the structure but it may be a selected subset as well. We have generated PN for C$\alpha$ atoms from MCP-1 only, C$\alpha$ atoms from the antibody were omitted since they were fixed anyway. So, in this case the PN has been generated for the chemokine part only. MCP-1 is a very small molecule so it is easy to analyze conformational changes during the simulation by following the trajectory displayed on computer graphics and comparing structural events to the PN graph generated "on the fly".

In the first part of the simulation, MCP-1 is progressively deformed because of two opposite forces *i.e.* the pulling force and attraction forces between the antigen and the antibody. Finally those bonds broke and MCP-1 was separated from the antibody. The MCP-1 antigen was changing its shape even after the dissociation, probably because the antigen was not any longer bound in the complex and it might adopt a new conformation – more suitable for the unbound state. We have observed that the long loop at the end of the MCP-1 molecule came closer to the main body of the protein. In the final part of the simulation the chemokine is more stable and no substantial changes in the conformation have been observed. The initial conformation of the complex and three representative structures from the different part of the same simulation are presented in Figure 4a. For this SMD simulation the analysis with the OPOC algorithm was used with the threshold set at 1.1 Å. The value of the threshold in this case was smaller than before in order to obtain more details in the PN and generate a bigger network because the MCP-1 simulation was only 2 ns long. The corresponding PN is presented in Figure 4b. One can see that the major part of the network, except of the last places and transitions, has a form similar to that shown in the SMD TTR case (Fig. 3). It is a straight sequence of changing conformations, represented by a sequence of places and transitions. This strongly corresponds to observations made from the visual analysis of the trajectory. However some interruptions in this sequence occur, namely few times the protein goes back to a previous conformation or adopts new ones, and this affects the sequence in the graph. That features present in the PN graph indicate that the MCP-1 molecule has some opportunity to relax and to sample new regions of the conformational space. It may happen because, in contrast to the previous TTR example, the structure of MCP-1 is not deformed in steady way during the simulation. The last part of the network (places 40–43, transitions 48–55) represents a quite stable form of the chemokine, this event happens

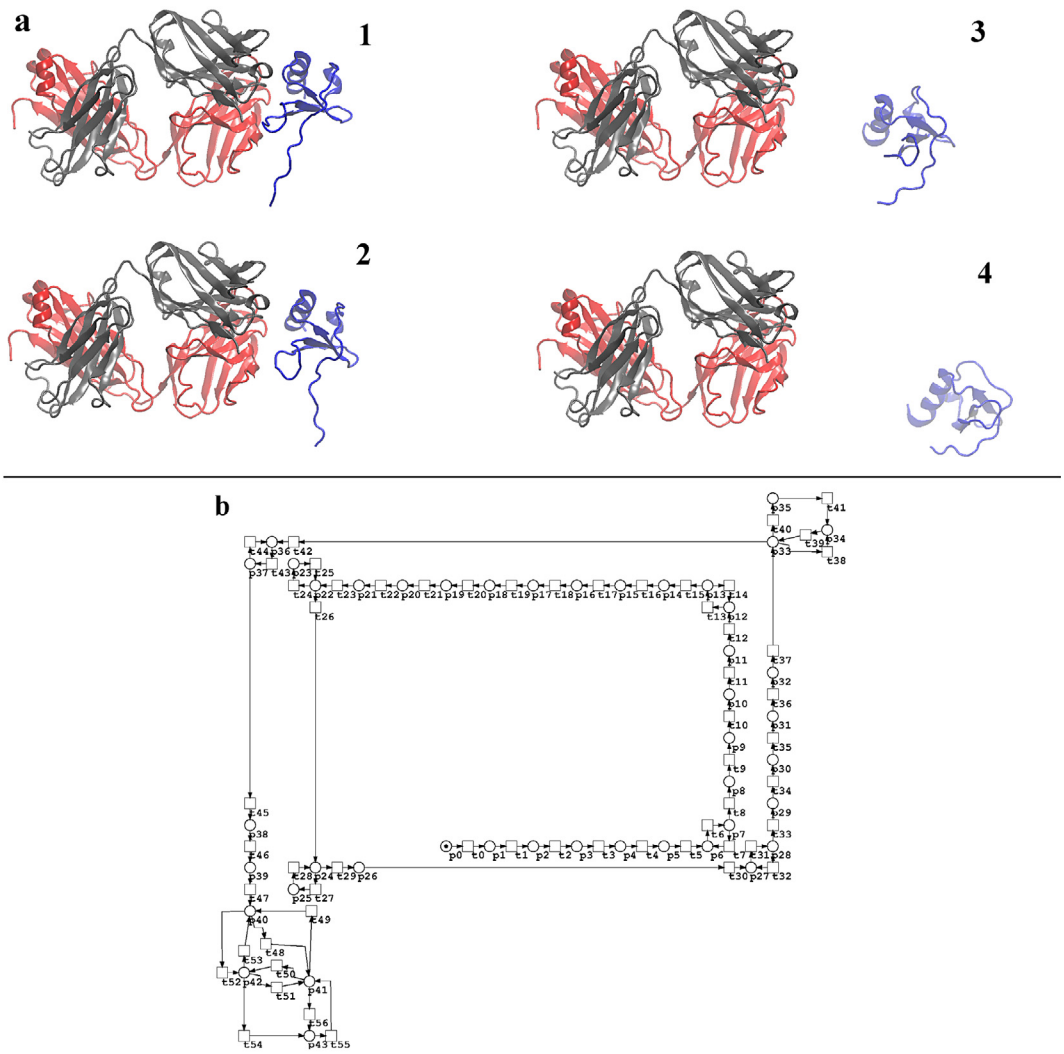


FIGURE 4. (a) Four frames from SMD simulation of MCP-1 (*blue*) dissociation from the antibody (*black, red*). 1 − the starting frame, 2 − the structure when bonds between the antibody and the antigen are broken, 3 − conformation changes during dragging, 4 − conformation obtained at the end of the simulation. (b) A Petri net generated by the OPOC algorithm for the same SMD trajectory. The high resolution figure is available at http://www-users.mat.umk.pl/~leii/rairo/figure4.png (Color online).

at the end of the trajectory, when MCP-1 is not deformed by acting forces but only dragged though the space. In this part of the PN one can see that MCP-1 changes freely its conformations back and forth.

Presented cases show that PNs generated by the OPOC algorithm strongly correspond to events happening in the MD trajectories. The last example will show advantages of the OPOC algorithm in analysis of several long simulations.

The third case studied is a PN created for simulations of TTR L55P mutant dynamics. The initial structure of the TTR was based on the 1ICT PDB entry [28]. Two distinct classical MD simulations were used to generate just one PN. Both trajectories were 100 ns long and were divided into two parts of 50 ns. The threshold of

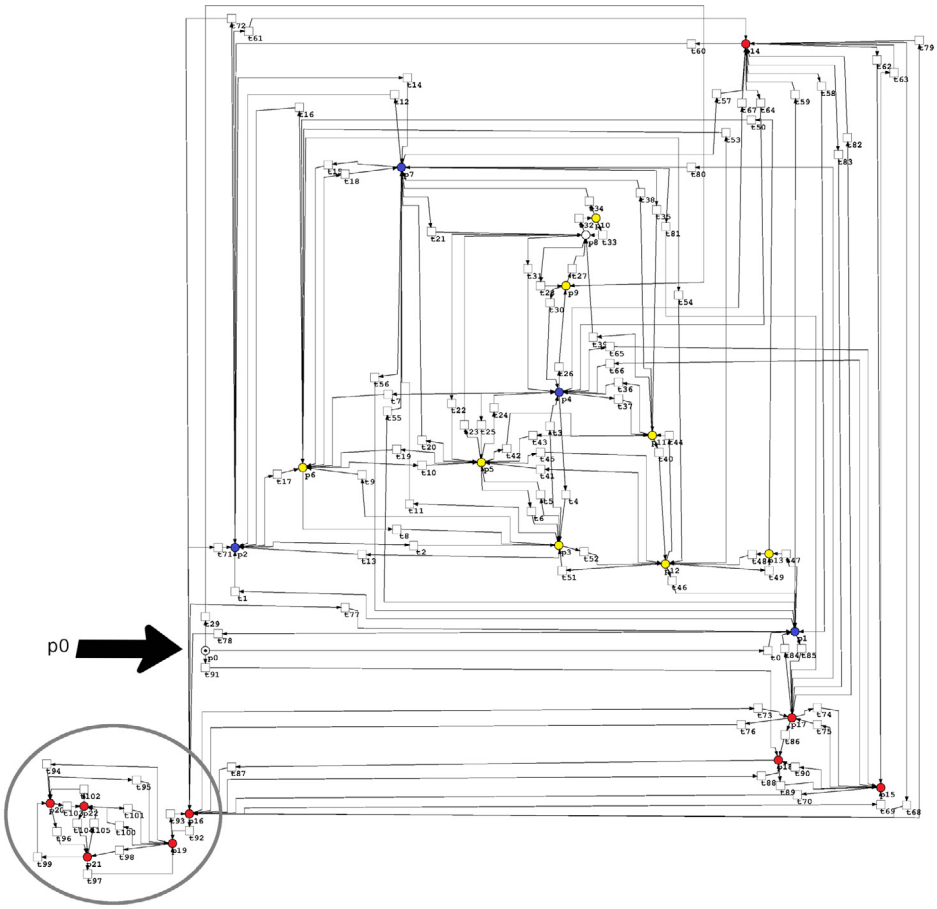


FIGURE 5. A schematic example of a complex Petri net shape generated by the OPOC algorithm for two 100 ns long MD simulations of TTR mutant L55P (1.5 Å threshold). Places common in both simulations are in blue, places obtained only in the 1st simulation are yellow, places present only in the 2nd simulation are red. The arrow shows p0 starting place. The high resolution figure is available at http://www-users.mat.umk.pl/~leii/rairo/figure5.tif (Color online).

1.5 Å was used in the OPOC algorithm. The bigger threshold was chosen in order to obtain a smaller network. The resultant PN is presented in Figure 5. This network has 22 places and 105 transitions. One can notice disproportion between those two numbers, for instance, the PN generated for the same number of simulations of the wild type TTR had only 19 places and 57 transitions (PN not shown). Because transitions represent changes between conformations, it is clear that the mutant L55P has been changing its conformation very often (>100 times) and it was quite floppy. More details may be extracted by careful analysis of the network.

The place p0 (Fig. 5, indicated by an arrow) is an artificial starting place. For both simulations the first conformation is the one represented by p1. Two additional transitions (t29, t91) for which p0 is an input place are created because of our division of simulations into two files and they do not have any true meaning. Only four conformations, represented by p1, p2, p4 and p7 (*blue circles*, Fig. 5), have been observed in both simulations and they were mainly present at the beginning of simulations, after which the trajectories diverged. Such splitting has been observed also in other proteins. Conformations represented by places up to p13 have been obtained from the first MD simulation (*yellow circles*, Fig. 5). One can observe (Fig. 5) that in this part of the network a

lot of transitions occur, some conformations can be transformed into any other, and also backwards transitions into initial conformations have been possible. This suggests that the protein cannot find a stable minimum in this MD simulation and it is still sampling the conformational space. Such observations would be very difficult to make by watching the MD trajectory only. In the second trajectory from the starting, common conformations protein can transform into conformations represented by places: p14, p16, p17 and further p15, p18 (*red circles*, Fig. 5). Between those conformations many transitions exist. However, from the conformation p16 the system can also transform into conformation p19 and even though there is an inverse transition, with the duration of the simulation, L55P "stays" in a separate part of the PN created by places p19–p22 (*grey area*, Fig. 5). The protein "oscillates" between those conformations which may suggest that they form some close-by local minima in the limited conformation space. This observation – *i.e.* a discovery of a separate subspace – would be very difficult to make by simply watching the trajectory. The created PN shows that both analyzed simulations were quite different dynamical processes.

## 5. Discussion and conclusions

Our case studies have shown that PNs generated by the OPOC algorithm properly reflect events and features present in MD trajectories, therefore the proposed approach can be used to analyze them. The OPOC algorithm is a type of a clustering algorithm – for hundreds or thousands of conformational frames it assigns a much smaller number of places located in a PN graph. Places in graphs correspond to clusters. The degree of PN clustering depends on the assumed threshold parameter, values from 1 Å to 1.5 Å range give a reasonable resolution of PN graphs. Clusters themselves can bring scientists useful information, therefore such type of MD data analysis is not new. Clusters of structures may be generated by numerous clustering algorithms [26]. However, the novel feature of the OPOC algorithm discussed here is a way of presentation of relationships between clusters. Clusters are connected *via*transitions, which are just as important as places. Thanks to transitions shown in PN graphs one can see from which conformation a protein can transform into another one and how those transformations proceed. Such type of study is also possible (in principle) using other algorithms, for example, in the Principal Component Analysis (PCA, [13]), where for every cluster a list of assigned frames are generated. However, in PCA to get relationships between clusters, additional tedious calculations are necessary. In the OPOC algorithm proposed here such an analysis is a natural part of the algorithm.

Moreover, relationships between conformations and their changes represented by places and transitions are generated as a PN graph. The PN graph describes a journey of the protein through the conformational space and can be easily displayed. Only small practice is required to characterize the process of simulations by looking at the PN graph. One can decide very quickly whether in the analyzed MD trajectory a protein finds any more stable conformations or constantly probes new regions of the conformational space. Such inspections of PN graphs help to decide what is the character of the studied dynamical process. Thus, the research on biomolecules might be further optimized. PN graphs, together with additional information gathered during a PN generation step give a holistic picture of the MD simulation. Such a complex analysis is very difficult and cannot be performed on a computer graphics only.

Nice feature of the OPOC algorithm is the fact that PNs have their own mathematical theory. For example, PNs are dynamic systems: transitions may be fired and tokens transferred. This process is called PN simulation [24]. By simulating PNs generated by the OPOC algorithm a sequence of conformations obtained from the real MD simulation can be quickly reproduced. However, one should note that because of a complicated structure of real networks (see Fig. 5) the sequence of conformations produced by the PN simulation will be only an approximation encompassing main features of the input MD simulations. This "PN regenerated" trajectory may be used in Monte Carlo studies of proteins or in search for conformational changes not necessarily directly observed in the MD.

PNs allow also for a general characterization of proteins conformational space [12]. More than 1800 ns of simulations of TTR and its two mutants were analyzed (here only a part of that data are presented) and each new trajectory added new conformations, only a few were sometimes reproduced. Owing to process simplification

that renders dynamics assessment more intuitive, PN graphs are an invaluable aid in exploration of the nature of such huge conformational spaces. We hope that algorithms presented in this study will facilitate research in computational biology and bioinformatics.

*Acknowledgements.* This work was supported by NCN Grant No. N202 262083 (WN) and Faculty of Physics, Astronomy and Informatics, NCU Grant No. 1623-F (RJ). AG thanks "Krok w przyszlosc – stypendia dla doktorantow V edycja" fellowship.

## References

[1] A. Amadei, A.B. Linssen and H.J. Berendsen, Essential dynamics of proteins. *Proteins: Structure Function and Bioinform.* **17** (1993) 412–25.
[2] B.R. Brooks *et al.* Charmm: the biomolecular simulation program. *J. Comput. Chem.* **30** (2009) 1545–1614.
[3] D.A. Case, Th. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K.M. Merz, A. Onufriev, C. Simmerling, B. Wang and R.J. Woods, The amber biomolecular simulation programs. *J. Comput. Chem.* **26** (2005) 1668–1688.
[4] J. Chen, Sh. Zhang, X. Liu and Q. Zhang, Insights into drug resistance of mutations d30n and i50v to hiv-1 protease inhibitor tmc-114: Free energy calculation and molecular dynamic simulation. *J. Mol. Mod.* **16** (2010) 459–468.
[5] A.A. Chen and A.E. García, High-resolution reversible folding of hyperstable RNA tetraloops using molecular dynamics simulations. *Proc. Natl. Acad. Sci.* **110** (2013) 16820–16825.
[6] F.E. Cohen and M.J.E. Sternberg, On the prediction of protein structure: the significance of the root-mean-square deviation. *J. Mol. Biol.* **138** (1980) 321–333.
[7] R.O. Dror, R.M. Dirks, J.P. Grossman, H. Xu and D.E. Shaw, Biomolecular simulation: a computational microscope for molecular biology. *Ann. Rev. Biophys.* **41** (2012) 429–452.
[8] H. Grubmüller, B. Heymann and P. Tavan, Ligand binding: molecular mechanics calculation of the streptavidin-biotin rupture force. *Science* **271** (1996) 997–999.
[9] L. Holm and Ch. Sander, Mapping the protein universe. *Science* **273** (1996) 595–602.
[10] R.C.G. Holland *et al.* Biojava: an open-source framework for bioinformatics. *Bioinform.* **24** (2008) 2096–2097.
[11] W. Humphrey, A. Dalke and K. Schulten, Vmd: visual molecular dynamics. *J. Mol. Graph.* **14** (1996) 33–38.
[12] R. Jakubowski, A. Gogolinska, L. Peplowski, P. Skrzyniarz and W. Nowak, Computational studies of ttr related amyloidosis: exploration of conformational space through petri net-based algorithm. *TASK Quarterly* **18** (2014) 289–300.
[13] I. Jolliffe, Principal Component Analysis. Wiley Online Library (2002).
[14] E. Lindahl, B. Hess and D.V. Der Spoel, Gromacs 3.0: a package for molecular simulation and trajectory analysis. *J. Mol. Mod.* **7** (2001) 306–317.
[15] K. Lindorff-Larsen, N. Trbovic, P. Maragakis, S. Piana and D.E. Shaw, Structure and dynamics of an unfolded protein examined by molecular dynamics simulation. *J. Amer. Chem. Soc.* **134** (2012) 3787–3791.
[16] T. Murata, Petri nets: Properties, analysis and applications. *Proc. of IEEE* **77** (1989) 541–580.
[17] W. Nowak, Applications of Computational Methods to Simulations of Proteins Dynamics. In *Handbook Comput. Chem.* Springer (2012) 1127–1153.
[18] J.L. Peterson, Petri nets. *ACM Comput. Surveys (CSUR)* **9** (1977) 223–252.
[19] J.C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, Ch. Chipot, R.D. Skeel, L. Kale and K. Schulten, Scalable molecular dynamics with namd. *J. Comput. Chem.* **26** (2005) 1781–1802.
[20] C. Reid *et al.* Structure activity relationships of monocyte chemoattractant proteins in complex with a blocking antibody. *Protein Engineering Design and Selection* **19** (2006) 317–324.
[21] W. Reisig, Petri nets: an introduction, vol. 4. Springer Science & Business Media (2012).
[22] Ch. Rohr, W. Marwan and M. Heiner, Snoopya unifying Petri net framework to investigate biomolecular networks. *Bioinform.* **26** (2010) 974–975.
[23] A. Godzik, The structural alignment between two proteins: Is there a unique answer? *Protein Sci.* **5** (1996) 1325–1338.
[24] A. Gogolinska and W. Nowak, Petri nets approach to modeling of immune system and autism. In *Artificial Immune Systems.* Springer (2012) 86–99.
[25] A. Gogolinska and W. Nowak, Molecular basis of lateral force spectroscopy nano-diagnostics: computational unbinding of autism related chemokine MCP-1 from IgG antibody. *J. Mol. Mod.* **19** (2013) 4773–4780.
[26] J. Shao, S.W. Tanner, N. Thompson and Th.E. Cheatham, Clustering molecular dynamics trajectories: 1. Characterizing the performance of different clustering algorithms. *J. Chem. Theory Comput.* **3** (2007) 2312–2334.
[27] I.N. Shindyalov and Ph.E. Bourne, Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein Engineering* **11** (1998) 739–747.
[28] A. Wojtczak, P. Neumann and V. Cody, Structure of a new polymorphic monoclinic form of human transthyretin at 3 å resolution reveals a mixed complex between unliganded and t4-bound tetramers of TTR. *Acta Crystallographica Section D: Biological Crystallography* **57** (2001) 957–967.
[29] Y. Ye and A. Godzik, Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinform.* **19** (2003) ii246–ii255.