# Protein Folding by Bias Exchange Metadynamics on a Grid Infrastructure

Riccardo di Meo[1], Fabio Pietrucci[2], Alessandro Laio[2,3]
and Stefano Cozzini[3,1*]

[1] *The Abdus Salam International Centre for Theoretical Physics, Trieste, Italy*
[2] *SISSA-ISAS, Trieste, Italy*
[3] *CNR-INFM DEMOCRITOS National Simulation Center, Trieste, Italy*

*Lecture given at the*
*Joint EU-IndiaGrid/CompChem Grid Tutorial on*
*Chemical and Material Science Applications*
*Trieste, 15-18 September 2008*

LNS0924008

*cozzini@democritos.it

**Abstract**

We describe the BEMuSE computational tool that combines the advantages of distributed Grid computing infrastructure with the power of bias-exchange metadynamics, an enhanced sampling methodology specifically tailored for simulating complex conformational changes in biomolecules. By using this combined approach the Advillin headpiece is folded on a Grid infrastructure with moderate usage of resource (about 10 computational nodes) using an accurate but computationally expensive potential energy function describing the solvent molecules explicitly. Benchmarking analysis shows that, despite of the small amount of information that is exchanged among the processes on the Grid, the performance of the combined approach is not reduced with respect to the original MPI implementation. Using our algorithm on the large-scale Grid resources that are available nowadays can potentially make folding simulations of large proteins accessible.

# Contents

# 1 Introduction

Molecular dynamics simulations with accurate all-atom force fields [1, 29, 17] are a precious tool for studyng the mechanism of complex biophyiscal processes, such as protein folding and enzymatic reactions. Unfortunately, the time scale of direct simulation is limited to a few nanoseconds per processor per day, while proteins require between a microsecond and several seconds to fold. Even longer times may be required to observe the binding/unbinding between drugs and enzymes, or other biophysical processes.

Large and distributed computing facilities such as Grid infrastructures offer an invaluable occasion to face this problem, and bridge the time-scale gap between computer simulations and experiments. A succesful approach based on this idea has been recently proposed by the group of V.S. Pande (Folding@Home project) [25]. The algorithm consists in running short molecular dynamics simulations on thousands of home PCs, communicating through the Internet. Each simulation starts from an unfolded geometry of the protein and runs independently from the others. When one of the simulations, by a random thermal fluctuation, performs a transition to a different conformation of the protein, some of the simulations are restarted in this new state. This is iterated until the folded state is reached. By this approach, the 36-amino acids villin headpiece protein has been folded starting from an extended structure, obtaining a folding rate compatible with experiments [7]. However, the computational time required for this approach is impressive, of the order of 1000 years of CPU time divided among tens of thousands of processors.

In order to explore more efficiently the configuration space it is convenient to exploit some methodology that is capable to accelerate rare events. Several methods have been developed with this scope in the last years, and important advances have been made in many fields, ranging from solid-state physics to biochemistry [3]. A recent approach that has been successfully applied to study protein folding is bias-exchange metadynamics (BEM) [21]. The algorithm allows to obtain, at a moderate computational cost, detailed thermodynamic and kinetic information on the folding process [18]. For small proteins, such as the *Trp-cage* (20 amino acids), *Insulin* (30 amino acids) and *Advillin* (36 amino acids) it allows to predict the folded state using an accurate explicit water Hamiltonian using $\sim 10$ parallel simulations of only $\sim 100$ ns each [21, 22, 28]. The same methodology has been successfully applied to the study of enzymatic reactions and binding processes [15, 24].

Here we demonstrate that it is possible to combine the advantages of Grid computing, namely the availability of a large pool of loosely coupled computational resources, with the power of BEM [14, 13], which uses efficiently the available computer time. The two approaches were combined by developing a specific and optimized computational tool (BEMuSE), aimed at performing large simulation using BES algorithms in a simple and efficient way on different and unrelated computational infrastructures. We find that, remarkably, the performance of the BEM algorithm on the Grid is comparable to the one of its MPI implementation. This ultimately relies on the loosely coupled nature of the algorithm, that fits perfectly on the Grid paradigm, where multiple geographically distributed computational resources cooperate together. We anticipate that the features of the combined algorithm will extend the scope of Grid facilities and of BEM in the quantitative study of biomolecules by computer simulations.

This paper is organized as follows: in section 2 we review shortly the BEM algorithm. In section 3 we present the Grid-enabling procedure and describe the BEMuSE tool in detail. In Section 4 we discuss the results obtained with the approach. Finally, in section 5 we draw some conclusions.

## 2 Methods

### 2.1 Bias-exchange metadynamics

BEM [21] is a technique that allows overcoming the free-energy barriers associated to slow conformational motions that are common in biomolecules, providing at the same time a detailed reconstruction of the free-energy landscape of the system as a function of a large number of reaction coordinates. It is based on the combined use of replica-exchange [27, 11, 8] and metadynamics [14]. A set of $\sim 10$ collective variables (CVs) are chosen, which are expected to be relevant for describing the process (i.e. the number of hydrophobic contacts and hydrogen bonds, the helical content, the gyration radius, etc.). Then, a number of MD simulations are run in parallel, biasing each replica with a history-dependent potential $V_G(x,t)$ in the form of potential-energy Gaussians added along the trajectory, acting on just one or two CVs. At fixed time intervals, swaps of the history dependent potentials between pairs of replicas are attempted. The swap is accepted with a probability:

$$min(1, e^{\beta(V_G^a(x^a,t)+V_G^b(x^b,t)-V_G^a(x_b,t)+V_G^b(x_a,t))})$$

where $x^a$ and $x^b$ are coordinates of replica $a$ and $b$. In this manner each trajectory evolves through the high-dimensional free-energy landscape defined by the CVs, sequentially biased by low-dimensional potentials acting on one or two CVs at each time. Details on the choice of the CVs and of the other parameters of the simulation are given in the Results section. The BEM algorithm has been implemented in a modified version of the Gromacs 3.3 package [16].

## 2.2 Computational analysis of the BEM algorithm

A detailed analysis of the computational requirements of the BEM algorithms has been carried out in order to identify the best porting strategy. This was done in strict cooperation with BEM developers and users who actually guided the development of the application along their needs. We identified the favorable algorithm characteristics with respect to the Grid infrastructure at disposal in terms of network bandwidth usage and latency requirements.

The BEM algorithm can be classified as CPU intensive and loosely coupled. I/O requirements are negligible. The original MPI implementation was based on a slightly modified version of Gromacs [6] where the BEM algorithm was actually implemented as an MPI wrapper around the MD engine.

Every MPI process can thus be regarded as an almost independent "walker" evolving a molecular dynamics trajectory (called from now on a MD walker) and periodically exchanging information on the phase space sampling with all the other walkers. The data exchanged are indeed negligible in size. Moreover, the frequency of the exchanges is very low (order of tens of minutes). This makes the algorithm scalable over a large number of processors without any need of the high speed networks that are typical of an HPC platform. This makes BEM quite suitable for EGEE/gLite Grid infrastructures, mainly composed by farm-like computational resources geographically distributed, each of them identified as a Computing Element (CE). However, MPI support on such an infrastructure is far from optimal and usually limited within each Computing Element. Thus, no MPI program can run across different geographically distributed resources.

We therefore decided to avoid the porting of the original MPI implementation of the algorithm in favor of a client/server architecture where computational nodes behind the CE (the so-called Worker Nodes (WNs)) can be recruited among different Computing Elements and exchanges among them can be managed by the server. Such an approach not only overcomes the above problem but allows the dynamic recruiting of computational resources. This means that relatively large BEM complex experiments can be performed in a asynchronous way, without being forced, as in the MPI synchronous approach, to wait until all the required CPUs are available.

Such a scenario modifies significantly the normal BEM workflow: one wonders how does the algorithm react to the asynchronous scheduling and availability of the resources on a heterogeneous Grid infrastructure, and how MD walkers running on different CPUs (and thus at different speed) contribute to the simulation. We will discuss these issues in section 4.

## 3 Grid enabling deployment

Client and Server were implemented in python 2.5 and designed with the following important aspects in mind:

- **Modularity**. The idea is to keep scientific application as much as possible decoupled from the technical implementation of the client/server mechanism. The scientific code is thus kept oblivious about how and where the simulation is running. All the exchanges are performed in a higher, separated layer, which hides all the computational infrastructure complexity. This approach guarantees the easiness of further enhancement of the scientific aspects of the algorithm and keeps the python layers compatible with future versions of it.

- **Portability and Interoperability**. The client/server mechanism should be easily portable in different environments. Moreover, the software should be able to interoperate in different computational infrastructures at the same time: i.e. one single simulation can be performed on resources belonging to different platforms (like Grid infrastructure, desktop computers and HPC resources) even at the same time.

- **Fault Tolerance**. We consider of outmost importance to provide fault tolerance mechanisms to cope with failure in the underlying infrastructure. The software is responsible to maintain the integrity of

the simulation even in the event of numerous client crashes, with a limited external intervention by the user.

Our client/server modular implementation coexists with the original MPI implementation: it is possible to run the same gromacs executable in both client/server and MPI mode. This was achieved including only a small patch in the modified gromacs package that allows clean shutdown of the Grid functionalities.

The exchanges required in BEM are no longer performed by the MD program, but are now implemented in our server: this means that with a minimum effort we can adapt our client/server approach to execute the same simulation with a different molecular dynamics engine, provided that normal metadynamics is implemented.

Figure 1 presents the BEMuSE architecture and we will refer to it in the subsequent discussion where server and client components are presented in detail.

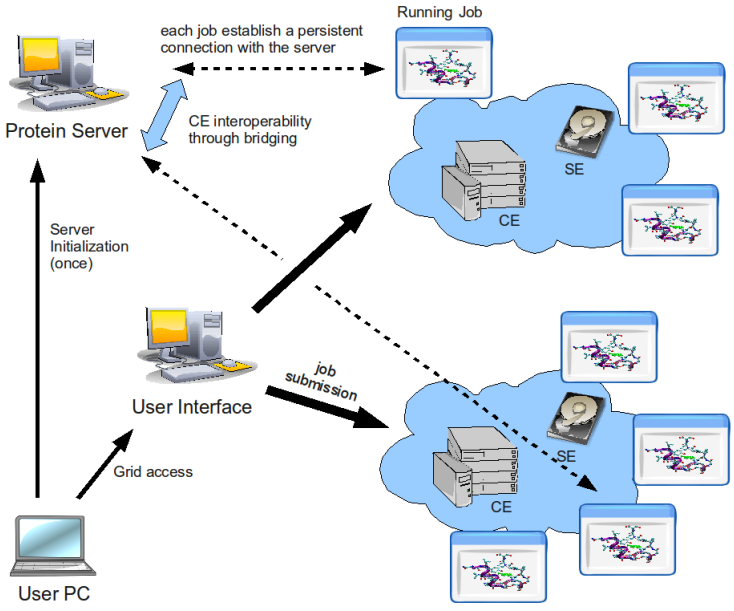


Figure 1: The overall architecture of BEMuSE. The server is launched by the users and then monitored. Job submission mechanism is handled externally.

### 3.1 The server

The server, which runs on a resolved host (usually but not necessarily on the top of the User Interface (UI)) actually implements the BEM algorithm: it has the task to accept the connections from the WN, associate them to a MD walker (by sending specific input data) and bridge the communication among them at each Bias Exchange.

It receives the connections from the clients as soon as the submitted Grid jobs start executing on WNs (possibly rejecting them if the maximum number of walkers is reached), handling at each time the Bias exchange procedure with the available clients (if any).

At fixed time intervals, the server randomly groups all available clients into couples and verifies if, for each pair, the bias exchange should be accepted or not. Moreover, it downloads all simulation data from each client to synchronize itself in the case of a client failure. Finally, if the exchange is accepted, sends the bias of the first client to the second and vice versa.

This setup offers several advantages: it synchronizes the exchanges between different walkers and provides a centralized clock to the simulation. Moreover, the approach not only enables communication among WNs belonging to different Computing Elements (CE) but also among resources belonging to different computational infrastructures. It also deals with job failures, keeping a synchronized copy of the data in order to cope with sudden crashes of the worker nodes. Finally, it ensures that all the restart files not saved during exchanges are periodically stored in the server filesystem by downloading them in background from each client.

The synchronization of most of the physical data allows the researcher to constantly keep track of the status of the simulation, to check its correctness and to run statistical analysis tools on the data. The server stores also detailed logs on the status of each client (updated in real time) which can be used for debugging as well as to gather statistics on the performance of all the processes on the Grid.

### 3.2 The client

The client software will be executed on the WN and requires at most four parameters: two for the location of the server (host and port), a password for authentication and an optional label which will identify the computational infrastructure where it will run on. The label allows the server to provide the correct information about the environment in a complete and transparent

way to the user.

Once the job has landed on the WN, a bootstrapping bash script is executed in order to download and install the correct version of python if not available locally, and then launch the python client.

The client has the task to start an MD walker, to stop it when requested by the server (at every exchange cycle), to handle data transfer from and to the server and to restart the MD walker, if requested by the server. This approach is simple and completely transparent to the MD engine executed on the WN: this means there is no need to make it aware of the computational environment where it is executed

In more details: at the beginning the client contacts the server and, after a basic authentication, receives a storage location[1] where to retrieve the gromacs binary (which can be specifically tailored for each infrastructure), the data required to start the simulation, and another storage location where to upload the statistical data.

The client then launches the simulation in background and contacts the server to inform it is waiting for the next exchange cycle. Gromacs, the molecular engine, is highly optimized and easily portable to a wide range of computational architectures: this is indeed a quite favorable situation for Grid approach due to the heterogeneity of the resources available on the Grid.

### 3.3 How the computational resources are obtained

The task to recruit computational resources is actually completely decoupled from our client/server software. The user is demanded to submit a certain number of identical jobs according to his needs.

We provided a dedicated interface (using the curse library, to accomodate the lack of graphic in the majority of the UIs), to configure a simple simulation and submit the jobs on a Grid-like environment. However, this is not strictly needed and even better results can be obtained by manually configuring and submitting the jobs. This allows the user to carefully choose computational resources that can deliver better performance both in terms of speed and reliability.

We note that the decoupling of the submission mechanism provides an easy way to make the whole application interoperable among different com-

[1] a variety of different protocol can be employed: gsiftp:, ftp:, http: and other

putational platforms, including desktop computers which can be used without any installation of specialized software.

Tests have been successfully executed in scenarios involving heterogeneous infrastructures where Grid, HPC and desktop computers were all used at once in the same simulation: this can be very advantageous and allows tackling simulations with a large number of walkers.

# 4 Results

## 4.1 Benchmarking run

As a benchmark to test the performance of the Grid implementation of BEM compared to MPI, we simulated the folding of a 36-amino acids protein, the human Advillin c-terminal headpiece subdomain (pdb code 1UND [20]). The protein is formed by three highly packed alpha helices and an hydrophobic core. Due to its fast folding rate, Advillin has been used as a benchmark system for several computational investigations of protein folding, including massive distributed computing [7].

We performed NPT molecular dyanmics simulations at $p = 1$ atm and $T = 330.15K$, using Berendsen barostat and thermostat with relaxation times of 4 and 2 ps, respectively. We adopted the Amber03 all-atom force field [4] and TIP3P water model [12]. The protein was solvated with 3633 water molecules in a 121 $nm^3$ cubic periodic box. The net positive charge of the system was neutralized with two Cl-ions. The particle-mesh Ewald method [2, 5] was used for long-range electrostatic with a short-range cutoff of 0.8 nm. A cutoff of 0.8 nm was also used for the Lennard-Jones interactions. All bond lengths were constrained to their equilibrium distance with the LINCS algorithm [9]. Overall, the molecular dynamics setup is identical to that of Ref. [22].

For BEM we used the following five collective variables (CVs): helical content in the first and second half of the protein, number of backbone hydrogen bonds, number of hydrophobic contacts, number of salt bridges. None of these variables requires the a priori knowledge of the folded state. Details on the definition the CVs can be found in Ref. [22]. Each CV is biased by a metadynamics potential on a different walker. The Gaussian have a height of 0.2 kJ/mol and a width of 0.8, 0.8, 2, 4, and 0.5, along the five CVs. Gaussians are deposited every ps. On the Grid, bias exchanges are attempted every 30 min of clock time, corresponding to typical simulation

times between accepted exchanges of 20–700 ps, depending on the CPU model. In the MPI simulation, used for comparison, bias exchanges were attempted every 20 ps [22].

Simulations have been run on different numbers of clients (8, 16), on different CPU architectures, both homogeneous and heterogeneous, and with different degrees of synchronization between processes. In this way the performance of our algorithm in a realistic Grid infrastructure can be directly compared to an MPI simulation with the same setup on 8 processors. In Fig. 2 we report the fraction of protein conformations explored as a function of simulation time/processor. This fraction is evaluated by dividing the five-dimensional CV space explored by the MPI simulation in small cubic cells (approximately 4000), and by counting how many of these cells are explored by at least one process during the Grid simulation. As is evident in Fig. 2, all the typologies of Grid simulations explore efficiently the conformation space. This is an important result, which points to the fact that, in passing from MPI to Grid, the loss of synchronization between processes and the heterogeneous architecture of the CPUs does not lead to a sizable loss of performance. Instead, exchanging the bias potential less often marginally improves the performance of the algorithm. This apparently surprising result can be rationalized considering the features of BEM, in which the enhanced sampling is achieved by successively biasing different variables on each replica. The performance turns out to be optimal if each replica retains the bias for a sufficiently long time, in such a way that it performs a significant conformational change before getting biased in a different direction.

Four different trajectories of the Grid simulations (16 proc. hetero in Fig. 2) reach conformations close to the experimental folded state, measured as 3.5 A RMSD from the NMR positions of the $\alpha$-carbons (neglecting the first and last three residues of the chain, which are mobile in experiments). In Some of the structures with RMSD $< 4\mathring{A}$ are shown in Fig. 3: clearly their fold is very similar to the native state, with the three $\alpha$-helices formed and a correct overall topology.

Beside the C-$\alpha$ RMSD, another insightful indicator of the progress of folding is the cartesian distance, in CV space, between the folded state and the simulated trajectory (Fig. 4). For a same total simulation time, the Grid simulations tend to approach the folded region in CV space faster than MPI. This is consistent with the faster exploration of CV space reported in Fig. 2. Again, the more efficient exploration can be explained by the longer

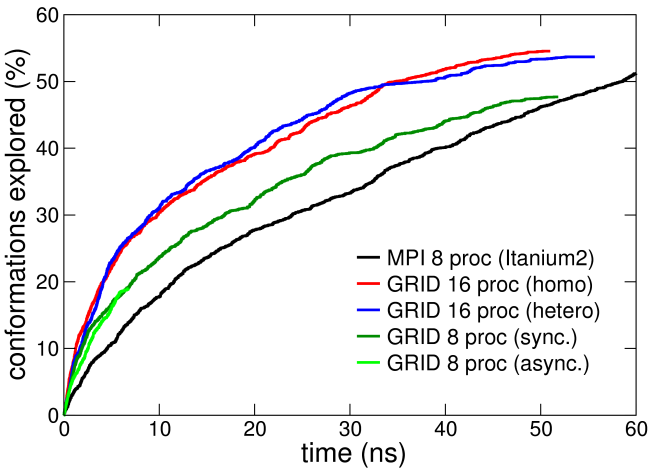


Figure 2: Explored fraction of conformational space as a function of the total simulation time. The reference conformations are taken from a converged BEM MPI simulation.

average exchange time in Grid simulations compared to MPI.

### 4.2 Production runs

As further validation of our Grid implementation of BEM several other systems have been simulated. First, the folding process of the 56-residues protein src-SH3 (pdb code 1SRL) [26, 10] has been simulated in explicit water using the BEMuSE approach and the same molecular dynamics setup described above. The native fold of this protein contains three beta strands and a small alpha helix, with a complicated topology which leads to an experimental folding time of several seconds. To date systems of this size and complexity are considered out of reach for accurate atomistic simulations. We employed BEM simulations on the Grid with up to 40 replicas, starting from extended states and biasing either C $\alpha$ inter-residue distances or contacts, defined by a switching function (see Ref. [22]). Preliminary results show the systematic exploration of a very large number of structures with different topologies. The formation of beta sheets turns out to be the bottleneck in the exploration of protein structures. The large amount of data collected shed light on the complexity of the folding free energy landscape and enabled significant optimization of the simulation setup, in particular of

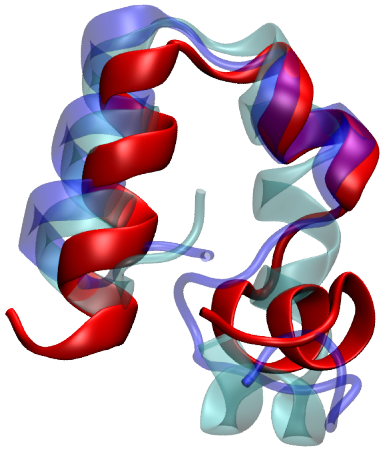

Figure 3: NMR native state of Advillin-headpiece (red) compared with three similar structrues ($C-\alpha$ RMSD $< 4\mathring{A}$) obtained by BEM on the Grid (blue).

the reaction coordinates driving the formation of beta sheets [23].

Finally, using BEMuSE we simulated the binding of a peptide substrate (Thr-Ile-Met-Met-Gln-Arg, cleavage site p2-NC) to HIV-1 protease [24]. Inhibition of this enzyme is the primary pharmaceutical strategy to treat AIDS, and a better understanding of the mechanism by which drug molecules bind to the protein would help designing more effective inhibitors less subject to mutation-induced drug resistance [19]. The binding process has been simulated for the first time in full atomistic detail, including the flexibility of the receptor and explicit water molecules. The detailed BEM setup is described in Ref. [24]. Up to 30 Grid nodes have been employed, biasing 7 reaction coordinates. The performance of BEMuSE is once more shown to be comparable to a reference standard MPI simulation which employed 8 replicas. In particular, for a given total simulation time a comparable number of configurations and volume of CV-space are explored. The good performance of BEMuSE opens the possibility to exploit the availability of large Grid computational resources to compare the mechanism of binding of several inhibitors, which might improve the effectiveness of the ongoing drug-design research effort.

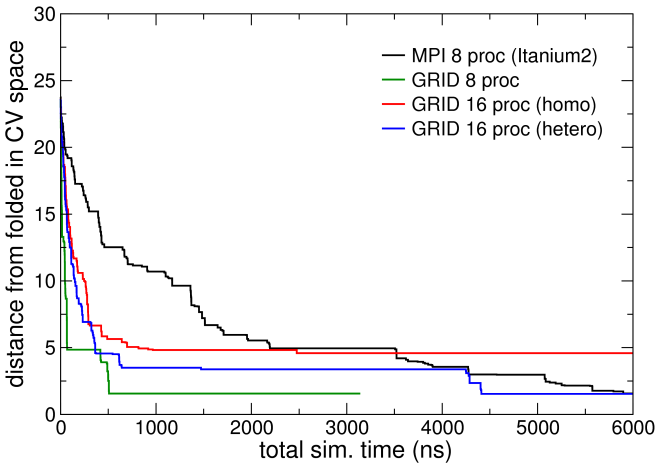


Figure 4: Minimum distance in CV-space from the folded state as a function of the total simulation time. The widths of metadynamics Gaussians are employed as units for the CV-space in order to compute the distance.

## 5 Conclusions

We reported and discussed a Grid-enabling procedure of the BEM algorithm and the BEMuSE tool originated by such work. Our benchmarking analysis shows that our approach is comparable in performance with the original MPI approach executed on HPC platforms. Using such a tool on the nowadays available large-scale Grid infrastructure can potentially increase the range of accurate protein folding simulations to larger proteins. We designed our tool to be as much as possible independent from the computational infrastructure ensuring in this manner portability and easiness of usage. The approach used guarantees as well that BEMuSE can be easily run across different computational infrastructures. This allows BEMuSE users to aggregate large scale computational resources to perform a BEM simulation.

The reported simulations of protein folding and protein-drug interaction demonstrate that the BEMuSE environment is well suited to address prominent scientific problems of great complexity. We expect several important researches to be carried out in the near future using this facility, in particular the comparison of the binding pathways of different drugs to important targets like HIV-1 protease and prion protein, and the elucidation of the folding process of proteins of larger and larger size.

## References

[1] Wendy D. Cornell, Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz Jr., David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell and Peter A. Kollmann, (1995). J. Am. Chem. Soc., 117:5179–5197

[2] T.A. Darden and D. York, (1993). Particle mesh ewald - an n.log(n) method for ewald sums in large systems. J. Chem. Phys., 98:10089

[3] C. Dellago and P.G. Bolhuis, (2008). Advanced Computer Simulation Approaches for Soft Matter Sciences III, volume 221 of Advances in Polymer Science. Springer Berlin/Heidelberg

[4] Y. Duan, C. Wu, S. Chowdhury, M.C. Lee, G.M. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J.M. Wang and P. Kollman, (2003). A point-charge force field for molecular mechanics simulations of proteins based on condensed-phase quantum mechanical calculations. J. Comput. Chem., 24 (16):1999–2012

[5] U. Essman, L. Perera, M.L. Berkowitz, T.A. Darden, H. Lee and L.G. Pedersen, (1995). A smooth particle mesh ewald method. J. Chem. Phys., 103:8577

[6] Gromacs website. http://www.gromacs.org

[7] Guha Jayachandran, V. Vishal and Vijay S. Pande, (2006). Using massively parallel simulation and Markovian models to study protein folding: examining the dynamics of the villin headpiece. The Journal of chemical physics, 124(16):164902

[8] U.H.E. Hansmann, (1997). Parallel tempering algorithm for conformational studies of biological molecules. Chem. Phys. Lett., 281:140

[9] B. Hess, H. Bekker, H.J.C. Berendsen and G.E.M.J. Fraaije, (1997). Lincs: A linear constraint solver for molecular simulations. J. Comput. Chem., 18:1463

[10] I.A. Hubner, K.A. Edmonds and E.I. Shakhnovich, (2005). Nucleation and the transition state of the sh3 domain. J. Mol. Biol., 349:424–434

[11] K. Hukushima and K. Nemoto, (1996). Exchange Monte Carlo method and application to spin glass simulations. J. Phys. Soc. Jpn., 65:1604

[12] W.L. Jorgensen, J. Chandrasekhar, J.D. Madura, R.W. Impey and M.L. Klein, (1983). Comparison of simple potential functions for simulating liquid water. J. Chem. Phys., 79:926–935

[13] A. Laio and F.L. Gervasio, (2008). Metadynamics: a method to simulate rare events and reconstruct the free energy in biophysics, chemistry and material science. Rep. Prog. Phys., 71:126601

[14] A. Laio and M. Parrinello, (2002). Escaping free-energy minima. Proc. Natl. Acad. Sci. U. S. A., 99(20):12562–12566

[15] V. Leone, G. Lattanzi, C. Molteni and P. Carloni, (2009). Mechanism of action of cyclophilin a explored by metadynamics simulations. PLoS Comput. Biol., 5:e1000309

[16] E. Lindahl, B. Hess and D. van der Spoel, (2001). GROMACS 3.0: a package for molecular simulation and trajectory analysis. J. Mol. Model., 7(8):306–317

[17] A.D. MacKerell Jr., D. Bashford, M. Bellot, R.L. Dunbrack Jr., J.D. Evanseck, M.J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F.T.K. Lau, C. Mattos, S. Michnick, T. Ngo, D.T. Nguyen, B. Prodhom, W.E. Reiher III, B. Roux, M. Schlenkrich, J.C. Smith, R. Stote, J. Straub, W. Watanabe, J. Wiorkiewicz-Kunczera, D. Yin and M. Karplus, (1998). J. Phys. Chem B, 102:3586–3616

[18] F. Marinelli, F. Pietrucci, A. Laio and S. Piana (2009). A kinetic model of trp-cage folding from multiple biased molecular dynamics simulations. PLoS Comput. Biol., in press

[19] H. Ohtaka and E. Freire, (2005). Adaptive inhibitors of the hiv-1 protease. Prog. Biophys. Mol. Biol., 88:193

[20] Protein data bank website. http://www.rcsb.org/pdb

[21] S. Piana and A. Laio, (2007). A bias-exchange approach to protein folding. J. Phys. Chem. B, 111(17):4553–4559

[22] S. Piana, A. Laio, F. Marinelli, M. Van Troys, D. Bourry, C. Ampe and J.C. Martins, (2008). Predicting the effect of a point mutation on a protein fold: The villin and advillin headpieces and their Pro62Ala mutants. J. Mol. Biol., 375(2):460–470

[23] F. Pietrucci and A. Laio, (2009). A collective variable for the efficient exploration of protein beta-sheet structures: application to sh3 and gb1

[24] F. Pietrucci, F. Marinelli, P. Carloni and A. Laio, (2009). Substrate binding mechanism of hiv-1 protease from explicit-solvent atomistic simulations

[25] Vijay S. Pande, Ian Baker, Jarrod Chapman, Sidney P. Elmer, Siraj Khaliq, Stefan M. Larson, Young Min Rhee, Michael R. Shirts, Christopher D. Snow, Eric J. Sorin and Bojan Zagrovic, (2003). Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. Biopolymers, 68:91–109

[26] J.E. Shea and C.L. Brooks, (2001). From folding theories to folding proteins: A review and assessment of simulation studies of protein folding and unfolding. Annu. Rev. Phys. Chem., 52:499–535

[27] Y. Sugita and Y. Okamoto, (1999). Replica-exchange molecular dynamics method for protein folding. Chem. Phys. Lett., 314:141

[28] N. Todorova, F. Marinelli, S. Piana and I. Yarovsky, (2009). Exploring the folding free energy landscape of insulin using bias exchange metadynamics. J. Phys. Chem. B, 113:3556–3564

[29] D. Van der Spoel, E. Lindahl, B. Hess, G. Groenhof, A.E. Mark and H.J.C. Berendsen, (2005). GROMACS: Fast, flexible, and free. J. Comp. Chem., 26(16):1701–1718