1 SIMULATION TASKS SUPPORTED BY IFOLD

1.1 Folding Simulation

Protein folding simulations are performed by starting from a linear conformation of the biomolecule at a high initial temperature and reducing the temperature at the specified rate until the system reaches the final temperature. Folding simulations are among the most useful methods for rapidly characterizing the biophysical properties of proteins. The user uploads the protein's structure file in Protein Databank (PDB) format or specifies the four letter PDB code of the corresponding protein and then selects the following simulation parameters:

(1) Initial temperature – desired temperature from where the simulation is initiated.

(2) Final temperature – final temperature at which the simulation is stopped.

(3) Heat exchange coefficient – rate of heat transfer in the system.

(4) Potential set of folding simulation outputs – trajectory of simulation, graphs and/or data for radius of gyration vs. simulation time, energy vs. simulation time, or energy vs. simulation temperature.

The DMD simulation parameters include initial temperature Ti, final temperature T_f, heat exchange coefficient C, and total simulation time t_{max}. By default $T_i = 1.2$; $T_f = 0.4$, $C = 0.0001$, and total simulation time $t_{max} = 100000$. The advanced user can assign arbitrary values for these parameters.

1.2 Unfolding Simulation

Unfolding simulations correspond to simulating the thermal denaturation of the chosen protein. In unfolding simulations, we start with the native structure of the protein, simulated at low temperature, and slowly raise the temperature of the system until the protein starts to unfold. Unfolding simulations are essentially an inverse operation of folding simulation, except that instead of starting from the linear conformation, we start from the native conformation. The user uploads the protein structure file in Protein Databank (PDB) format or specifies the four letter PDB code of the corresponding protein and then selects the following simulation parameters:

(1) Initial temperature – desired temperature from where the simulation is initiated.

(2) Final temperature – final temperature at which the simulation is stopped.

(3) Heat exchange coefficient – rate of heat transfer in the system.

(4) Potential set of unfolding simulation outputs – trajectory of simulation, graphs and/or data for radius of gyration vs. simulation time, energy vs. simulation time, or energy vs. simulation temperature.

The DMD simulation parameters include initial temperature Ti, final temperature T_f, heat exchange coefficient C, and total simulation time t_{max}. By default $T_i = 0.4$; $T_f = 1.2$, C=0.0001, and total simulation time $t_{max} = 100000$. The advanced user can assign arbitrary values for these parameters.

1.3 Thermodynamic Scan

In thermodynamic scan, we perform constant temperature DMD simulations of the given biomolecule over a range of temperatures to get the thermodynamic properties of the simulated protein. The user uploads the biomolecule's structure file in Protein Databank (PDB) format or specifies the four letter PDB code of the corresponding protein/DNA/RNA and specifies the following set of parameters required for thermodynamic scan simulations:

(1) Range of temperature used for thermodynamic scans. Since the transition temperature is in the range of $T = 0.7$-0.9, a default value of temperature $T = \{0.6, 0.7, 0.8, 0.9, 1.0, 1.1\}$ (temperature in of ε/k_B units of DMD scale) is kept for *guided user mode*.

(2) The desired set of thermodynamic scan outputs – graphs and/or data for variation of heat capacity of the biomolecule vs. simulation temperature, average frequencies of inter-residue or inter-nucleotide contacts made over the given temperature range, energy vs. simulation time, radius of gyration vs. simulation time, or energy vs. simulation temperature.

For each DMD simulation, we set the temperature as constant ($T_i = T_f$), default heat exchange coefficient C = 0.0001, and total simulation time t_{max}= 50000 time units. The advanced user can assign arbitrary values for these parameters.

1.4 Simulated Annealing

In this mode we start from a high temperature conformation where the protein is unfolded and slowly reduce the temperature to a very low value, whereby the protein folds to a stable conformation. This process is repeated a certain number of times, suddenly raising the temperature from the last stable conformation and allowing the protein to cool at a very slow rate. Multiple runs ensure that the protein reaches the most stable conformation - the native state. The user uploads the protein's structure file in Protein Databank (PDB) format

or specifies the four letter PDB code of the corresponding protein and specifies the following set of parameters required for simulated annealing simulations:

(1) Number of runs for simulated annealing -- default value of 5 runs is kept for *guided user mode*.

(2) The desired subset of unfolding simulation outputs – graphs and/or data for variation of energy of the biomolecule vs. simulation temperature, average frequencies of inter-residue or inter-nucleotide contacts made over the given temperature range, energy vs. simulation time, radius of gyration vs. simulation time, or energy vs. simulation temperature.

1.5 Folding Probability Analysis

Folding probability (p_{fold}) refers to the probability likelihood that a given decoy conformation of the protein will fold before unfolding. It is a quantitative measure of progress in protein folding of the given conformation. We have developed a method for performing p_{fold} scan using DMD simulations. A default value of 10 iterations at transition temperature $T = 0.8$ is kept for *guided user mode*. The user uploads the protein's structure file in Protein Databank (PDB) format or specifies the four letter PDB code of the corresponding protein and specifies the following set of parameters required for p_{fold} scan:

(1) The structure of the decoy conformation of the given protein.

(2) The desired set of p_{fold} analysis outputs – the folding probability value (p_{fold}) for the given decoy conformation.

2 DESIGN OF IFOLD SERVER

The design of iFold consists of a PHP front end and a Java back end (which connects to the protein-folding scripts) that communicate through sockets connections. Key functions provided by the system are user registration, job submission and execution, and administrative support. The front end handles all of the administrative and job preparation functions, while the back end is the workhorse that performs and manages the actual simulations. This division of function is important in order to support multiple back end processors and the commensurate workload. The front end is a sophisticated PHP application using advanced technologies such as Smarty templates and AJAX techniques. The backend is a long-running application-independent daemon that receives requests from the front end scheduler and tracks the current state of execution. With these technologies, the entire application is easily adapted to support additional applications.

2.1 User Registration

User registration is a three-step process:

(1) The user fills out the registration form on the main iFold server page, explaining the need for desired computational privileges – regular user/advanced user/administrator -- and submits the registration form.

(2) An email is automatically dispatched to the user's specified email address with a uniquely generated link that the user clicks on to verify the email address.

(3) After validation of the email address, iFold administrators are notified (also via email) to approve the user registration request.

The registration and administrative approval are important safeguards to assure that the user is given an appropriate level of access. Once the iFold administrators approve the user and assign appropriate compute privileges, another email is sent to the user notifying him that he may login to the server and submit folding simulation jobs.

2.2 Job Submission and Execution

Once the user submits the protein folding request, the simulation task is added to the queue of pending simulations, which is implemented on a MySQL database. Administrators are given full management access to this queue and can reprioritize or delete jobs. On the web server, a java-based task scheduler polls the database for queued simulation tasks and checks for daemon availability.

Once a task reaches the top of the queue, the scheduler will process it as soon as there is an available back end processor. The scheduler initiates the execution of the task by sending all information associated with the task (what outputs it needs and what parameters it has) across a socket connection to a daemon application running on a compute node – a machine that contains the protein folding scripts. The daemon application receives the message regarding the task and constructs a shell command to run the appropriate script to handle the requested task. The script runs and generates the requested outputs, which could be graphs of radius of gyration vs. simulation time, energy of simulated protein vs. simulation time, or energy of simulation vs. temperature or movies of simulation trajectories.

These outputs are placed in a temporary directory on the compute node. When the daemon thread finds that there are new outputs, it sends them across a socket connection back to the scheduler on the web server and then deletes them from the compute node. The simulation scheduler receives the outputs and updates the database to reflect that the simulation task has successfully completed, stores the simulation outputs in a directory for the user on the web server machine, and sends an email to the user notifying him that the task is complete -- along with any messages about its status. The user logs back in to the iFold web site, which displays the updated information from the database, and the user can download the outputs from the iFold server.