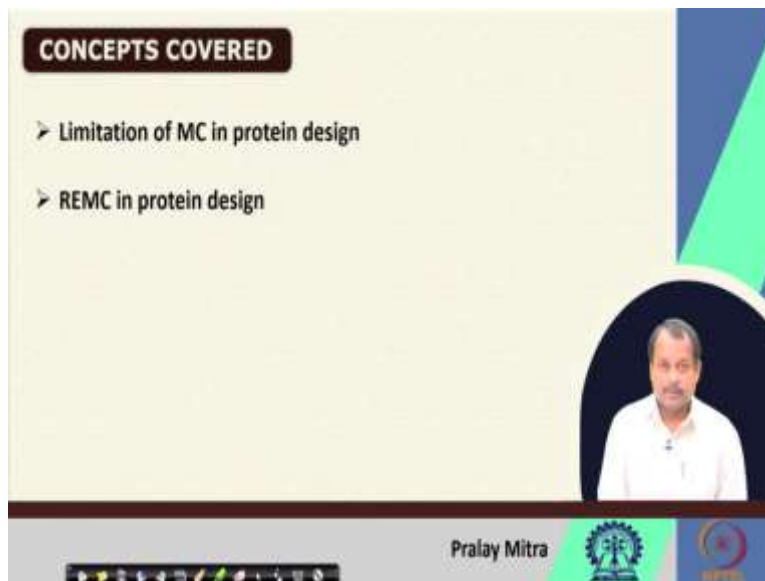


Algorithms for Protein Modelling and Engineering
Professor Pralay Mitra
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 46
RECM in Protein Design

Welcome to the course of Algorithms for Protein Modeling and Engineering. So, we are now in tenth week. So, in this week we will continue with the protein design. So, the protein design we started on the last week specifically we are talking about computational protein design and we also mentioned that computational tools can be considered as an alternative for experimental design, but definitely it is not the final one.

So, our proposal is that we can design one computational protein design framework which can rule out a number of false positive cases and thereby can reduce the search space or can reduce the number of experiments to be performed by the experimentalist. So, in this respect, so, in this week, we will also continue with protein design and then we will move on to other protein engineering topics.

(Refer Slide Time: 01:18)



CONCEPTS COVERED

- Limitation of MC in protein design
- REMC in protein design

Pralay Mitra

KEYWORDS

- Protein Design
- REMC

Pralay Mitra

Pralay Mitra

So, the concepts that we will be covering is limitation of some of MC in protein design and REMC in protein design that is why the keyword also I have chosen as protein design and REMC. Let us rewind and go back to our class on protein folding. During that lecture, I mentioned that the protein energy landscape or folding funnel is very much rugged in nature or in safe.

Now, it can be something like this where I have one defined native state or stable state or it can be something like this so, both are possible based upon what kind of protein it is or the protein it may vary. Now, you see on the left hand side, what I can see that, if I pick a blue color, this is the optimal region or the stable state as per the energy also this is the minimum state.

So, my aim will be to reach at this position when I will reach at this position, then I will understand that I have reached to the stable state and compared to that one I do not find as such any other pocket. So, like this, this, this region or this region, there is no pocket where if I reach then I will have some another instance which will be localized or locally trapped in that region.

But, on the right hand side, the plot indicates that if I assume that this is my folding funnel or energy landscape, then there are several regions like this, this, this, this along with of course, this one which may act as one local stable state. So, definitely this is going to be my native or stable state.

But, if I look at this position or this position, specifically this one then I may misinterpret that perhaps this is my stable state or that is the optimum situation where I wish to reach. So, this kind of situation I also mentioned is called as the local minima or I can also call that as a this one I can call as locally trapped state.

Now, because of this in protein folding, we suggested to move on to replica exchange Monte Carlo rather than the Monte Carlo simulation only. So, in case of protein design also we have the same proposal that when we are designing the algorithm then instead of Monte Carlo better go to replica exchange Monte Carlo.

Although, we noted that Monte Carlo simulation is good enough for a lot of applications, but the primary reason for being good enough for this Monte Carlo simulation is that if I consider the server Evo design that I mentioned on the last slide of the last lecture on the last week, then this Evo design do an intelligent way to get rid of this kind of locally trapped situation what is that, for that let me go to the flow diagram again.

(Refer Slide Time: 06:23)

An alternative - REMC

The diagram illustrates the REMC (Rapid Evolutionary Monte Carlo) process. It starts with a 'Scaffold' protein structure. 'Structural Analogues with TM-score=0.7' are identified. A 'Probe' is then generated from these analogues. This leads to 'Sequence Clustering' based on 'Pairwise Similarity' and 'Scaffold structural information'. The final step is the 'Designed Sequence'.

Pralay Mitra

Handwritten notes detailing the seed-based selection process:

- Seeds: Seed 1, Seed 2, ..., Seed 10
- MC values: $MC_1, MC_2, \dots, MC_{10}$
- Temperature: $e^{-\frac{MC_i}{T}}$
- Accepted Sequences: $\bigcup_{i=1}^{10} MC_i$
- Calculation: $10 \times 30K = 30K$
- Analogs: \downarrow Analogs

REMC protein design

- Decide on number of replicas: 10
- Decide on temperatures of each replica: $0.01 \leftarrow 0.02 \rightarrow 0.09$ (with 10 in a circle)
- Decide on number of local steps: 200 local steps
- Decide on total number of simulation: 30,000

$\frac{30,000}{200} = 150$ exchange
 $\frac{30,000}{200} = 150$ exchange

So, in this flow diagram, when it is performing this Monte Carlo simulation at this position then instead of starting from a random sequence or starting from one random sequence, it goes for 10 parallel simulations. So, what it used to do is so, they are a seed 1, seed 2, seed 10. So, 10 parallel simulations takes place simultaneously.

So, it started that Evo design server as per that algorithm it starts with 10 random sequences here I am denoting them as seed 1, seed 2 up to seed 10 and 10 separate Monte Carlo simulation take plus. So, you can consider MC 1, MC 2 up to MC 10 and after the simulations are over then here it combines all the results.

So, final accepted sequences are union up of i equals to 1 through 10 MC_i so, for each Monte Carlo simulation what are the sequences which are accepted. So, taking the union it got its final result. Now, after that one there will be analysis as we have discussed. So, this modification is done in Evo design and if you do in your protein design algorithm then even using Monte Carlo simulation, there is a possibility that you will get rid of that locally trapped situation.

However, this technique is limited by the fact that all the different 10 simulations run with the same temperature value. So, where this temperature will be utilized, for Metropolis Monte Carlo simulation or for Metropolis criteria, in the metropolis criteria you have to compute $e^{-\frac{\Delta E}{T}}$ where ΔE is the energy difference of the current state and the previous state.

Now, if it is say, higher than a random number that I have generated, then I will accept otherwise I will not accept. But this temperature value is kept as same for all the ten simulations. Also, during the simulation, there is no option of exchanging the information which is general practice or usual practice for replica exchange Monte Carlo.

So, that way also the Evo design or in that algorithm that we demonstrated on the last week, so there is less probability of being locally trapped, but it is not fully exploiting the concept of replica exchange Monte Carlo. Now, in this context, it will be good idea to extend our discussion on the fact that usually in Evo design.

So, what is done that each Monte Carlo simulation runs for 30,000 steps. Now, you see that this number of 30,000 steps maybe good for small protein, but may not be good for larger protein. However, this 30,000 step is a hard coded inside and this is fixed. So, there is no way that we can change this and even if it is, if we change then it will be changed for all the different simulations.

So, again there is no scope that we will dynamically converge when we see that for a long time as such there is no change in the energy. So, this is also interesting to note that say when 30,000 steps are running, so, in total at this union stage so, 10 multiplied with 30k. So, those many number of sequences will be generated by this time and out of this 10 cross 30k few numbers will be accepted.

Now, based upon this temperature what I am deciding if I assume say only 10 percent is being accepted then you can consider safely that 30,000 sequences are accepted. But my point is that why should I go for iteration or simulation in Monte Carlo steps, if I see that my energy is not reducing sufficiently or in general is not changing say for the last 1000 iterations.

So, since each Monte Carlo simulations are running 30,000 steps. So, if I assume that for the last 2500 or say 3000 iterations there is no change in the energy function, then, no change in energy function means the value of the energy corresponding to the newly generated and accepted sequences are not changing, which means it has converged it has reached to some local step which may be local minima or global minima, but it has reached there.

So, simulations are mostly converged why not come out of this state. So, this kind of small, small modifications you can do on the existing protein design algorithm and that we are going to

perform and we will see that that way, we can save a lot of computational time as well as we can have a better solution for our technique.

Nevertheless, when we are talking about the replica exchange Monte Carlo, then we need to fix few things, we need to decide on number of replicas, you remember for protein design, how many number of replicas was there? It was 40. In this case, let us restrict ourselves to say 10 then decide on temperature of each replica.

Now, in this case, if you have decided that your temperature, so, since 10 replicas are there then it should be uniformly distributed among the replicas. So, if in your previous example, you considered some temperature say 0.3 then with the same dimension, you can extend further.

So, maybe 0.1 to say 0.9 and 1 replica with very high temperatures say 3 or 4, which may reduce the bias or if the system goes bad, during the simulation, then it can come out of that state. If you consider 0.1 and 0.03, then it will go 0.01 it can you can reduce to say instead of 3.0, 1.0. So, 1 very high temperature may be there.

So, that if the system goes bad in the sense that it saturates, et cetera. Then to come out of that, it may be useful for you then decide on number of local steps. You remember that when I am running say replica exchange Monte Carlo which means that I am running Monte Carlo simulation and after a certain number of execution of the Monte Carlo simulation, then I am comparing between two neighbor replicas say if my replica as 1 2 3 4 5 6 7 8 9 10 then after a fixed number of Monte Carlo simulation say if I assume that my number of fixed number of Monte Carlo simulation will be 100 so, after 100 steps.

So, it will synchronize I mean all the 10 replicas will synchronize then they will compare their energy value with the neighbor one and if it feels that exchange is required because as per the temperature if it is expected that its energy should be less compared to the previous one, actually it is not then exchange the replica.

So, exchanging either can be done by changing the temperature or by changing the energy function as well as the sequence, second one is preferable and most of the time it is done with that one. Now, that before the exchange how many local steps it will run for that also you need to decide.

So, let us decide 200 numbers of such local steps are required. Next you decide on total number of simulations. So, last time as I mentioned that it was hard coded and usually 30,000 steps are used to run for 10 parallel Monte Carlo simulations. So, in total I am supposed to explore 3 lakhs or yes 3000 multiplied with 10.

So, 3 lakhs or 300,000 number of sequences or I wish to explore. Now, in this case, since 200 local steps are there if I assume 10 again the same so the 10 was there now also it is 10, 200. So, accordingly for the time being without any loss of generality you can assume that same 30,000 steps will be in execution.

Now, if 30,000 steps will be in execution then 30,000 divided by 200 that is my local steps. So, it is going to be 150 exchange is required. So, we have to set up my simulation accordingly. So, these are some values, say 10 number of replicas, this is the temperature 200 local steps and 30,000 steps, but definitely you have the freedom to customize this one for your own purpose and also we will see that the dynamic convergence will be good enough instead of these always 30,000 run irrespective of what is the size of the protein or whether it is converging or not.

(Refer Slide Time: 18:39)

REMC protein design

Scope for dynamic convergence

$\text{if } (\Delta E < 0)$
accept

else $q \leftarrow U[0, 1]$
 $\text{if } (q > e^{-\frac{\Delta E}{T}})$
accept

Pralay Mitra

REMC protein design

Algorithm 15:
A REMC protein design algorithm with dynamic convergence

Pralay Mitra

So, here is the scope of dynamic convergence. So, scope of the dynamic convergence as I was explaining is that when say one simulation is in execution either in Monte Carlo simulation or say Monte Carlo executing 10 different simulations in parallel or it is replica exchange Monte Carlo.

So, in all the cases, it is straightforward way to run the execution for say 30,000 steps including the local steps. So, I mean that 200 if I consider 200 local steps, so, 100, 50 times 200 local step in total 30,000 steps you can run but at the same time, if you look at the energy, so, you remember that what is the rule for our acceptance of the state that it says that if.

So, $\Delta E < 0$ then accept else. So, I generate one random number in between 0 and 1 and if $q > e^{-\Delta E / T}$ then accept. Now in this case in this else part I am ensuring that higher energy state will be accepted, but in this case it will be accepted only if in the current state the energy of the sequence is lower compared to its previous stage.

So, if it is lower compared to it is previous stage then energy will keep on gradually decreasing. Now, if I keep a track of the trajectory or the flow of my energy or flow of the energy of the accepted sequences, then it may possible during the simulation I will note that say after certain states say I assume that 30,000 steps will be in execution.

But it may possible, say after 10,000 I noted that there is a such no change in the energy, no change means change towards the lower state. So, it is fine that if it goes to the upper state and

after reaching to the upper state then it will come down but if it is the case that I am not moving to any new low energy state then the question arises is it worth of continuing the simulation.

Now, for the Monte Carlo simulation definitely no but for replica exchange it might be because after the exchanging the replica since the temperature value will be changed for one particular say state then it might be possible that there will be a sudden jump through this Metropolis criteria jump means energy will increase and after that one it will reach or arrive to a new local minima or global minima.

If it is the case then I can allow and that is the whole purpose of the replica exchange Monte Carlo. But for the only Monte Carlo so, it may not be possible that if I see that for 1000 steps or 2000 steps energy is not increasing the minimum energy has become stable. So, it is a better idea to abort the simulation or to come out of the simulation and declare that, I am converged.

So, whatever needs to be explored has been explored let us go and do the analysis, with this one. So, with this dynamic convergence scope, we are integrating our total method. So, what we are now going to do, we are going to move from Monte Carlo simulation to replica exchange Monte Carlo that is why we decided that say let us 10 replicas will be there.

But definitely users should have a control on how many number of replicas he can decide then what will be the temperature, minimum, maximum what will be the range so, that thing also I will decide, user can give an input on that one then how many local steps or Monte Carlo simulation it will run that I can decide.

And finally, for how many number of steps if the energy is not going to a new minima then I will declared that I have reached. So, come out and that way definitely I can save some computation time and for which I am not achieving anything new. So, combining all those things, we are now ready to move on to new algorithm for REMC protein design with dynamic convergence.

(Refer Slide Time: 24:05)

```

Algorithm 1: GreedyREMCDynConv( $nstep, max\_iter, max\_step$ )
1 for each simulation trajectory  $S_i, (i \in [1, 10])$  do
2   Initialize  $steps_i \leftarrow 1, best\_steps_i \leftarrow 1, best_g \leftarrow Infinity$ 
3   Randomly Initialize decay sequence  $D_i^{steps_i}$ 
4    $D_i = D_i^{steps_i}$ 
5 for each  $S_i$  do
6    $present\_D_i \leftarrow D_i^{steps_i}$ 
7    $TD \leftarrow RandomMutation(present\_D_i)$  // the TD list contains  $nstep_{max}$  temporary decays
8    $(temp_D, nstep) \leftarrow GreedyMMC(present\_D_i, TD, nstep_{max})$ 
9   if  $nstep > 1$  then
10    for  $j = 1$  to  $nstep - 1$  do
11       $steps_i \leftarrow steps_i + 1$ 
12       $D_i^{steps_i} \leftarrow present\_D_i$ 
13       $D_i \leftarrow D_i \cup D_i^{steps_i}$ 
14
15     $steps_i \leftarrow steps_i + 1$ 
16     $D_i^{steps_i} \leftarrow temp_D$ 
17     $D_i \leftarrow D_i \cup D_i^{steps_i}$ 
18     $(best\_steps_i, best_g) \leftarrow Update(best\_steps_i, best_g)$ 
19    if  $((steps_i >= best\_steps_i + max\_steps) or (steps_i = MAX\_STEP))$  then
20      return  $D_i$ 
21
22    if  $((steps_i \bmod 200 = 0) and (steps_i = steps_{i+1}))$  then
23      if  $((S_{i+1}$  is active) and  $((I_{S_i} - I_{S_{i+1}})(E_{MC}(D_i^{steps_i}) - E_{MC}(D_{i+1}^{steps_{i+1}})) < 0)$  then
24        Exchange( $D_i^{steps_i}, D_{i+1}^{steps_{i+1}}$ )

```

So, this algorithm as expected is starting for each simulation trajectory S_i , i belongs to 1 through 10 do initialize steps S_i . So, this is my initialization stage and in this initialization stage, so, what I am doing that I am generating new sequences that is going to be my said sequences and feeding that one and also I am assuming this is my best.

So, you can accept that is the first one after that one. So, for each replica what I am doing, so I am doing a random mutation on the new sequence. Then I am calling one function greedy Metropolis Monte Carlo. So, what is Greedy Metropolis Monte Carlo I will show in the next slide and if number of steps greater than 1 for j equals to 1 to number of step minus 1 do steps equals to steps plus 1.

So, I am increasing the number of steps. Now, after increasing the number of steps, so, the best result is being updated here and if the simulation reaches to maximum number of iterations or its reaches to max steps, then I will return. So, I will converge, otherwise, if steps mod 200 that I reach to local Monte Carlo simulation steps 200 then also what I will do, I will check that whether any exchange is required or not.

So, this exchange is the replica exchange. So, if I will reach to 200 steps locally, so, that I am checking with the mod 200. So, if I reach to this 200 step then I will check whether the exchange of the replica is required or not, if yes, I will perform using this function call, otherwise I will not. But before that here what I am checking?

I am checking for the dynamic convergence, if max number of steps has been mentioned, let us assume this is my 3000 steps that I have mentioned. Then I will come out naturally because I mentioned that 30,000 steps are required but at the same time if it is best steps plus max iteration so, for the last max iteration number of steps, there is no change in the energy, no change means that change may be there in order to increase the energy since I am using Metropolis criteria.

But new minimum energy is not achieved for the last max iterations. If it is then I will also return. So, there are two exit criteria or return condition 1. So, when I will reach to 30,000 step let us assume that I mention maximum number of simulation that I will perform is 30,000 that is the maximum otherwise I will exit if I see that for the last max iteration number of steps new minimum energy is not coming.

Now, this max iteration or MAX STEP a user can set say let us assume for max step it is 30,000 and max iteration you can say consider 2000. So, the rest of the parts are simple and is similar to replica exchange Monte Carlo. Now, let us see what is there in greedy MMC in the next slide.

(Refer Slide Time: 27:51)

```

Procedure GreedyMMC( $d, TD, m\_nstep$ )
1 Initialize  $best_{diff} \leftarrow 0, nstep \leftarrow m\_nstep, metro \leftarrow 0, temp_D \leftarrow d$ 
2 for  $k = 1$  to  $m\_nstep$  do
3    $E_{diff} \leftarrow E_{MC}(d) - E_{MC}(TD_k)$ 
4   if  $E_{diff} > best_{diff}$  then
5      $temp_D \leftarrow TD_k, best_{diff} \leftarrow E_{diff}, nstep \leftarrow 1, metro \leftarrow 1$ 
4   else
7     if  $((rand < e^{-\frac{E_{diff}}{k_B T}}) \text{ and } (metro = 0))$  then
8        $temp_D \leftarrow TD_k, nstep \leftarrow k, metro \leftarrow 1$ 
4   return ( $temp_D, nstep$ )
  
```

Pralay Mitra

A parallel REMC protein design

Algorithm 16:

Pralay Mitra

So, the procedure greedy MMC it says that, it will take small d , TD m number of steps, initialized best difference is 0 number of steps m underscore n step and metro 0 tempD is initialized to d . Now, for k equals to 1 to m n steps so, local number of steps what I am doing, I am computing the energy difference $EMC_d - EMC_{TDk}$.

So, what was the minimum energy with that one I am computing my new energy. So, after this difference if it is greater than best then it is fine, otherwise, so, it is best means it will be accepted here otherwise using this Metropolis Monte Carlo technique I will accept this. So, two acceptance criteria, one is if I get a minimum energy compared to the previous state then I will accept, otherwise I will accept based upon the fact that I got Metropolis criteria as true.

So, this is in summary our replica exchange Monte Carlo simulation with greedy approach also I would like to mention that there is a possibility that this replica exchange Monte Carlo technique can be implemented in parallel. So, as you understand by this time that I am running different replicas, say replica 1, 2, 3, 4 and inside that one, there is a local step say 200.

Now, during this local step, nobody is exchanging any information. So, I may safely assume that if I have n number of replicas then I can allocate them in n number of processors or in shared memory programming n number of threads, then each will run on each thread and after 200 local step they will synchronize, they will check for the exchange of the information.

In this case exchange of information means whether replicas will be exchanged or not and after that will take place then again it will go for parallel simulations and that way it has been demonstrated that if you exploit say n number of processors then there is a possibility that you can gain so, at least n times the speed up.

So, on a 6 core system when it was tested, then it was giving, about 6.21 times more speed compared to a serial program which is running on only one thread or one processor. So, that is it in this lecture. We will continue this in the next lecture. Thank you.