IMPROVING SEARCH FOR LOW ENERGY PROTEIN STRUCTURES WITH AN ITERATIVE NICHE GENETIC ALGORITHM

Glennie Helles
University of Copenhagen, Department of Computer Science, Universitetsparken 1, 2100 Copenhagen, Denmark

Keywords: Protein structure prediction, Parallelism, Genetic algorithm, Parallel tempering.

Abstract: In attempts to predict the tertiary structure of proteins we use almost exclusively metaheuristics. However, despite known differences in performance of metaheuristics for different problems, the effect of the choice of metaheuristic has received precious little attention in this field. Particularly parallel implementations have been demonstrated to generally outperform their sequential counterparts, but they are nevertheless used to a much lesser extent for protein structure prediction. In this work we focus strictly on parallel algorithms for protein structure prediction and propose a parallel algorithm, which adds an iterative layer to the traditional niche genetic algorithm. We implement both the traditional niche genetic algorithm and the parallel tempering algorithm in a fashion that allows us to compare the algorithms and look at how they differ in performance. The results show that the iterative niche algorithm converges much faster at lower energy structures than both the traditional niche genetic algorithm and the parallel tempering algorithm.

1 INTRODUCTION

Metaheuristics are known to perform well on high-complexity problems where the search space becomes too big for exhaustive search to be feasible. Prediction of the three-dimensional structure of proteins from their primary sequence alone, known as *ab initio* or textitde novo folding[1], is such a problem and metaheuristics are almost exclusively used to solve this problem (Helles, 2008; Oakley et al., 2008).

Proteins are made up by amino acids that are strung together like pearls on a string such that each amino acid is connected to its neighboring amino acid(s) via a peptide bond, ω. However, despite the rigid nature of the peptide bond, atoms can in theory rotate almost freely around the two other backbone bonds – the N–C_α bond, ϕ, and the C_α–C' bond, ψ – which means that just like a pearl necklace, a protein can be folded up in infinitely many ways, which is the reason that protein structure prediction poses such a big problem. Fortunately, steric clashes between atoms in neighboring amino acids do impose a significant restraint on the flexibility of the ϕ and ψ angles actually observed for amino acids (Ramachandran and Sasisekharan, 1968), but searching exhaustively for the structure with the lowest energy remains elusive.

Judging from the literature, the Monte Carlo variant known as Simulated Annealing appear to be the preferred meta-heuristic for *ab initio* structure prediction followed by Genetic Algorithms (Helles, 2008). Both metaheuristics can be parallelized, and the parallel versions are generally believed to perform better in rugged energy landscapes like those associated with protein structure prediction (Earlab and Deem, 2005). Oddly enough, the parallel versions are nevertheless used to a much lesser extent than their sequential counterparts in the field of protein structure prediction. We speculate that that is mostly because the effect of the choice of metaheuristics has been paid little attention in this field that is notoriously haunted by many other fundamental problems. The most significant obstacle probably being finding an appropriate energy function that can be used to score a protein,

[1]The term *ab initio* traditionally refers to prediction methods that start without any knowledge of any globally similar folds thereby setting them aside from homology modeling techniques. However, many so called *ab initio* methods do in fact use secondary structure prediction algorithms that are trained from knowledge of already known structures, or they use fragment assembly compiled from known structures. Some choose to refer to this as *de novo* prediction rather than *ab initio* prediction. The term *ab initio* will be used in this publication

Helles G. (2010).
IMPROVING SEARCH FOR LOW ENERGY PROTEIN STRUCTURES WITH AN ITERATIVE NICHE GENETIC ALGORITHM.
In *Proceedings of the First International Conference on Bioinformatics*, pages 226-232
DOI: 10.5220/0002743702260232

which has by far received the most attention over the years.

To our knowledge there exists only one parallel version of the Simulated annealing algorithm known as the Parallel Tempering or Monte Carlo Replica Exchange algorithm (Swendsen and Wang, 1986; Earlab and Deem, 2005). In Parallel tempering (PT) many simulations, or replicas, are started and run in parallel. The solutions are sampled in the same fashion as in the regular simulated annealing approach by making small alterations to the solution and accepting the change with a certain probability. However, instead of lowering the temperature like in the simulated annealing approach the simulations are run at different but steady temperatures throughout the search. Two replicas may be swapped with a probability that depends on both differences in energy and temperatures, such that a replica running at a lower temperature can be exchanged with a replica running at a higher temperature, thereby giving replicas a greater chance of overcoming local minima barriers.

For genetic algorithms there exists several parallel variants that generally offer significant improvements by converging faster at often better solutions than the non-parallel version. There are two major approaches to parallelizing genetic algorithms. One is often referred to as the master-slave model, where a single process (the master) controls the genetic algorithm, but uses a number of other processes (the slaves) to evaluate and possible breed the individuals. The slave processes are run in parallel.

The other parallelization paradigm frequently used is the niche model (also known as the island hopping or deme model). A niche genetic algorithm (nGA) is an implementation where several instances of a genetic algorithm are run in parallel, evolving sub-populations independently from each other (the different niches). At certain points during evolution individuals migrate to other niches and become part of the population of that niche. The major advantage of nGAs is that they not only allow evolvement of multiple solutions at the same time, they exploit the fact that different runs of the same genetic algorithm is likely to produce different suboptimal solutions, that combined are likely to yield better results. Like PT the advantage of nGA is expected to be more profound when the fitness landscape is very rugged.

PT and nGA, with N replicas and niches respectively, essentially requires N times more computational time than a single run of their sequential counterparts. However, with multi-core computers and CPU clusters being readily available to most researchers, they can be executed in parallel and the extra computational time required does thus not impose a problem. On top of that, PT and nGA generally search more efficiently and usually arrive at much better results, which makes the parallelization of these meta-heuristics an attractive feature indeed.

In this paper we propose an iterative variant of a nGA, called inGA (iterative niche genetic algorithm) for protein structure prediction. The algorithm is designed to increase search efficiency by locating and converging on the low energy structures faster than both nGA and PT. Essentially, the strategy corresponds to letting all niches converge before migrating individuals between them and restarting as described by Cantu-Paz and Goldberg (Cant-Paz and Goldberg, 1996). However, while running each niche to convergence worked well for the problem instances chosen by Cantu-Paz and Goldberg, work by Heiler (Heiler, 1998) suggest that for protein structure prediction the quality of predicted structures *decrease* when the individuals are locally optimized before the genetic operators are applied. Rather than running to convergence we thus suggest a kind of early stopping, which generate low energy structures without spending too much time on refining suboptimal structures.

2 METHODS

In the traditional niche genetic algorithm (nGA), evolvement of several populations are run in parallel and completely independent from each other. At certain points individuals from one or more niches (islands or demes) are chosen according to some selection strategy and migrated to other niches, where they replace individuals also chosen according to some selection strategy. Usually the selection strategies are based on the fitness values of the individuals such that the best individuals from one niche are migrated to another niche where they replace the worst individuals, as this migration strategy yields the fastest convergence (Alba, 2005).

We propose an iterative niche genetic algorithm (inGA) that performs a type of elitist refinement. Like the traditional niche genetic algorithm multiple populations are initially created and evolved in parallel, but unlike the traditional niche algorithms, individuals do not migrate to other niches. Rather we stop evolvement of all populations after a predefined number of generations, g, and choose the best solution from each of the n niches. The individuals not selected are destroyed while the selected individual are put together in a new population, pop. pop is then cloned n times and the cloned populations are placed on the n niches where evolvement of new (and initially identical) populations is then carried out for g

generations. The procedure of stopping, selecting, cloning and restarting is repeated until the algorithm converges. The pseudo code for the algorithm is given in Algorithm 1

Algorithm 1. Pseudo code for inGA.

```
niches ← CREATE_THREADS(n)
pop ← NULL
while !DONE() do
  for each n in niches do
    if pop equals NULL then
      population ← CREATE_POPULATION()
    else
      population ← CLONE(pop)
    end if
    niches[n] ← start(GA(population, g))
  end for
  WAIT_FOR_COMPLETION(niches)
  pop ← NULL
  for each n in niches do
    pop ← GET_BEST_INDIVIDUAL(niches[n])
  end for
end while
```

This strategy requires the number of individuals in each niche to be the same as the number of niches, although a different strategy could, of course, also be utilized. In this work we ran 20 parallel niches with 20 individuals in each niche.

We did preliminary experiments to determine how many generations to run the niches in each iteration. We wanted the GA to run just long enough to reach a good solution that captured the best traits of that niche and by analyzing the development of the energy we found that by far the largest improvements happen during the first 100 generations. We thus settled at running 100 generations per iteration. We did try to run the GA for 200 generations, but found that while the initial niche solutions were improved, the final result was not, which is much in keeping with the findings presented in (Heiler, 1998).

The selection strategy employed both an elitism strategy and the fitness proportionate selection strategy known as a roulette wheel. The elitism strategy clones and transfers the 10% top scoring individuals unaltered to the next generation thereby ensuring that the best individuals are always kept. However, the 10% best individuals are also allowed to compete in the roulette wheel selection, where each individual is chosen with a probability corresponding to its fitness value. This strategy is chosen over rank selection to ensure a better chance for low scoring individuals to be selected.

Individuals selected by the roulette wheel are subjected to crossover and mutation. A multi-point crossover strategy is used where the number of crossover sites, *c*, are chosen according to a Gaussian distribution. The *c* actual crossover sites are chosen at random. The advantage of multi-point crossover over single point crossover is that it eliminates the bias of the end segments that is commonly raised as an issue with the vector representation employed by most genetic algorithms. Also multi-point crossover typically results in bigger alterations of the solutions causing the genetic algorithm to explore very different regions of the search space.

As is often the choice, the mutation rate is set fairly low to a value of 0.001. Mutation is thus not the driving force in the folding process, but is used mainly as a way to introduce new genes into the existing gene pool.

It should be noted that setting the hyper-parameters of genetic algorithms (such as selection and recombination strategies) is an optimization problem in itself. We have looked to the literature for inspiration and carried out numerous experiments with different combinations before settling on the ones described here.

2.1 Encoding

In this work we use a physics based energy potentials, called POISE (Lin et al., 2007), which requires a full atom model. However, rather than searching the Cartesian space, only dihedral angles, bond angles and bond lengths (referred to as the set of structural variables) are explicitly represented as atom positions can be calculated directly from these using standard matrix operations. A protein is thus encoded as a vector (chromosome) of $S_{structvar}$, where each $S_{structvar}$ represents the structural variable of one amino acid.

One of the problems often encountered during encoding of proteins is the occurrence of clashing atoms. When two atoms come within very close proximity, the laws of physic dictate that the energy will rapidly grow towards infinity and the atoms will be forced apart. When using a full atom model one of two strategies can be utilized; either one can explicitly check and make sure that atoms do not clash or clash are tolerated, but heavily penalized by the energy function, such that solutions with clashing atoms stand little chance of being accepted/selected. The latter works best in conjunction with statistics based energy function where the 'energy' term is usually an artificial pseudo energy made up from many parameters that are non-numerical in nature. With a pure physics based energy potential, the infinitely large increase in energy caused by two clashing atoms will

cause overflow on a computer. As we utilize a pure physics based potential, a check for clashing atoms is thus carried out, before a solution is evaluated and only clash-free structures are accepted.

The protein is encoded sequentially and for every new residue added we check whether the $S_{structvar}$ chosen causes any atoms of the new amino acid to clash with atoms in the residues that have already been added. If a clash occur a new $S_{structvar}$ for the amino acid is chosen. If the problem with atom clash has not been resolved after 20 different $S_{structvar}$ have been tried for the amino acid, we backtrack and choose a new $S_{structvar}$ for the previous residue. Although the theoretical running time for this strategy is $O(2^n)$ the running time was not found to be an issue in practice.

2.2 Move Set

The move set is defined as the set of possible combinations of bond lengths, angles and dihedral angles for each amino acid. Theoretically, the move set is unrestricted, but in practice we know that bond lengths and angles vary very little and dihedral angles are heavily biased towards certain areas of the dihedral angle space. Here, we thus choose bond angles and bond lengths for amino acids randomly from within a small interval (up to ± 0.1) of the optimal angles and lengths as defined in the AMBER 99 parameters.

The dihedral angles space is likewise restricted. In Fonseca and Helles (Fonseca and Helles, 2009) a probability distribution is predicted for each amino acid in a sequence by considering the neighboring amino acids. This probability distribution is used here such that the dihedral angles are chosen from this sequence-dependent distribution thereby maximizing the probability of sampling a realistic area of the dihedral angle space.

2.3 Energy Function

Generally speaking, energy functions can be divided into two categories: physics based energy function and statistics based energy functions. The energy function used here, called POISE, is a purely physics based potential, described in more details in (Lin et al., 2007). It combines the AMBER force field (Swendsen and Wang, 1986):

$$E_{\text{protein}} = \sum_{i}^{bonds} K_b(b_i - b_0)^2 + \sum_{i}^{angles} K_\theta(\theta_i - \theta_0)^2 + \sum_{i}^{dihedrals} k_x[1+\cos(n\phi-\gamma)] + \sum_{i}^{N}\sum_{i<j}^{N}\left(\frac{q_i q_j}{r_{ij}} + 4\varepsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right]\right) \quad (1)$$

with a Generalized Born component:

$$V_{\text{GB}} = -\frac{1}{2}\left(\frac{1}{\varepsilon_p} - \frac{1}{\varepsilon_w}\right)\sum_i\sum_{i<j}\frac{q_i q_j}{f_{ij}^{GB}(r_{ij})} \quad (2)$$

$$f_{\text{ij}}^{GB}(r_{ij}) = \left[r_{ij}^2 + R_i R_j exp\left(\frac{-r_{ij(2)}}{4R_i R_j}\right)\right]^{\frac{1}{2}}$$

and a hydrophobic mean force potential:

$$V_{\text{HMFP}} = \sum_{i\in SA_i>A_c}^{N_c} tanh(SA_i) \sum_{j\in SA_j>A_c, j\neq i}^{N_c} tanh(SA_j) \; x\sum_{k=1}^{3} h_k exp\left(-\left[\frac{r_{ij}-c_k}{w_k}\right]^2\right) \quad (3)$$

We refer to (Lin et al., 2007) for an explanation of the parameters.

The potential considers interatomic energies between all pairs of atoms and the time complexity of calculating the potential is thus quadratic ($O(n^2)$). Fortunately, the vast majority of atoms in a protein are simply too far apart to exert any power on each other and by simply omitting calculations of these interactions, the function is implemented such that the potential is calculated in linear time.

It should be noted, that in this current work we are primarily concerned with testing the performance of different parallel meta-heuristics. We have chosen the POISE energy function because it gives a realistic impression of the very rugged fitness landscape that the algorithms have to navigate around

2.4 Test Proteins

A standardized, diverse set of proteins that is guaranteed to provide an adequate representation of protein structures does unfortunately not exist. The test set used here is constructed such that 2 proteins from each of the categories α, β and $\alpha\beta$ have been picked from the PDB Select 25% [2]. Only small to medium sized proteins have been included. The smallest protein includes 46 amino acids and the largest protein includes 81 amino acids (Table 1).

[2] http://bioinfo.tg.fh-giessen.de/pdbselect/recent.pdb_select25

Table 1: Test proteins.

	Category	Residues
1C75	α	71
1NKD	α	59
1YK4	β	52
2O9S	β	67
1EJG	αβ	46
1IQZ	αβ	81

2.5 Benchmarking Algorithms

In order to evaluate the performance of inGA we implemented both the traditional nGA and the parallel version of SA, called parallel tempering (also known as the Replica Exchange Monte Carlo algorithm). Parallel algorithms have numerous times been reported to outperform the sequential algorithms, and here we thus focus only on parallel algorithms [3].

The nGA use the same GA to evolve populations as in inGA, and like for inGA we also used 20 parallel niches in nGA. The convergence rate of a nGA is strongly affected by the migration scheme (Alba, 2005). Migrating and replacing only a few randomly chosen individuals leads to very slow convergence whereas migrating the best and replacing the worst leads to the fastest convergence. To make the comparison with inGA, which is highly elitist, fair, we employed a rather strong selection scheme such that every 100 generation we chose the 50% best individuals from each population, cloned them and migrated them to another niche where it replaced the 50% worst individuals.

The PT algorithm is implemented such that it utilizes the same encoding strategy, energy function and move set as described above in order to ensure comparability. The lowest and highest temperatures were determined in the way proposed in (Sanvicente-Snchez and Frausto-Sols, 2004), such that

$$c_{highest} = -\delta E_{max}/ln(P^A(\delta E_{max})) \quad (4)$$

and

$$c_{lowest} = -\delta E_{min}/ln(P^A(\delta E_{min})) \quad (5)$$

Initial experiments measuring differences in the energy, E, between neighboring structures were run to determine the values of E_{max} and E_{min}. $P^A(\delta E_{max})$ and $P^A(\delta E_{min})$ were set to 0.95 and 0.05 respectively. This resulted in $c_{highest}$ = 3800 and c_{lowest} = 10 with temperatures of the different replicas spaced according to:

$$temp\_replica_{i+1} = 10 * i * 2 + temp\_replica_i \quad (6)$$

The observed average probability of accepting a swapping move between neighboring replicas was roughly 20% in accordance with (Kone and Kofke, 2005), but with swapping of course occurring much more frequently between replicas running at high temperatures and much less frequently between replicas running at low temperatures.

In each time step every replica goes through N moves for a N-residue long protein. A move consists of randomly selecting an amino acid and picking a new $S_{structvar}$ for that amino acid. As is typical for the simulated annealing approach a move from structure s to some neighbor s' is accepted with the following probabilities:

$$P(accept) = \begin{cases} exp^{-(E_{s'}-E_s)/K_bT} & E_{s'} \geq E_s \\ 1 & E_{s'} < E_s \end{cases} \quad (7)$$

where E is the energy of the structure and T is the temperature.

We ran 20 parallel simulations. The probability of accepting a swap between to replicas, i and j, was given by:

$$P(i \leftrightarrow j) = min\{1, exp^{[-(\beta_i-\beta_j)(E_j-E_i)]}\} \quad (8)$$

where β is the inverse temperature $\beta = 1/k_BT$ and $\beta_i > \beta_j$. Defining the probability of swapping replicas such that it decreases exponentially as the gap between temperature increase is usually employed in PT (Earlab and Deem, 2005) and also the reason why it was chosen here.

3 RESULTS AND DISCUSSION

Early results from experimentation with the three different parallelization schemes on the test proteins, shown in Table 2, look very promising with inGA quickly and consistently locating structures of lower energy than both nGA and PT. Please note that we have not calculated RMSD of the final structures, because as such the search algorithms are all oblivious to the concept of a native structures. They seek merely to minimize energy as specified by the POISE potential and in this experiment we are only interested in determining how efficient the different algorithms are in finding low energy structures in the highly rugged energy landscape associated with protein structure prediction energy functions. A different energy function would most likely lead to different (either better or worse) quality of the final structures in terms of RMSD to the native structure, but the differences in how well the algorithms perform with

[3] Although the results are not reported here, experiments with both the standard non-parallel genetic algorithm and simulated annealing was carried out as well, and they did indeed perform worse than the parallel versions

Table 2: Early results of the three algorithms. Energies are calculated with the POISE potential. Lower energies are better.

	nGA	inGA	PT
1C75	319	**287**	513
1NKD	113	**-14**	200
1YK4	216	**142**	275
2O9S	385	**223**	875
1EJG	4	**-7**	30
1IQZ	414	**348**	665

respect to each other would (expectedly) remain the same.

Given unlimited time, all meta-heuristics would probably find the same low energy structures. Unfortunately, time is usually not unlimited in practice and designing search algorithms that increases search efficiency such that we can obtain better results faster becomes important. Parallelization has in itself increased search efficiency, but from the results presented here it is evident that how the algorithms are parallelized can also have a profound impact on how efficiently the algorithms travel the energy surface in their search for the global minima.

The solution space for a given protein sequence is infinitely big and the key to success for a meta-heuristics is usually a good balance between exploration and exploitation. Minima should be explored thoroughly while still allowing the algorithm to move relatively freely across energy barriers. PT, nGA and inGA all differ from each other in this exploration-exploitation balance.

In PT the balance between exploration and exploitation is kept by running parallel simulations at different temperatures. The advantage of PT is that it can be run for exactly as long as time permits, because while it may settle at a minima, it does not really converge but rather keep exploring for a preset number of iterations or until it is interrupted. As such the PT algorithm enjoys the same theoretical guarantee of finding the global minima as the simulated annealing algorithm. However, while parallel tempering reaches low energy structures faster than sequential Monte Carlo simulations (Earlab and Deem, 2005), the number of replicas used depend not so much on available processors, but on what makes sense in order to maintain proper communication between the different replicas. In other words, there appears to be an upper limit on what we can expect to gain in performance that depend on the problem and not on CPU power. For proteins of the length used here, 20 replicas ensure appropriate communication across temperatures, and more replicas would thus only increase the level of communication thereby setting off the exploration-exploitation balance, which would not be desirable. Of course, for larger proteins where the energy span between different structures is likely to be greater than for the proteins used here, more replicas would most likely be required to ensure proper communication.

One of the reasons why PT does not reach the low energy structures as fast as the genetic algorithms is that although many replicas are run at the same time they do not exchange information between replicas. If a good solution is encountered at one temperature it may be exploited by swapping it to a lower temperature, but it does not share its favorable characteristics with any of the other replicas. Parallel tempering would most likely reach the same results as achieved by inGA, but we postulate that because of the lack of information sharing it can generally be assumed to take much longer.

The genetic algorithms on the other hand have a high degree of information sharing via their crossover operator, which explains why the genetic algorithms reach the lower energy structures much quicker. The migration scheme we have used here for nGA is fairly aggressive to ensure faster convergence that would be comparable with inGA. From the results it is evident that while nGA finds lower energy structures than PT it does not reach structures with energies as low as inGA. It should also be noted that despite the aggressive migration strategy, nGA does not fully converge within 20 iterations for any of the test proteins, although signs of convergence is beginning to show. We did initially experiment with a less aggressive migration scheme (that migrated only the best individual), but energies were significantly worse after the 20 iterations than with the chosen migration scheme and it did of course not come near convergence within 20 iterations.

An issues with inGA is that it may simply converge prematurely. The iterative strategy of inGA is highly elitist and with 20 niches it usually converges fully within 20 iterations for the small to medium sized proteins used here. An elitist strategy (always picking the best) favors exploitation heavily and will normally only work well in smooth energy landscapes. The energy landscape of proteins is obviously anything but smooth, but interestingly a balance with exploration does nevertheless appear to be maintained in inGA by the niche approach. Exploration can thus be controlled by simply adding more or less niches. This is indeed a nice feature, since better performance can then be brought to depend more on available CPUs rather than on available time. Obviously, adding more niches would most likely require more iterations to fully converge, but the number of it-

erations required to converge would expectedly grow much slower for inGA than for nGA thereby making the difference in performance between inGA and nGA greater as the number of niches increase.

4 CONCLUSIONS

We presented an iterative variant of the parallel niche genetic algorithm for protein structure prediction. Early results show that the algorithm finds significantly lower energy structures than both the traditional niche genetic algorithm and the parallel tempering algorithm within comparable time. The algorithm converges quickly and the exploration-exploitation balance can be controlled with the number of niches included, which means that search efficiency can be expected to scale nicely with the number of available CPUs.

REFERENCES

Alba, E. (2005). *Parallel Metaheuristics*. Wiley.

Cant-Paz, E. and Goldberg, D. E. (1996). Modeling idealized bounding cases of parallel genetic algorithms. In *In*, pages 353–361. Morgan Kaufmann Publishers.

Earlab, D. J. and Deem, M. W. (2005). Parallel tempering: Theory, applications, and new perspectives. *Phys. Chem. Chem. Phys.*, 7:3910.

Fonseca, R. and Helles, G. (2009). Predicting dihedral angle probability distributions for protein coil residues from primary sequence using neural networks. *In submission with BMC Bioinformatics*.

Heiler, M. (1998). Massively parallel gas for protein structure.

Helles, G. (2008). A comparative study of the reported performance of *Ab Initio* protein structure prediction algorithms. *J. R. Soc. Interface*, 5:387396.

Kone, A. and Kofke, D. A. (2005). Selection of temperature intervals for parallel-tempering simulations. *J. Chem. Phys.*, 122:206101.

Lin, M. S., Fawzi, N. L., and Head-Gordon, T. (2007). Hydrophobic potential of mean force as a solvation function for protein structure prediction. *Structure*, 15:727–740.

Oakley, M. T., Barthel, D., Bykov, Y., Garibaldi, J. M., Burke, E. K., Krasnogor, N., and Hirst, J. D. (2008). Search strategies in structural bioinformatics. *Current Protein and Peptide Science*, 9:260274.

Ramachandran, G. N. and Sasisekharan, V. (1968). Conformations of polypeptides and proteins. *Adv. Protein Chem.*, 23:283–437.

Sanvicente-Snchez, H. and Frausto-Sols, J. (2004). A method to establish the cooling scheme in simulated annealing like algorithms. *LNCS*, 3945:755–763.

Swendsen, R. H. and Wang, J.-S. (1986). Replica monte carlo simulation of spin-glasses. *Physical review letters*, 57:2607–2609.