ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Supervised learning to tune simulated annealing for *in silico* protein structure prediction

Alejandro Marcos Alvarez, Francis Maes and Louis Wehenkel *

Department of Electrical Engineering and Computer Science
University of Liège, Sart-Tilman B28, B-4000 Liège - Belgium

Abstract. Simulated annealing is a widely used stochastic optimization algorithm whose efficiency essentially depends on the proposal distribution used to generate the next search state at each step. We propose to adapt this distribution to a family of parametric optimization problems by using supervised machine learning on a sample of search states derived from a set of typical runs of the algorithm over this family. We apply this idea in the context of *in silico* protein structure prediction.

1 Introduction

Protein structure prediction is a topical and challenging open problem in bioinformatics. The significance of this problem is due to the importance of studying protein structures in biomedical research in order to improve our understanding of the human physiology and to accelerate drug design processes.

The most reliable way to determine protein structures is to use experimental methods such as X-ray crystallography or NMR spectroscopy. These methods are however expensive and time consuming, and hence the design of *in silico* protein structure prediction methods has become a very active research field. The computational problem may be formulated in the form of a parametric optimization problem whose goal is to find the global minimum of the energy function of the protein, which corresponds to the sought structure [1], based only on its amino-acid sequence. This modeling however generates, for usefully sized proteins, a very high dimensional[1] optimization problem in which the objective function has typically a huge number of local minima [1]. Therefore, meta-heuristic global optimization strategies (such as *Simulated Annealing* (SA) [2]) are often used to try to solve this problem [1, 3, 4].

Since the optimization problem has to be solved for many different proteins, and since the efficiency of the simulated annealing algorithm critically depends on the proposal distribution used to generate the next candidate structure at its intermediate steps, we propose to adapt this distribution to the protein structure optimization problem by using supervised learning on a sample of search states derived from a set of typical runs of the algorithm over this family.

The rest of this paper is organized as follows: Section 2 formulates abstractly the general problem we address and Section 3 sketches the solution

*This work was funded by the Biomagnet IUAP network of the Belgian Science Policy Office and the Pascal2 network of excellence of the EC. The scientific responsibility rests with the authors.

[1]The search space is $\mathbb{R}^{3n}$ where n is the number of atoms of the protein.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

Algorithm 1 Simulated annealing

Let B be a budget of iterations, $\mathcal{E}(\cdot)$ the oracle evaluating the energy and $T(i)$ a non increasing cooling schedule defined over $\{1, \ldots, B\}$.

Input: λ the problem instance, $\mathcal{S}_\lambda$ its solution space, $s^0 \in \mathcal{S}_\lambda$ the chosen initial state, and $p\left(o \mid s\right)$ a proposal distribution over $\mathcal{O}$ conditional on $s \in \mathcal{S}_\lambda$

$s = s^0$; $e = \mathcal{E}\left(s^0\right)$
for $i = 1 \ldots B$ **do**
  propose a state modification operator $o \in \mathcal{O}$ by drawing from $p\left(o \mid s\right)$
  $s' = o\left(s\right)$; $e' = \mathcal{E}\left(s'\right)$
  with probability $\min\left(1, \exp\left(\frac{e-e'}{kT(i)}\right)\right)$ (i.e. *Metropolis criterion*) **do**
    $s = s'$; $e = e'$
  end
end for
return s

framework that we propose. Section 4 explains how we applied this framework to protein structure prediction and outlines some first simulation results. Section 5 concludes and suggests future work directions.

2 General problem statement

We are interested in solving repeatedly a parametric optimization problem, defined by a parameter space Λ: for any $\lambda \in \Lambda$, we want to find the minimum over $\mathcal{S}_\lambda$ of an energy function $\mathcal{E}(\cdot)$ defined over the union of admissible state spaces $\mathcal{S} = \bigcup_{\lambda \in \Lambda} \mathcal{S}_\lambda$. We consider a sequential stochastic optimization procedure that receives the description of a problem instance (namely λ, and other relevant details), generates a sequence of length $B+1$ of states $\left(s_\lambda^0, \ldots, s_\lambda^B\right)$ and then returns the last state of this sequence s_λ^B. The procedure uses a proposal distribution $p\left(o \mid s\right)$, in order to generate this trajectory, as explained in Algorithm 1, which sketches the well-known simulated annealing procedure [2].

Given λ, the value returned by the algorithm, s_λ^B, is a random variable that results from the particular trajectory $\psi(\lambda, p, B) = \left(s_\lambda^0, \ldots, s_\lambda^B\right)$ the SA algorithm followed during its execution, which depends on the proposal distribution $p = p\left(o \mid s\right)$ and also on the budget B of iterations. We denote by $\mathrm{SA}_{p,B}(\lambda)$ the conditional distribution of s_λ^B given λ, B, and p. We assume that there is a set $\mathcal{P}$ of candidate proposal distributions over $\mathcal{S}$ and $\mathcal{O}$, and define our objective as the choice of $p \in \mathcal{P}$ in order to minimize the value $\mathcal{E}\left(s_\lambda^B\right)$ on average over all problems and over all induced trajectories of the SA algorithm. Denoting by $\mathcal{D}_\lambda$ the distribution of optimization problems, and fixing the optimization budget B, this corresponds to computing

$$p^* = \arg\min_{p \in \mathcal{P}} \left\{ \mathbb{E}_{\lambda \sim \mathcal{D}_\lambda, s_\lambda^B \sim \mathrm{SA}_{p,B}(\lambda)} \left[\mathcal{E}\left(s_\lambda^B\right) \right] \right\}. \tag{2.1}$$

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

3 Proposed supervised learning based framework

Rather than directly searching for a solution of problem (2.1), which is intractable in general, we propose an indirect approach to improve a given proposal distribution. First, we parameterize the space of candidate distributions $\mathcal{P}$, so that finding an optimal $p^* \in \mathcal{P}$ amounts to solving

$$\theta^* = \arg\min_{\theta \in \mathbb{R}^d} \left\{ \mathbb{E}_{\lambda \sim \mathcal{D}_\lambda, s_\lambda^B \sim \mathrm{SA}_{p_\theta, B}(\lambda)} \left[\mathcal{E}\left(s_\lambda^B\right) \right] \right\}. \tag{3.1}$$

where the vector $\theta \in \mathbb{R}^d$ represents our degrees of freedom of choice of p_θ. We also represent the states s_λ by a vector of features $\phi(s_\lambda) \in \mathbb{R}^p$ and formulate $p_\theta\left(o \mid s\right)$ as a function of θ and ϕ (see Section 4, for a precise example).

From the initial proposal distribution p we would then like to find a value $\widetilde{\theta}$ defining a better proposal distribution. To do this, we proceed as follows:

- First we choose a sample of problems $\Pi = (\lambda_i)_{i=1}^m$ according to $\mathcal{D}_\lambda$ and we apply to each λ_i Algorithm 1 with p as proposal distribution.
- We then exploit the resulting set of trajectories $S = \left\{ s_{(\lambda_j)}^i \right\}_{(i,j)=(0,1)}^{(B,m)}$:
  - For each $s \in S$ (or for a subset of S), we apply a search algorithm to determine $\tilde{o}(s) \in \mathcal{O}$ leading to a good decrease in energy.
  - We build a learning sample composed of the pairs $(\phi(s), \tilde{o}(s))$ and use a supervised learning algorithm to derive a model $p_\theta\left(o \mid s\right)$ maximizing the (conditional) likelihood of $(\phi(s), o(s))$.

This strategy leads to the creation of a $p_\theta\left(o \mid s\right)$ that will, at each iteration of SA, select an operator that well decreases (on average) the energy of the system. The result will be a quite greedy operator selection policy that will, at first glance, speed up the convergence but might lead the system to a local minimum. This is however counterbalanced by SA that already has a countermeasure to avoid this trap. The main rationale behind this approach is that average efficiency of the SA algorithm, in terms of the average optimality of its final states, is well reflected by the average quality of its proposal distribution in terms of its average local improvements of the objective function. Of course, the whole procedure may be bootstrapped, leading then to a sequence of 'approximate policy gradient' iterations over the space of proposal distributions.

4 Application to *in silico* protein structure prediction

A protein is entirely described by its amino acid sequence, we thus used this sequence as the parameters λ of a problem instance. The set $\mathcal{S}_\lambda$ is the set of tridimensional positions of the atoms of protein λ and $\mathcal{E}$ is an energy reflecting the chemical stability of the protein.

We use the Rosetta software [4] to handle the proteins (energy computation and implementation of structure modification operators). The operators

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

$o(m, \gamma)$ used by SA are defined by the type $m \in \mathcal{M}$ of modification they produce on the structure and by a vector of parameters $\gamma_m = (\gamma_{m,1}, \ldots, \gamma_{m,L_m})$ in which each component is either continuous (magnitude of the modification e.g.) or discrete (choice of the amino acid on which the operator is applied). We use three different operators that produce, for two of them, rotations of the backbone of the protein and, for the last one, respective translation and/or rotation of two rigid sets of amino acids (see [5], for further details of our work).

4.1 Features describing a protein structure

The 23 first features encode the amino acids histogram of the protein. The other features are obtained by discretizing the following three quantities: the length n of the protein, its current energy and a compactness measure $f_c(s_\lambda) = \frac{x}{n}$ where x is the average distance between amino acids in s_λ. These quantities are discretized into, respectively, 12, 10 and 20 bins before being concatenated to form the feature vector $\phi(s_\lambda)$ of dimension $p = 23 + 12 + 10 + 20 = 65$.

4.2 Assumptions about $\mathcal{P}$

We make the assumption that operators can be drawn from p_θ in two steps, by first drawing the type m of the operator and then accordingly the parameters γ_m of the drawn operator. We also make the assumption that the parameters of each operator are drawn independently of each other, therefore allowing to learn their marginal distributions independently. The probability distribution p_θ, with $\theta = \left(\theta^t, \theta^p_{1,1}, \ldots, \theta^p_{|\mathcal{M}|,L_{|\mathcal{M}|}}\right) \in \mathbb{R}^d$, can thus be formulated as follows[2]

$$p_\theta\left(o = (m, \gamma_m) \mid \phi(s_\lambda)\right) = p^t_{\theta^t}\left(m \mid \phi(s_\lambda)\right) \prod_{i=1}^{L_m} p^p_{\theta^p_{m,i}}\left(\gamma_{m,i} \mid \phi(s_\lambda)\right). \tag{4.1}$$

Operator-type proposal distribution. We model the choice of the operator type $m \in \mathcal{M}$ with a log-linear conditional probability distribution, parameterized by $\theta^t = \left(\theta^t_1, \ldots, \theta^t_{|\mathcal{M}|}\right)$, whose formulation is

$$p^t_{\theta^t}(m \mid \phi(s_\lambda)) = \frac{1}{Z_{\theta^t}(\phi(s_\lambda))} \exp\langle \theta^t_m, \phi(s_\lambda)\rangle \tag{4.2}$$

where $Z_{\theta^t}(\phi(s_\lambda))$ is a normalization factor. This distribution implements the well-known maximum entropy classifier and is learned using the corresponding algorithm that maximizes the log-likelihood of the data [6].

Operator-parameter proposal distribution. We impose that the discrete parameters (amino-acid choices) are drawn uniformly, while the continuous parameters are drawn from a Gaussian distribution which mean and standard deviation are learned functions of the protein features. One distribution is

[2]The total number d of parameters is 975.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

used for each continuous parameter of each operator type. The distributions, parameterized by $\theta^p = (\theta^p_\mu, \theta^p_\sigma) \in \mathbb{R}^2$, are of the form

$$p^p_{\theta^p}\left(\gamma \mid \phi(s_\lambda)\right) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{-0.5\left(\frac{\gamma-\mu}{\sigma}\right)^2\right\}; \quad (4.3)$$

$$\text{with } \mu = \langle \theta^p_\mu; \phi(s_\lambda)\rangle \quad \text{and} \quad \sigma = \log\left\{1 + \exp\left(-\langle \theta^p_\sigma; \phi(s_\lambda)\rangle\right)\right\}. \quad (4.4)$$

These distributions are trained using a stochastic gradient descent algorithm that maximizes the log-likelihood of the data at hand. The particular expression of σ has been chosen to improve numerical stability.

4.3 Creation of the learning database

We use the framework detailed in Section 3 in order to save a certain number of protein structures in the database D necessary to learn the probability density function (4.1). We then apply an estimation of distribution algorithm (EDA, [7]) to each structure in D to discover good operators for these structures. The EDA works iteratively and, at each iteration, randomly samples K operators from a distribution[3] and then evaluates their quality with the energy function of the protein. The $k < K$ best operators are then chosen to compute a new distribution making good operators more probable. This process is repeated to generate progressively better operators, until convergence.

4.4 Simulation results

The training set used to learn the distribution p_θ and the test set used to assess the performances are composed, respectively, of 100 and 10 proteins of less than 100 amino acids randomly selected from the database PSIPRED [8]. The Metropolis criterion used in SA has been determined by a rule of thumb based on what can be found in official Rosetta tutorials (see [5] for more details).

Figure 1 illustrates the results of one run of the optimization procedure. The curves represent the evolution of the average energy of the proteins of the test set. The blue curve corresponds to the optimization by SA with the original proposal distribution while the red curve corresponds to the use of the learned one. This latter outperforms the first one in terms of convergence speed and of final result. More precisely, it yields, after around 2×10^4 iterations, the same results than the first method after 2.5×10^5 iterations. While these results are encouraging, the structures predicted after one such learning iteration are still very different from the real structures.

5 Conclusions and future work

The idea developed in this article shows that we can get improvement in performance by learning the proposal distribution used by SA from automatic experiments with this algorithm. When applied to the *in silico* protein structure prediction problem, the procedure already shows promising results that could lead in the end to important breakthroughs in this field.

[3]This distribution is the same as the one described in Section 4.2 but, in this case, $\phi(s_\lambda) = \{1\}$ because we want to learn $p(o)$ while, in Section 4.2, we were interested in $p(o \mid \phi(s_\lambda))$.

ESANN 2012 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2012, i6doc.com publ., ISBN 978-2-87419-049-0.
Available from http://www.i6doc.com/en/livre/?GCOI=28001100967420.

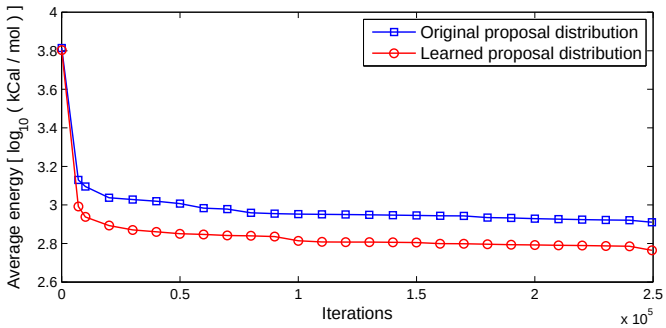

Fig. 1: Evolution of average energy of the test set proteins during optimization.

This approach could be called *learning for search* as it consists in learning a good way to search through the state space of a problem, and can of course be applied to other complex optimization problems and other heuristic search methods. In the present work, the improvement was obtained through the learning of a *locally* good proposal distribution. In general, better efficiency may be expected if learning could also take into account more *global* information about the optimization process. In the context of protein structure prediction, such information can be provided by supplying the learning algorithm with the description of the true structure, e.g. obtained from wet-lab experiments.

Further work includes the improvement of the optimization algorithm (e.g. fine tuning of its other parameters and testing of other algorithms) and improvement of the learning procedure itself (e.g. local- vs global considerations and improvement of feature and model selection techniques).

References

[1] P. E. Bourne and H. Weissig. *Structural bioinformatics*. Wiley, first edition, 2003.

[2] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, New Series*, 220(4598):671–680, 1983.

[3] P. Koehl. Protein structure prediction. In *Biomedical Applications of Biophysics*, volume 3 of *Handbook of Modern Biophysics*, pages 1 – 34. Humana Press, 2010.

[4] A. Leaver-Fay *et al.* Rosetta3: An object-oriented software suite for the simulation and design of macromolecules. In Michael L. Johnson and Ludwig Brand, editors, *Computer Methods, Part C*, volume 487 of *Methods in Enzymology*, pages 545 – 574. Academic Press, 2011.

[5] A. Marcos Alvarez. Prédiction de structures de macromolécules par apprentissage automatique. Master's thesis, University of Liège, Faculty of Engineering, 2011.

[6] A. L. Berger, V. J. D. Pietra, and S. A. D. Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[7] P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Springer, October 2002.

[8] D. T. Jones. Protein secondary structure prediction based on position-specific scoring matrices. *Journal of Molecular Biology*, 292(2):195 – 202, 1999.