Protein Folding in 2-Dimensional Lattices with Estimation of Distribution Algorithms

Roberto Santana, Pedro Larrañaga, and José A. Lozano

Intelligent System Group
Department of Computer Science and Artificial Intelligence
University of the Basque Country
P.O. Box 649, 20080 San Sebastián - Donostia, Spain
{rsantana,ccplamup,lozano}@si.ehu.es

Abstract. This paper introduces a new type of evolutionary computation algorithm based on probability distributions for the solution of two simplified protein folding models. The relationship of the introduced algorithm with previous evolutionary methods used for protein folding is discussed. A number of experiments for difficult instances of the models under analysis is presented. For the instances considered, the algorithm is shown to outperform previous evolutionary optimization methods.

Keywords: Estimation of Distribution Algorithms, protein folding, HP model.

1 Introduction

Searching for the minimum conformation structure of a protein given its sequence is a difficult problem in computational biology. Even for a small number of amino acids, the conformational space of proteins is huge. This fact has led to the need of using simplified models that can help to find approximate structures of proteins. Lattice models are an example, where each amino acid of a protein can be represented as a bead, and connecting bonds are represented by lines, which follow the geometry of the chosen background lattice [10]. The problem consists in finding the structure in the lattice that minimizes a predefined fitness function associated with protein structure stability. These problems can be dealt with using search algorithms that try to optimize the fitness function. In these simplified models, the dimension of the search space remains huge. Therefore, the efficiency of the optimization algorithm is critical for the success of the search.

Several different heuristics [1, 3, 8, 11, 14, 19] have been applied to a simplified version of the protein folding problem called the Hydrophobic-Polar (HP) model [4]. The HP model is based on the fact that hydrophobic interactions are a dominant force in protein folding. Although simple, this model has proven to be useful as a test bed for folding algorithms.

In this paper, to solve the HP model we propose a new type of evolutionary computation algorithm that belongs to the class of Estimation of Distribution Algorithms (EDAs) [13, 17]. The EDA is also applied to a version of the HP

J.M. Barreiro et al. (Eds.): ISBMDA 2004, LNCS 3337, pp. 388–398, 2004.

model called the functional model protein [6], which has a unique native state (i.e. a unique global optimum), and thus difficult to solve with optimization algorithms. We show that the results achieved with the EDA are better than those obtained with other evolutionary optimization techniques [9, 12, 19], and are also competitive with other approaches.

The paper is organized as follows. In the next section, we introduce the HP model and present the problem representation. Section 3 briefly reviews a number of previous approaches to the solution of simplified models. Section 4 presents the class of EDAs, and introduces the EDA used for protein folding. Section 5 presents the experimental benchmark and the numerical results of our experiments. In section 6, the conclusions of our work are given, and possible extensions of EDAs for dealing with the protein folding problem are discussed.

2 The Hydrophobic-Polar Model

In the HP model a sequence comprises residues of only two types: hydrophobic (H) and hydrophilic or polar (P). Residues are located in regular lattice models forming self-avoided paths. There is a zero contact energy between P-P and H-P pairs. Different values can be taken to measure the interaction between hydrophobic non-consecutive residues; a common choice that we use in this paper is -1. The energy interactions can be represented by the following matrix:

$$E = \begin{pmatrix} -1 & 0 \\ 0 & 0 \end{pmatrix}$$

The function evaluation of a given configuration or protein conformation is simplified to the sum of every two hydrophobic residues that are non-consecutive nearest neighbors on the lattice.

In this paper, we consider the 2-dimensional regular lattice. In the linear representation of the sequence, hydrophobic residues are represented with the letter H and polar ones with P. In the graphical representation, hydrophobic proteins are represented by black beads and polar proteins by white beads. Figure 1 shows an optimal folding for the sequence $S1 = HPHPPHHPHPPHPHHPPHPH$. The optimal energy corresponding to this sequence is -9.

2.1 Functional Model Protein

The functional model protein is a 'shifted' HP model. The name comes from the fact that the model supports a significant number of proteins that can be characterized as functional. This model has native states, some of which are not maximally compact. Thus, in some cases, they have cavities or potential binding sites, a key property that is required in order to investigate ligand binding using these models [6]. The energy matrix associated with the model contains both attractive and repulsive interactions. Its representation is as follows:

$$E = \begin{pmatrix} -2 & 1 \\ 1 & 1 \end{pmatrix}$$

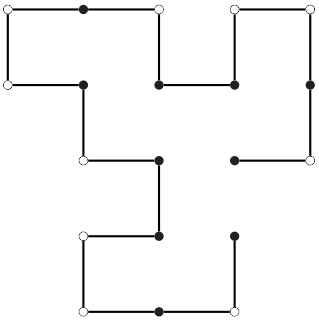

Fig. 1. Best solution found for the sequence $S1$.

2.2 Problem Representation

We use X_i to represent a discrete random variable. A possible instance of X_i is denoted x_i. Similarly, we use $\mathbf{X} = (X_1, \ldots, X_n)$ to represent an n-dimensional random variable and $\mathbf{x} = (x_1, \ldots, x_n)$ to represent one of its possible instances. For a given sequence on the lattice, X_i will represent the relative move of residue i in relation to the previous two residues. Taking as a reference the location of the previous two residues in the lattice, X_i takes values in $\{0, 1, 2\}$. These values respectively mean that the new residue will be located on the left, forward or on the right with respect to the previous locations. Therefore, values for X_1 and X_2 are meaningless. The locations of these two residues are fixed as positions $(0, 0)$ and $(1, 0)$. The representation corresponding to the sequence shown in figure 1 is $\mathbf{x} = (0, 0, 0, 1, 0, 0, 2, 2, 0, 1, 0, 0, 2, 0, 2, 2, 0, 0, 1, 0)$.

3 Previous Approaches to the Solution of Simplified Protein Models

We focus on approaches to simplified protein folding based on Genetic Algorithms (GAs) [7] and Monte Carlo (MC) methods [16]. Some versions of these methods are compared with EDAs in the experiment section.

A pioneer work in the use of population based optimization algorithms for the simplified protein folding problem is [19]. In this paper, the authors proposed a GA that used heuristic based crossover and mutation operators. The GA outperformed a number of variants of MC methods at different sequences. In [11], a search strategy called pioneer search was used together with a simple GA. Although the algorithm improved some of the results achieved in [19], it was unable to find the optimal solutions for the longest instances considered.

In [3] and [9] evolutionary algorithms for the 3-D HP problem are proposed. While in [9] a simple GA shows no better results than those achieved in [19], a more sophisticated approach is presented in [3]. By using a backtracking based repairing procedure, the latter algorithm guarantees that the search is constrained to the space of legal solutions. Since the number of self-avoided paths on square

lattices is exponential in the legth of the sequence [19], generating legal solutions with a backtracking algorithm is a feasible alternative.

The MultiMeme Algorithm (MMA) for protein structure prediction [12] is a GA combined with a set of local searches. From this set, the algorithm self-adaptively selects which local search algorithm to use for different instances, states of the search, or individuals in the population. The results achieved for a number of instances were better than those reported in [19], but in some of the most difficult instances the algorithm failed to reach the optimal solution.

Traditional MC methods sample from the protein folding space one point at the time. However, and due to the rugged landscape, traditional MC methods tend to get trapped in local minima. Two alternatives to avoid this problem are either to use chain growth algorithms [1], or to sample the space with a population of Markov chains in which a different temperature is attached to each chain [15]. A common and remarkable characteristic of these methods is that they employ problem information to improve the results of the optimization algorithms.

4 Estimation of Distribution Algorithms

We study the suitability of EDAs as a non-deterministic search procedure for the HP model. A main difference between EDAs and GAs is that the former constructs an explicit probability model of the solutions selected. This model can capture, by means of probabilistic dependencies, relevant interactions among the variables of the problem. The model can be conveniently used to generate new promising solutions. The main scheme of the EDA approach is shown in algorithm 1. Although the introduction of EDAs is relatively new, there already exists a number of succesful applications of EDAs in computational biology [2, 18].

EDAs differ in the type of models that they use and the corresponding factorizations of the probability that these models determine. For the protein folding problem, we define a probability model that assumes that proteins adjacent in the sequence are related in their lattice positions. The probability model then encodes the dependencies between the move of a residue and the moves of the previous residues in the sequence. This information is used in the generation of solutions.

Let $p(\mathbf{x})$ be the probability distribution of random variable $\mathbf{X}$. Our probability model considers that the configuration of variable X_i depends on the configuration of the previous k variables, where $k \geq 0$ is a parameter of the model. $p(\mathbf{x})$ can be factorized as follows:

$$p(\mathbf{x}) = p(x_1, \ldots, x_{k+1}) \prod_{i=k+2}^{n} p(x_i \mid x_{i-1}, x_{i-2}, \ldots, x_{i-k}) \tag{1}$$

The learning phase of our EDA will only comprise a parametric learning of the parameters in contrast to some state-of-the-art EDAs that make structural

Algorithm 1: Main scheme of the EDA approach

```
1  D_0 ← Generate M individuals (the initial population) randomly
2  l = 1
3  do {
4     D^s_{l-1} ← Select N ≤ M individuals from D_{l-1} according to a selection method
5     p_l(x) = p(x|D^s_{l-1}) ← Estimate the joint probability of selected individuals
6     D_l ← Sample M individuals (the new population) from p_l(x)
7  } until A stop criterion is met
```

and parametric learning of the model. Therefore, the computational complexity of the algorithm is reduced, and it can be faster than sophisticated GAs that incorporate complex local search procedures. In addition to the probabilistic model, there are two particular features that characterize our EDA approach to the protein folding problem. The first one is the inclusion of a restart step in the EDA. The restart step tries to avoid early convergence of the population. The other feature added to our EDA is a method to ensure that all the vectors evaluated are valid (i.e. self-avoided) paths. We describe these two additions to the EDA scheme shown above in detail. It should also be noted that none of these changes use knowledge about the problem.

Every time that the diversity of solutions in the population goes under a predefined threshold, all solutions except the best are randomly modified with a given probability value in the same way mutation operators are applied in GAs, but with a higher mutation probability. Diversity is measured by calculating the number of different vectors in the selected population divided by N. Restart tries to avoid the early convergence of the population.

In the representation that we used, not all vectors correspond to self-avoiding sequences. Our search procedure organizes the search within the space of valid solutions. To enforce the validity of the solutions, we employ the backtracking method proposed in [3]. This method can be used in two different ways: as a generator procedure or as a repairing algorithm. In the first case, a solution is incrementally constructed in such a way that the self-avoidance constraint is fulfilled. At position i, the backtracking call is invoked only if self-avoidance cannot be fulfilled with any of the three possible assignments to X_i.

Used as a repairing method, the algorithm inspects every sampled solution. It checks whether the current vector position assignments violates the self-avoidance constraint. If such is the case, another value is assigned to the position and tested. The order of the assignment of variables is random. If all the three possible values have been checked, and self-avoidance is not fulfilled yet, backtracking is invoked. Further details about the backtracking algorithm, originally proposed for the 3-D HP model, can be found in [3].

The repairing procedure destroys some of the statistical dependencies generated from the model. However, the effect of this step is beneficial because solutions will be altered only if their current assignment violates the constraint.

There exist EDAs that are able to generate solutions that consider the fulfillment of constraints at the generation step [13]. A similar approach could be applied to the protein folding problem. Nevertheless, this procedure has an additional computational cost because the fulfillment of the constraints must be checked at each step of the solution generation.

5 Experiments

We compare the results achieved by the EDA approach to results obtained with previous GAs and other MC heuristics. First, we present the set of instances used for the experiments. The algorithms compared are later presented. Finally, the results of the experiments are shown and discussed.

5.1 Function Benchmark

Two different sets of instances are used in our experiments. Table 1 shows HP sequences $S1$-$S7$, which were originally proposed in [19]. The optima corresponding to some of these sequences were incorrectly determined in [19]. Optimal values shown in table 1 have been taken from [15]. These sequences have been used as a benchmark for different algorithms [1, 11, 14, 15, 19].

Table 1. HP instances used in the experiments.

name	size	opt.	sequence
$S1$	20	-9	$HPHPPHHPHHPHPHHPPHPH$
$S2$	25	-8	$PPHPPHHP^4HHP^4HHP^4HH$
$S3$	36	-14	$P^3HHPPHHP^5H^7PPHHP^4HHPPHPP$
$S4$	48	-23	$PPHPPHHPPHHP^5H^{10}P^6HHPPHHPPHPPH^5$
$S5$	50	-21	$HHPHPHPHPH^4PHP^3HP^3HP^4HP^3HP^3HPH^4\{PH\}^4H$
$S6$	60	-36	$PPH^3PH^8P^3H^{10}PHP^3H^{12}P^4H^6PHHPHP$
$S7$	64	-42	$H^{12}PHPH\{PPHH\}^2PPH\{PPHH\}^2PPH\{PPHH\}^2PPHPHPH^{12}$

Table 2 shows sequences $S8$-$S18$ that belong to the functional model protein and they were previously used as a benchmark in [12]. All these intances have size 23. They are an example of a challenging set of problems with only one solution.

5.2 Design of the Experiments

To evaluate the behavior of the EDA introduced in this paper, we compare its results with the results achieved by the GA and MC algorithms presented in [19] and MMA [12]. In the case of the HP model, results of the MMA are available only for some of the sequences shown in table 1. On the other hand, experiments

Table 2. Two dimensional functional model protein instances.

name	opt.	sequence
$S8$	-20	$PHPPHPPHHHHPPHPPHPHPPHH$
$S9$	-17	$PHPPHPPHHHHHPPPPHPPHPPH$
$S10$	-16	$HPHPHPHHHPPHPPPHPHHPPHH$
$S11$	-20	$HHHPHHHPPHHPPPHPHPPHHHH$
$S12$	-17	$PHPPPPPPHPHHPHPHHHHPHPH$
$S13$	-13	$HHPHPPHPPPPHPPPPHPPPHHH$
$S14$	-26	$PHPHHPHHHHHHPPHHHPHHHHH$
$S15$	-16	$HPHPPPHHHHPHPPPPHPHPHHH$
$S16$	-15	$PHPHHPHHPHHPHPHPHPPPPPH$
$S17$	-14	$HPHPHPPPPPHHPPPHPHPHPHH$
$S18$	-15	$PHPPHHHPHPPHPHHPHPPPPPH$

for the instances of the functional model shown in table 2 were not provided in [19], and results are only available for MMA.

The GA population size used in [19] was $M = 200$ and the maximal number of generations was $g = 300$. The MC algorithms performed 50000000 steps. MMA uses tournament selection, crossover probability 0.8, mutation probability 0.3 and replacement strategies with different population sizes. The main criteria that we used to compare the algorithms were their effectiveness to find the optimum, and the number of evaluations needed to reach the optimum.

In all of the experiments done in this paper, the EDA uses truncation selection of parameter $T = 0.1$. We use best elitism, a replacement strategy where the population selected at generation t is incorporated into the population of generation $t + 1$. Let M be the population size, only $M - T * M$ individuals are generated at each generation except the first one. The threshold used for the restart process was 0.5. The probability of modifying the value at every position is 0.9. Although better results can be achieved by tuning the EDA parameters for each sequence, we kept all the parameters except population size fixed.

5.3 Results of the Experiments

Table 3 presents the results achieved by the different algorithms for sequences $S1$-$S7$. The results of EDAs are the average of ten experiments for the given parameters. The results of GA, MC and MMA are the best taken as the most efficient run in five, so the comparison gives only an idea of the relative performance of the algorithms. In the table, B is the energy of the best solution found when it was not the optimal one. Optimal values of the energies for the sequences are shown in table 1.

We calculate the number of times that the best solution was found (S) and the average number of function evaluations ($\bar{e}$) needed by the EDA to find the this solution. $k = 2$ was the choice for parameter k. We use parameter $k = 3$ only when the optimum was not found with $k = 2$. When the results are improved

Table 3. Results achieved by the different algorithms for the HP instances.

		EDA				*GA*		*MC*		*MMA*	
inst.	k	B	M	S	$\bar{e}$	B	$\bar{e}$	B	$\bar{e}$	B	$\bar{e}$
$S1$	2		1000	10	4510		30492		292443		14621
$S2$	2		4000	10	13880		20400		2694572		18736
$S3$	2		5000	1	113000		301339	−13	6557189		208233
$S4$	2		5000	2	53995	−22	126547	−20	9201755	−22	1155656
$S5$	2		10000	10	118000		592887		15151203		
$S6$	2	−35	10000	2	473500	−34	208781	−33	8262338		
$S7$	2	−41	10000	7	595900	−37	187393	−35	7848952		
$S7$	3		10000	1	154000						

using this value, they are shown. In the table, we include the best results achieved by the GA and MC algorithms presented in [19] and MMA [12].

The first conclusion of our experiments is that the EDA is able to find the best known optimum for all instances except instance $S6$, where nevertheless it achieves better results than the other algorithms. For the instances where all the algorithms find the optima, the average number of evaluations needed by the EDA is smaller than the best result achieved by the experiments conducted with the three other algorithms. $S6$ and $S7$ are the most difficult instances among those shown in table 1. For $S7$, the best result achieved with one of the best MC algorithms [1] that uses information about the problem structure was -40. The EDA has been able to find more than one optimal solution for this instance. In figure 2, the configurations corresponding to two optimal solutions found by the EDA are shown. As far as we know, only two methods have been able to deal with this instance successfully [14, 15].

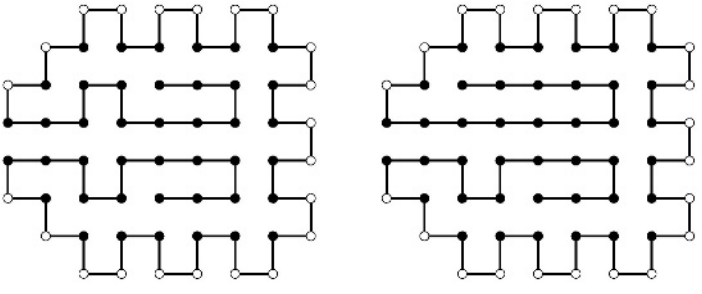

Fig. 2. Two optimal solutions found for the $S7$ sequence.

Now we evaluate the EDA for the functional model protein instances. The number of evaluations needed by MMA to optimize the functional model protein instances are shown in table 4. The results correspond to the best out of five experiments where the optimum has been reached at least once. To make a fair

comparison, we run our algorithm in similar conditions. We find the population size (M) for which the EDA finds the optimum in at least one of the five experiments. Additionally, we present the number of times (S) the optimum was found with the given population size and the number of evaluations (e) for the best run where it was found.

For the functional model protein instances, we found that the simple EDA with $k = 1$ was able to improve the results achieved by MMA. In table 4, it can be appreciated that the EDA is able to find the optimum for all the instances with a number of evaluations that is, in all cases, lower than the number needed by MMA. Another observation is that, for eight of the eleven instances treated, the optimum could be found with the minimum population size tested, i.e. $M = 500$.

Table 4. Results for the two dimensional functional model protein instances.

	EDA				MMA
inst.	k	S	M	e	e
$S8$	1	4	500	12650	15170
$S9$	1	5	500	2750	61940
$S10$	1	1	1000	35900	132898
$S11$	1	3	1000	15900	66774
$S12$	1	4	500	20950	53600
$S13$	1	5	500	5420	32619
$S14$	1	1	500	19450	114930
$S15$	1	1	1500	10350	28425
$S16$	1	1	500	4950	25545
$S17$	1	2	500	8950	111046
$S18$	1	5	500	2950	52005

6 Conclusions

The probability models used by the EDA introduced in this paper can be regarded as Markovian models where the parameter k determines the extent of the dependency on previous variables. Markovian models have been used in computational biology to identify coding regions in genes and in sequence alignment. However, the authors are not acquainted with any previous application of these models in the context of population based search algorithms applied to computational biology problems.

On the other hand, while most of current EDA applications consider structural learning algorithms, our proposal emphasizes the convenience of using the dependencies determined by the sequence ordering to construct the model structure. This strategy reduces the computational cost of the EDA learning phase. Even so, the use of the parameter k gives some flexibility to the model learning step without the need of the more costly structural learning.

The experimental results have proven the effectiveness of the EDA approach and the fact that it is a usable search algorithm. The EDA approach can be extended in many ways. We enumerate some of these possible developments:

1. The structure of the probability model does not have to be fixed and can be learned from the data.
2. Off-lattice HP problems [8], where continuous variables represent the angles between contiguous residues, can be dealt with using EDAs that store probability models for continuous variables [13].
3. Problem information can be added by incorporating structural and parametric priors in the probabilistic models.
4. The algorithm can be combined with local optimizers in different ways.
5. EDAs could be applied to less simplified versions of the protein folding problem. In this area, a number of GA applications have been proposed [5].

Acknowledgments

This work was supported in part by the Spanish Ministerio de Ciencia y Tecnología under grant TIC2001-2973-C05-03, by Etortek-Genmodis and Etortek-Biolan projects from the Basque Government, and by the University of the Basque Country under grant 9/UPV 00140.226-15334/2003.

References

1. U. Bastolla, H. Frauenkron, E. Gerstner, P. Grassberger, and W. Nadler. Testing a new Monte Carlo algorithm for protein folding. *Proteins: Structure, Function, and Genetics*, 32:52–66, 1998.
2. R. Blanco, P. Larrañaga, I. Inza, and B. Sierra. Selection of highly accurate genes for cancer classification by Estimation of Distribution Algorithms. In *Proceedings of the Workshop 'Bayesian Models in Medicine' held within AIME 2001*, pages 29–34, 2001.
3. C. Cotta. Protein structure prediction using evolutionary algorithms hybridized with backtracking. In J. Mira and J. R. Alvarez, editors, *Artificial Neural Nets Problem Solving Methods*, volume 2687 of *Lecture Notes in Computer Science*, pages 321–328. Springer Verlag, 2003.
4. K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.
5. W. Garrison, W. Greenwood, and J.-M. Shin. *Evolutionary Computation in Bioinformatics*, chapter On the Evolutionary Search for Solutions to the Protein Folding Problem, pages 115–136. Morgan Kaufmann, 2002.
6. J. D. Hirst. The evolutionary landscape of functional model proteins. *Protein Engineering*, 12:721–726, 1999.
7. J. H. Holland. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, Ann Arbor, MI, 1975.
8. H.-P. Hsu, V. Mehra, and P. Grassberger. Structure optimization in an off-lattice protein model. *Physical Review E*, 68(2):4 pages, 2003.

9. M. Khimasia and P. Coveney. Protein structure prediction as a hard optimization problem: The genetic algorithm approach. *Molecular Simulation*, 19:205–226, 1997.
10. P. Koehl and M. Delarue. Building protein lattice models using self consistent mean field theory. *Journal of Chemical Physics*, 108:9540–9549, 1998.
11. R. König and T. Dandekar. Improving genetic algorithms for protein folding simulations by systematic crossover. *Biosystems*, 50:17–25, 1999.
12. N. Krasnogor, B. Blackburne, E. K. Burke, and J. D. Hirst. Algorithms for protein structure prediction. In J. M. Guervos, P. Adamidis, H.-G. Beyer, J.-L. Fernandez-Villacañas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VII*, pages 769–778, Paris, France, 2002. Springer Verlag. LNCS 2439.
13. P. Larrañaga and J. A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Optimization.* Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
14. N. Lesh, M. Mitzenmacher, and S. Whitesides. A complete and effective move set for simplified protein folding. Technical Report TR-2003-03, Mitsubishi Electric Research Laboratories, February 2003.
15. S. Liang and W. H. Wong. Evolutionary Monte Carlo for protein folding simulation. *Journal of Chemical Physics*, 115:3374–3380, 2001.
16. N. Metropolis, A. Rosenbluth, A. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1091, 1953.
17. H. Mühlenbein and G. Paaß. From recombination of genes to the estimation of distributions I. Binary parameters. In A. Eiben, T. Bäck, M. Schoenauer, and H. Schwefel, editors, *Parallel Problem Solving from Nature - PPSN IV*, pages 178–187, Berlin, 1996. Springer Verlag.
18. Y. Saeys, S. Degroeve, D. Aeyels, P. Rouzé, and Y. VandePeer. Feature selection for splice site prediction: A new method using EDA-based feature ranking. *BMC Bioinformatics*, 4:5–64, 2004.
19. R. Unger and J. Moult. Genetic algorithms for protein folding simulations. *Journal of Molecular Biology*, (231):75–81, 1993.