Optimization in biological models that use recurrent neural nets

George Marnellos and Eric Mjolsness

Technical Report Number CS97-524

January 1997

Computer Science and Engineering
University of California, San Diego

Optimization in biological models that use recurrent neural nets

George Marnellos and Eric Mjolsness

Technical Report Number CS97-524

January 1997

Optimization in Biological Models that Use Recurrent Neural Nets

George Marnellos[1] and Eric Mjolsness[2]

[1] Section of Neurobiology, Yale University, New Haven CT 06510, USA, and
Sloan Center for Theoretical Neurobiology, The Salk Institute,
La Jolla CA 92037, USA
[2] Department of Computer Science and Engineering and
Institute for Neural Computation, UCSD,
La Jolla CA 92093, USA

Abstract. We have applied genetic algorithms and simulated annealing to optimize energy functions in three biological models. These models use recurrent neural nets to produce desired dynamical patterns; their cost functions are functions of the neural net activation patterns over time. We have compared the performance of the two optimization methods on these problems and and found that, despite the apparent problem similarity, the method that performed better varied from problem to problem. We consider possible explanations for this result.

1 Introduction

In recent decades there has been intense study at the molecular level of fundamental biological processes in development, physiology and evolution. In order to integrate the experimental data that have been gathered and understand the structure and dynamics of complex biological systems and their evolution, computational models and simulations are needed.

We have used recurrent neural nets to simulate how genes and their interactions in cells determine the phenotypes of animal organs or simple organisms and their development, as well as how such gene interactions evolve under particular (simulated) environmental conditions or constraints: nodes in these neural nets correspond to genes and node activation levels to gene expression levels (see [1] and [2]). We have optimized the parameters in these models (node interaction strengths, activation and decay rates, thresholds and so on) in order to either fit experimental data (gene expression patterns) or to impart desired features to the simulated system and make it conform to constraints.

We have examined stochastic techniques to optimize the analytically intractable energy (or scoring) functions of our models; specifically, we have used the techniques of simulated annealing (SA) and genetic algorithms (GA) on the following problems: (a) simulating the development of a simple multicellular organism capable of reproducing and its evolution under various selective

environments, (b) fitting gene-expression patterns observed during early neurogenesis in *Drosophila*, and (c) fitting multi-dimensional Lissajous curves with the use of a recurrent neural net.

In this paper we will be describing the SA and GA algorithms we have used, as well as the problems we have applied them on, and will be comparing their performance. Our results indicate that the performance of both algorithms varies across problems.

2 Optimization Algorithms

The energy functions in our models have a large number of variables and are highly nonlinear and cannot be solved analytically or readily optimized with deterministic methods. We have therefore used numerical, stochastic techniques to optimize them. Both optimization methods we have employed have a number of parameters that can greatly affect their performance and so need to be tuned for each individual problem.

2.1 Simulated Annealing

Our simulated annealing algorithm incorporates standard annealing features (see [3]) and uses Lam's temperature decreasing schedule and move generation strategy ([4], [5]): a move thus consists of increasing or decreasing the value of one state variable x_i at a time by an amount selected randomly from an exponential distribution with mean θ_i

$$x_i^{new} = x_i \pm \theta_i \ln \xi, \tag{1}$$

where the sign is chosen randomly and ξ is a random number uniformly distributed in $[0, 1)$. Moves are generated for each state variable in turn and are either accepted or rejected. The temperature T is lowered after every move by

$$T_{n+1} = \frac{T_n}{1 + k(T_n)T_n}, \tag{2}$$

where T_n is the temperature at move n and $k(T_n)$ is a function of the estimated variance in the energy at that temperature and of the ratio of accepted to proposed moves.

Lam has shown that the state space is sampled most effectively when the acceptance ratios for moves in each of the variables are maintained close to 0.44. This is done by updating the θ's at regular intervals, using

$$\theta_i^{new} = \theta_i \exp(\rho_i - 0.44), \tag{3}$$

where ρ_i is the acceptance ratio for variable x_i. The mean move size is thus individually controlled for each variable: when ρ_i is smaller than 0.44, mean move size θ_i is decreased and, when ρ_i is greater than 0.44, θ_i is increased.

2.2 Genetic Algorithms

The genetic algorithm we have developed uses continuous variable encoding and has all the features that are normally found in such algorithms, like mutation, recombination and selection (see [6] and [7]). It consists of 5 to 9 populations (running in parallel on different processors) each of 200 chromosomes (300 or 400 in some runs); there is a low migration rate of chromosomes from population to population, each population usually receiving less than one chromosome per generation from the rest of the populations.

Recombination works as follows: pairs of chromosomes are chosen from a population and a crossover operator is applied to them to produce one offspring chromosome; the fitter a chromosome, the more likely it will be chosen for recombination, this being one of the sources of selection pressure in the algorithm. The crossover operators used are one-point, two-point and many-point crossover, each chosen with equal probablity, 1/3, at each recombination event. The offspring replace culled chromosomes and the population size is kept constant from generation to generation; about 10 percent of a population is replaced at each generation; less fit chromosomes are culled with greater probability, another source of selective pressure.

Each chromosome has a number of real-valued loci equal to the number of state variables of the function that is optimized. The values of the variables in the loci are mutated as in equation (1) of the SA algorithm, except that in the GA the θ's are the same for all variables at any time; this means that in the GA mean move size is not individually controlled for each variable. Mutation rates are uniform across loci: chromosomes are mutated at not more than one locus per generation. The fittest chromosome of each population survives unmutated to the next generation, which is also a source of selective pressure.

The fitness of only a fraction (about 1/3) of mutated chromosomes is evaluated at each generation; which means that at any time a number of chromosomes in a population have inaccurate fitness. Half of the populations in a run have high selective pressure for recombination and random outward migration, while the rest have low recombination selection and outward migration of fitter chromosomes. Finally, in our GA each population is replaced in its entirety at regular intervals by new random chromosomes and mutations of the best chromosome encountered in the population up till that stage; a similar feature has been described in [8]. All these features help the algorithm converge faster to better solutions.

3 Biological Problems

In our simulations of biological development and evolution we have used recurrent neural nets to model gene interactions within and between cells; gene product concentrations correspond to neural net node activation levels. Thus within a cell, the concentration $v_a(t)$ of gene a product at time t changes, because of interactions with other genes, by an amount

$$\Delta v_a(t) = R_a g(u_a(t) + h_a) - \lambda_a v_a(t), \tag{4}$$

where $u_a(t)$ is a linear sum of the inputs from genes in the cell itself and from genes in other cells, g is a monotonic, non-linear function, e.g. a sigmoid, R_a is the rate of production of gene a's product, h_a is the threshold for activation of gene a and λ_a is the rate of decay of gene a product. For a comprehensive description of the formal structure of this modeling framework see [1].

The energy functions in these problems are functions S of gene expression over time:

$$E = S(\mathbf{v}(t)), \tag{5}$$

where $\mathbf{v}(t)$ is the vector of concentrations of genes products.

3.1 Life History

In this model we look at the development of a reproducing multicellular organism and at how evolution might adapt its life-history traits (e.g. size, frequency of reproduction, number of offspring) to particular selective environments. The model has been described in [2].

In the simulated environments of the model, opportunities for new organisms to establish themselves and grow (termed *slots*) appear periodically and last for limited amounts of time. An organism starts as a single cell in a slot and grows by cell divisions; cells may be of different types, may die or may become reproductive cells that disperse to occupy other available slots in the environment; organisms die off within a prespecified amount of time from the appearance of their slot (see Fig. 1). Each cell in an organism has a gene network as those described in eqn. 4 (see also eqn. 7) and rules for cell division, cell death, cell dispersal and so on are triggered by gene expression levels in the cell, according to a grammar that the model incorporates; cell type is also determined by gene expression levels (see [1]).

In an evaluation of the energy function, organisms in slots that have appeared over a certain period of time are scored for cell number, proportions of cell types and growth expenditure. The scoring function is of the form

$$E \simeq \sum_{slots} \sum_{time} \left\{ (\frac{n}{n_{target}} - 1)^2 + \sum_a (\frac{n_a}{n} - f_a^{target})^2 + \frac{\Delta n}{n^\gamma} \right\}, \tag{6}$$

where n is the number of cells (size) of an organism in a slot at a given time and n_{target} the desired number of cells, $\frac{n_a}{n}$ is the fraction of cells of type a at that time and f_a^{target} the desired fraction, and $\frac{\Delta n}{n^\gamma}$ the resources spent for producing Δn new cells at this time step, expressed as a fraction of the total resources available to the organism, n^γ, a function of its size with $\gamma \leq 1.0$.

Since organisms in all slots are descendants of the organism in the first slot and since they are all scored according to the same criteria, the optimization gives solutions in which all organisms in the slots are similar (as can be seen in Fig. 1), in effect the solutions are self-reproducing organisms, which is the desired outcome.

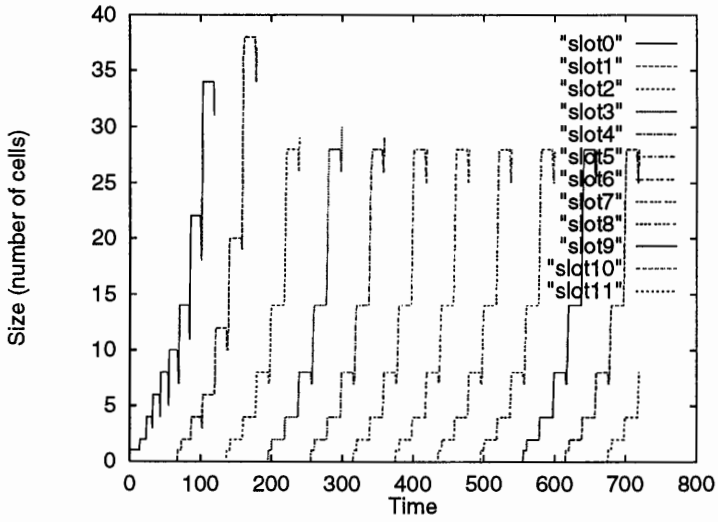

Fig. 1. A self-reproducing organism in the life-history model: the progeny of the organism in the first slot has populated all 12 slots that appeared in the environment. The size of each organism is plotted as a function of time; in this respect and in others, like proportion of cell types and pattern of cell lineages (data not shown), the organisms are similar to each other.

3.2 Neurogenesis

In this problem we try to fit gene expression patterns that are observed in early neurogenesis in *Drosophila*, when neuroblasts and sensory organ precursor (SOP) cells differentiate from epithelial sheets of cells to generate the central nervous system of the *Drosophila* embryo and peripheral sensory organs of the adult fly, respectively (for reviews see [9], [10] and [11]). Several of the genes involved in this specification of cell-fate are expressed in characteristic spatial and temporal patterns during the process. We have constructed a computer model to optimally fit these expression patterns and deduce the gene interactions that could produce them. A similar approach has been used to study gene expression patterns in the *Drosophila* blastoderm, an earlier stage of *Drosophila* development (see [12] and [13]).

Neuroblasts and SOP cells segregate from proneural clusters of cells; clusters consist initially of equivalent cells that all express proneural genes of the *achaete-scute* locus, but eventually only neuroblasts and SOP cells retain *achaete-scute* expression (see Fig. 2). This is the transition our model simulates; cells in the model "express" a small number of genes (corresponding to the genes whose expression patterns we want to fit), which interact as fully connected neural nets, with connection weights depending on the type of interaction. The concentration of gene a product in a particular cell at time t changes according to eqn. (4)

with

$$u_a(t) = \sum_b T_{ab} v_b(t) + \sum_{i \in N} \Lambda^i \sum_b \hat{T}_{ab} \hat{v}^i_b(t), \tag{7}$$

where T is the matrix of gene interactions and $\mathbf{v}(t)$ the vector of gene product concentrations within the cell, $\hat{T}$ is the matrix of gene interactions with neighbouring cells, $\hat{\mathbf{v}}^i(t)$ the vector of gene product conecntrations in neighbouring cell i, N the set of neighbouring cells (in our model we use a hexagonal array for the cell sheet and the neighbourhood of a cell consists of the six surrounding cells) and Λ^i a factor depending on the surface overlap of the cell with neighbouring cell i.

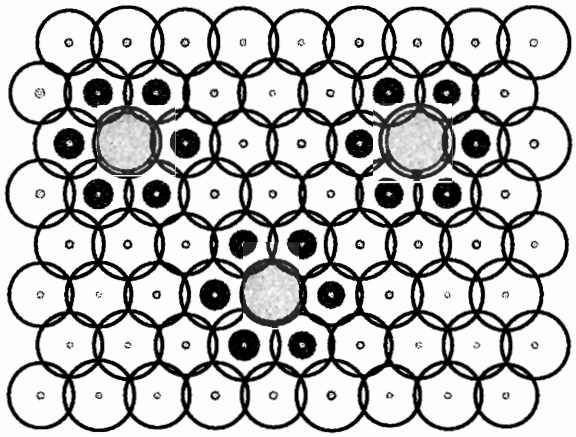

Fig. 2. Frame from a run of the neurogenesis model: neuroblasts segregate from the middle of 3 *achaete-scute* clusters. Cells are represented by circles and gene expression by colored disks, with disk radius proportional to level of expression; *achaete-scute* gene expression is depicted in brown and its overlap with expression of another gene in the simulation is in yellow. The amount of overlap between neighbouring cells determines the strength of interaction of their genes.

We use this model to fit qualitative gene expression pattern datasets (taken from the literature), by finding what matrix values for T and $\hat{T}$ of eqn. (7) minimize the following energy function:

$$E = \sum_{cells, genes, times} (v^i_{aMODEL}(t) - v^i_{aDATA}(t))^2, \tag{8}$$

which is the squared difference between gene product concentrations in the model and those in the dataset summed over all cells and over all gene products and times for which data is available.

3.3 Lissajous Curves

In this problem we try to fit a multi-dimensional curve with a recurrent neural net. We need to find the connection weights that produce an activation pattern of the neural net nodes matching the curve (each node corresponding to one dimension of the curve). The curve is a six-dimensional Lissajous curve; components are sinusoids with the same amplitude and different frequencies. The curve we use for this work is in fact a fit (produced by a recurrent neural net) to an original Lissajous curve (see Fig. 3), i.e. we do a "fit to a fit" (this ensures that the energy function we are optimizing has a global minimum with energy close to zero).

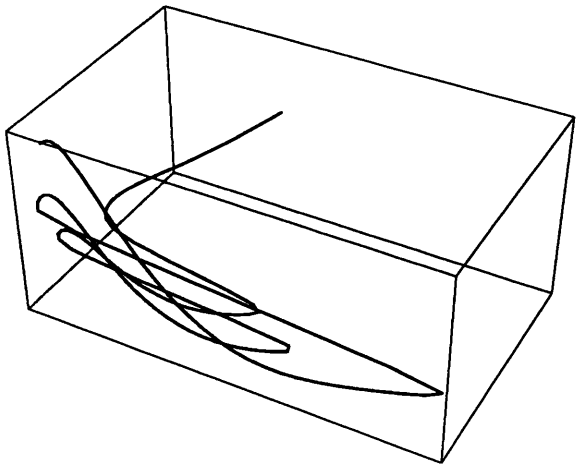

Fig. 3. Projection onto 3 dimensions of a (fit to a) 6-dimensional Lissajous curve, which we try to fit with the node activation pattern of a recurrent neural net.

Although this problem is not strictly biological but more general and abstract, the curve is biologically plausible if one considers that gene product concentrations in a cell may trace such trajectories. In this respect this problem is very similar to the neurogenesis problem described above (Sect. 3.2) restricted to a single cell; in fact, the energy function of the curve problem is the same as that of the neurogenesis problem, eqn. 8 (with the number of cells being one). The proximity of the test curve to the target one is scored many times along the curve which makes this problem more constrained than the other two, despite the fact that it is simpler in its formulation.

4 Results and Discussion

We compared the performance of simulated annealing and the genetic algorithm, in terms of the number of total energy function evaluations, on the three problems described above; the major finding was that, despite the apparent similarity

between the three problems, the performance of the two algorithms varied across problems.

In terms of total energy function evaluations, the GA was about 3 times faster than SA on the life-history problem (Fig. 4), or about 20 times faster in clock time, since the GA was implemented in parallel; the advantage of the GA over SA in this problem was maintained for various parameter values of the two algorithms. SA was many times faster than the GA on the neurogenesis problem

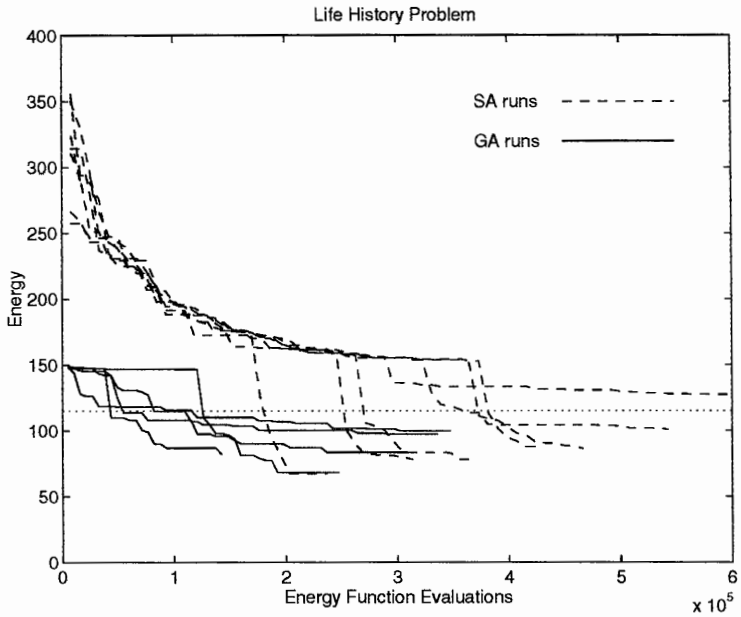

Fig. 4. Life-history optimization runs - Energy plotted against the number of total energy function evaluations. GA runs reach low energy levels about three times faster than SA runs. The dotted line indicates energy levels below which solutions are considered successful for the purposes of the biological model.

(Fig. 5) for wide ranges of parameter values of the algorithms. SA was faster even in terms of clock time on this problem. On the curve-fitting problem SA was again faster than the GA in reaching low energy levels that corresponded to the best solutions (Fig. 6), although the GA had an advantage early in the runs at higher energy levels.

It is not clear why the performance of the two optimization algorithms varied so much across models that are so similar in structure. It is conceivable that, because both algorithms have a large number of parameters that can significantly affect their performance, we may have not found algorithm parameters that result in optimal performance on each problem, despite our efforts. Also, the neurogenesis and curve-fitting problems had considerably fewer variables than

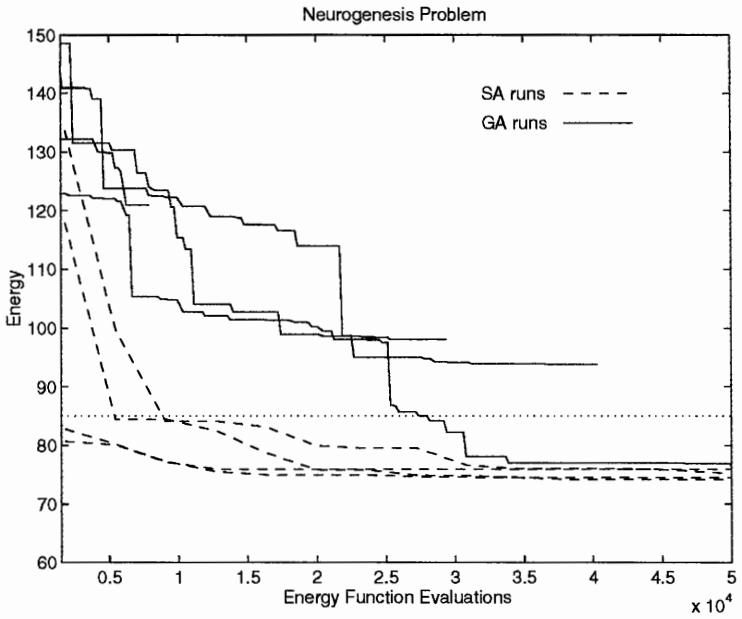

Fig. 5. Neurogenesis optimization runs - SA has a large advantage over GA in this problem. Same conventions as in Figure 4.

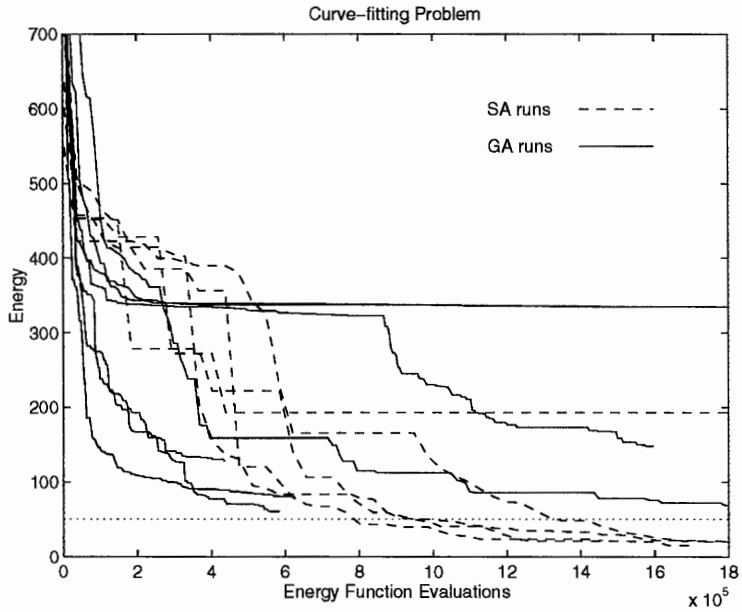

Fig. 6. Curve-fitting optimization runs - SA performs better than GA, although GA has a slight advantage early in the runs. Same conventions as in Figure 4.

the life-history one in most of the runs we did; but, again, it is unlikely that this is the main reason for differences in performance: increasing or decreasing the number of variables in each problem did not seem to affect the relative performance of the algorithms.

The energy functions of the neurogenesis and curve-fitting problems differed from the life-history one; the former were smooth functions of neural net node activations (squared sums of differences between actual and desired activation levels), whereas the latter had discontinuities, since it scored non-continuous features, like cell number and discrete cell types. It is possible that SA is better for optimizing smooth functions while GAs have an advantage with discontinuous functions.

5 Acknowledgements

This work was partially supported by the Yale Institute for Biospheric Studies (Center for Computational Ecology), the Neuroengineering and Neuroscience Center at Yale and the Yale Center for Parallel Supercomputing.

References

1. Mjolsness, E., Sharp, D.H. and Reinitz, J.: A connectionist model of development. J.Theor.Biol. **152** (1991) 429–453
2. Mjolsness, E., Garrett, C.D., Reinitz, J. and Sharp, D.H.: Modeling the connection between Development and evolution: Preliminary report. In *Evolution and biocomputation, Computational models of evolution*, Banzhaf, W. and Eeckman, F.H. (eds.), Springer, Berlin, 1995, 103–122
3. Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P.: Optimization by simulated annealing. Science **220** (1983) 671–680
4. Lam, J. and Delosme, J.-M.: An efficient simulated annealing schedule: Derivation. Technical Report 8816, Yale Electrical Engineering Department, New Haven, CT.
5. Lam, J. and Delosme, J.-M.: An efficient simulated annealing schedule: Implementation and evaluation. Technical Report 8817, Yale Electrical Engineering Department, New Haven, CT.
6. Holland, J.H., *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975
7. Goldberg, D.E., *Genetic algorithms in search, optimization and machine learning.* Addison-Wesley, Reading, 1989
8. Mathias, K.E. and Whitley, L.D.: Changing representations during search: a comparative study of delta coding. Evolutionary Computation **2** (1994) 249–278
9. Artavanis-Tsakonas, S. and Simpson, P.: Choosing a cell fate: a view from the *Notch* locus. Trends Genet. **7** (1991) 403–408
10. Campuzano, S. and Modollel, J.: Patterning of the *Drosophila* nervous system - the *achaete-scute* gene complex. Trends Genet. **8** (1992) 202–208
11. Campos-Ortega, J.A.: Early neurogenesis in Drosophila melanogaster. In *The development of Drosophila melanogaster*, Bate, M. and Martinez-Arias, A. (eds.), Cold Spring Laboratory Press, 1993, 1091–1129

12. Reinitz, J., Mjolsness, E. and Sharp, D.H.: Model for cooperative control of positional information in *Drosophila* by Bicoid and maternal Hunchback. J.Exp.Zool. **271** (1995) 47–56
13. Reinitz, J. and Sharp, D.H.: Mechanism of *eve* stripe formation. Mech.Dev. **49** (1995) 133–158