Proceedings of the 1998 Symposium on Applied Computing, New York: The Association for Computing Machinery, 1998.

SCALABILITY OF AN MPI-BASED FAST MESSY GENETIC ALGORITHM

Laurence D. Merkle
Center for Plasma Theory and Computation
Air Force Research Laboratory
Kirtland AFB, OH 87117
merklel@plk.af.mil

George H. Gates, Jr.
Air Force Research Laboratory
WL HPC Technical Focal Point
Wright-Patterson AFB, OH 45433
gatesgh@asc.hpc.mil

Gary B. Lamont
Department of Electrical and Computer Engineering
Graduate School of Engineering
Air Force Institute of Technology
Wright-Patterson AFB, OH 45433
lamont@afit.af.mil

Key Words: Parallel Genetic Algorithms, Messy Genetic Algorithms, Polypeptide Structure Prediction, Fixed Solution Quality, Population Sizing.

ABSTRACT

The fast messy genetic algorithm (fmGA) belongs to a class of algorithms inspired by the principles of evolution, known appropriately as "evolutionary algorithms" (EAs). These techniques operate by applying biologically-inspired operators, such as recombination, mutation, and selection, to a population of individuals. EAs are frequently applied as optimum seeking techniques, by way of analogy to the principle of "survival of the fittest." In contrast to many EAs, the fmGA consists of several evolutionary phases, each with distinct characteristics of local/global computation. These are explained in the paper.

Previous scalability analyses of island-model EAs have been based on either fixed global population size or fixed subpopulation size. Recently developed population sizing theory enables scalability analysis based on fixed expected solution quality.

Parallel computational experiments are performed to determine the effectiveness and efficiency of an MPI-based fmGA using each of these scaling techniques. The optimization problem for these experiments is the minimization of the CHARMM energy model of the pentapeptide [Met]-Enkephalin.

1998 ACM 0-89791-969-6/98/0002

1 INTRODUCTION

The fast messy genetic algorithm (fmGA) belongs to a class of algorithms inspired by the principles of evolution, known appropriately as "evolutionary algorithms" (EAs). These techniques operate by applying biologically-inspired operators, such as recombination, mutation, and selection, to a population of individuals. EAs are frequently applied as optimum seeking techniques, by way of analogy to the principle of "survival

of the fittest." EAs are discussed in more detail elsewhere (see, for example, Bäck [1]).

In contrast to many EAs, the fmGA consists of several evolutionary phases, each with distinct characteristics of local and global computation. These are explained in Section 2, along with the MPI-based island-model fmGA implementation used in this research.

Previous scalability analyses of island-model EAs have been based on either fixed global population size or fixed subpopulation size. Recently developed population sizing theory [3] enables scalability analysis based on fixed expected solution quality (Section 3). Parallel computational experiments are performed to determine the effectiveness and efficiency of an MPI-based fmGA using each of these scaling techniques (Section 4).

The results of these experiments are presented in Section 5, followed by conclusions and recommendations in Section 6. Finally, the optimization problem for these experiments — the minimization of the CHARMM energy model of the pentapeptide [Met]-Enkephalin — is discussed in detail in Appendix A.

2 FAST MESSY GA

In contrast to the uniform-length binary string representation scheme of the widely known simple genetic algorithm (sGA) [7], the fmGA scheme is order-invariant and such that individuals are not necessarily of uniform length [9]. Also in contrast to the sGA, the fmGA algorithm consists of distinct phases, and is typically applied iteratively. Successive iterations are performed varying the *building block size* k. Larger values of k imply better expected solution quality at the cost of greater execution time and required memory. Thus, the upper limit of the iteration controls a tradeoff between computational resources and solution quality. The three phases of each iteration, which are illustrated in Figure 1, are the *initialization*, *primordial*, and *juxtapositional* phases.

- The initialization phase consists of a random sampling of individuals of length $\ell' = \ell - k$, where ℓ is the number of discrete optimization variables. Probabilistically Complete Initialization (PCI) is a method of generating the initial population.
- The goal of the primordial phase is to obtain a population containing individuals of length k which can with high probability be juxtaposed to obtain an optimal individual. Tournament selection focuses the search on highly fit individuals, while building block filtering (BBF) is a mutation operator which is used to periodically reduce the lengths of the individuals.
- The juxtapositional phase uses cut-and-splice (a recombination operator) to construct highly fit individuals of length ℓ from the length k individuals surviving the primordial phase. Tournament selection is used to focus the search on highly fit combinations.

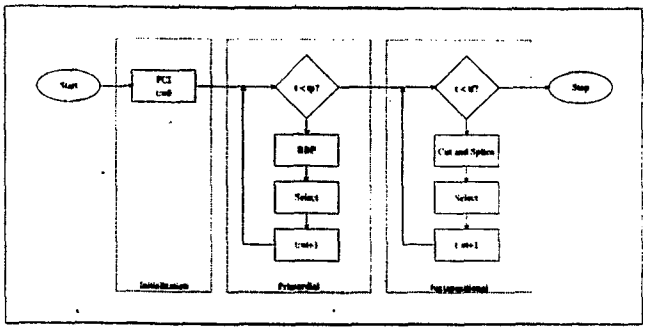

Figure 1: Fast Messy Genetic Algorithm Flow Chart

In both the primordial and juxtapositional phases, a locally optimal solution, called the *competitive template*, is used to "fill in the gaps" in partially specified solutions to allow their evaluation. Also, in order to prevent the cross-competition between building blocks caused by non-uniform scaling, competition is restricted to those individuals which are defined at some threshold number of common loci.

PCI generates a population of random individuals in which each building block has an expected number of copies sufficient to overcome sampling noise. Each individual in the population is defined at $\ell' = \ell - k$ loci, which are selected randomly without replacement (it is assumed that $k \ll \ell$). The population size is

$$N = n_g n_a, \quad \text{where} \quad n_g = \frac{\binom{\ell-k}{\ell'-k}}{\binom{\ell}{\ell'}} = \frac{\ell!(\ell-2k)!}{(\ell-k)!^2}, \qquad n_a = 2z_\alpha^2 \beta^2 (m-1) 2^k, \quad (1)$$

m is the number of building blocks in a fully specified solution, α is a parameter specifying the probability of selection error between two competing building blocks, $P[Z \geq z_\alpha] = 1 - \alpha$ for Z a standard normal random variable, and β^2 is a parameter specifying the maximum inverse signal-to-noise ratio per subfunction to be detected [9].

The fast messy GA primordial phase enriches the initial population via alternating tournament selection and building block filtering (BBF). Tournament selection increases the proportion of individuals containing highly fit building blocks. BBF then randomly deletes some number of genes from every individual, the number being chosen so that BBF is expected to disrupt many but not all of the highly fit building blocks. Those individuals still containing highly fit building blocks receive additional copies in subsequent iterations of tournament selection. The net effect is to produce a population of partial strings of length k with a high expected proportion of highly fit building blocks.

Competition is restricted to those pairs of individuals that contain a specified number θ of common defining loci. The threshold θ for each generation is specified as an input parameter. Current practice is to use an empirically determined filtering and thresholding schedule, although theoretical work is in progress to allow *a priori* schedule design [13, 17]. Frequently, in order to obtain good effectiveness, it is necessary to use quite restrictive thresholding. In this case, the algorithmic complexity of selection is dominated by the $O(N^2)$ search for pairs that satisfy the threshold, where N is the population size.

A parallel fast messy GA (pfmGA) is designed based on the island model [10], and implemented in C on the IBM SP2. A "controller" processor inputs the GA parameters, creates a competitive template, and broadcasts them to the remaining processors. Then, each processor (including the controller processor) independently performs PCI to generate and evaluate an initial population of N individuals. Because N depends on building block size, it is determined independently for each iteration.

For these experiments, each processor (including the controller) independently performs tournament selection on its subpopulation. Thus, selection does not require communication. The search trajectories resulting from this strategy, called *local selection*, depend fundamentally on the number and size of the subpopulations, and hence on the processor count. Following selection, each processor (including the controller) performs BBF and function evaluation for its subpopulation.

For these experiments, each processor (including the controller) independently applies tournament selection and cut-and-splice to its local population (i.e. each processor's initial juxtapositional phase population is identical to its final primordial phase population).

At the end of each iteration (i.e. at the end of each juxtapositional phase), each processor other than the controller sends its best solution to the controller. The controller determines the overall best solution, which becomes the competitive template for the next iteration, and reports execution statistics.

3 POPULATION SIZING

A key element of fmGA theory is the sizing of the initial population, as described by Equation 1, which is based on the theory proposed by Goldberg, et al. [8]. Specifically, the population size is calculated based on the probability $p = \Pr[Z \leq z]$ of correct decision making, where Z is a standard normal random variable, and for a "signal" d to be correctly detected in the presence of noise σ_M^2, given n' expected samples,

$$z^2 = \frac{d^2}{2\sigma_M^2/n'} \quad . \tag{2}$$

It is assumed that the ultimate success or failure of the algorithm with respect to a particular subfunction is determined by its success or failure in the first generation. This assumption leads to a simple GA population size of n_a as given in Equation 1.

More recently, Harik, et al. proposed a population sizing model based on random walk theory [11]. The number of correct copies of a building block for a particular subfunction is modeled as a random walk in one-dimensional space. The initial condition is given by the expected number of copies in a random population, and absorbing barriers exist corresponding to the extinction and takeover events. This model leads to the approximate probability of ultimate success

$$P_N \approx 1 - \left(\frac{1-p}{p}\right)^{N/2^k} \tag{3}$$

given a population of size N, a probability of 2^{-k} that a random individual contains the correct building block, and a probability p of correct decision making.

Based on this model, Cantú-Paz, et al. proposed a model for parallel GA subpopulation sizing [3]. The parallel GA is assumed to succeed with respect to a particular subfunction provided that at least one subpopulation does. After a lengthy derivation involving several approximations, the recommended subpopulation size for a parallel GA with r subpopulations, and a fitness function having m subfunctions, is

$$n_d = \frac{2^k \ln(1 - P_{bb}^{*})}{\ln((1-p)/p)} \quad , \tag{4}$$

where P^*_{bb} satisfies

$$\frac{b - mP^*_{bb}}{\sqrt{mP^*_{bb}(1 - P^*_{bb})}} = \sqrt{-\frac{\pi}{2}\ln(1 - (1 - 2(1 - 2^{-1/r}))^2)} \tag{5}$$

4 EXPERIMENTAL DESIGN

This section describes computational experiments which evaluate the performance of an MPI-based IBM SP2 island model [10] pfmGA implementation. Specifically, experiments are conducted to determine the

- effectiveness (overall minimum energy),
- speedup,
- (absolute) efficiency,
- fixed-subpopulation-size scaled efficiency, and
- fixed-expected-solution-quality scaled efficiency.

Absolute efficiency is measured by fixing the global population size while varying the processor count. Specifically, the block size k is iterated from 1 to 4, and the global population size N_k is determined strictly by Equation 1, where

- the string length $\ell = 240$, as determined by the optimization problem (see Appendix A);
- $z_\alpha^2 = 6$, corresponding to a probability of selection error $\alpha \approx 1\%$;
- the inverse signal-to-noise ratio $\beta^2 = 0.25$, as determined by Gates' technique for ordinal-based selection methods [6]; and
- for each iteration, the number of building blocks $m = \ell/k$.

The population sizes for each block size are shown in Table 1.

Scaled efficiency for fixed-subpopulation-size is measured by fixing the subpopulation sizes while varying the processor count. The subpopulation size for each iteration of k is N_k/P_0, where $P_0 = 4$ is chosen based on the effectiveness observed in the absolute efficiency experiments (see Section 5).

Fixed-expected-solution-quality scaled efficiency is measured by choosing the subpopulation sizes for each processor count using the population sizing theory of Section 3. The probabilities of correct decision making p are estimated by assuming that the signal to be detected is $(f_{max} - f_{min})/2^{2k}$, using an empirically determined fitness variance to compute the signal to noise ratio, and evaluating the cdf of a standard normal random variable (see Table 2). The resulting sGA subpopulation sizes $n_d(r,k)$, pfmGA subpopulation sizes $N_{fq}(r) = \sum_k n_g(k) n_d(r,k)$, and pfmGA global population sizes $rN_{fq}(r)$ are shown in Table 3.

The optimization problem for these experiments is the minimization of the CHARMM energy model [19] of the pentapeptide [Met]-Enkephalin. Previous research efforts have established [Met]-enkephalin conformations which are known to be low in energy, as measured by the CHARMM energy model [12, 18]. Appendix A describes the general polypeptide conformation problem in more detail.

Table 1: Population sizes for absolute efficiency and fixed subpopulation size scaled efficiency experiments.

k	N_k	N_k/P_0
1	1440	360
2	1452	363
3	1968	492
4	3028	757

Table 2: Estimated probabilities of correct decision making as a function of block size.

k	1	2	3	4
p	0.993763	0.694879	0.549305	0.512477

Table 3: Population sizes for fixed-expected-solution-quality experiments.

	n_d					
r	$k=1$	$k=2$	$k=3$	$k=4$	$N_{fq}(r)$	$rN_{fq}(r)$
1	16	43	220	1473	1473	1473
2	15	41	209	1403	1668	3336
4	14	37	190	1258	1499	5996
8	13	34	171	1130	1348	10784
16	12	32	156	1027	1227	19632
32	11	29	145	946	1131	36192
64	11	28	136	880	1055	67520

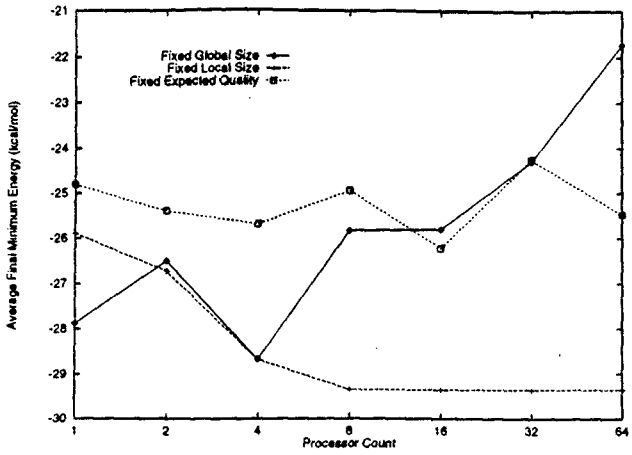

Figure 2: Average final minimum energy vs. processor count

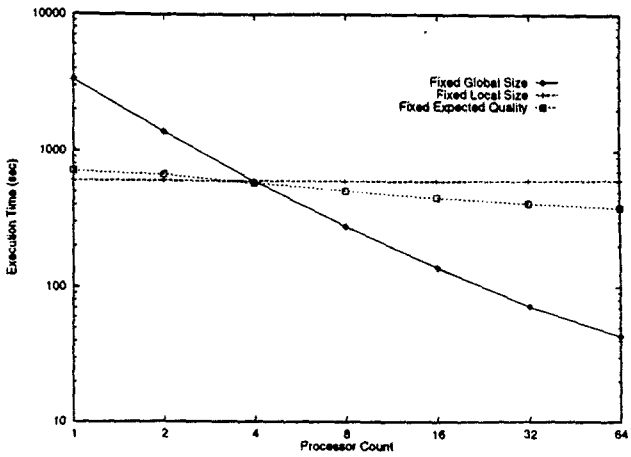

Figure 3: Average execution time vs. processor count

In all cases, results are averaged over five executions using independent pseudo-random number sequences.

5 RESULTS AND ANALYSIS

This section discusses the observed performance of the pfmGA. The final minimum energy obtained (averaged over 5 independent runs) is shown in Figure 2 as a function of processor count for each of the three scaling methods. As is commonly observed in parallel GAs with fixed global population size, there is no clear relationship between the effectiveness and processor count. This is because of the competing effects of decreasing subpopulation size and increasing subpopulation count. The former decreases the effectiveness of each subpopulation, while the latter increases the effectiveness of the overall algorithm. The net effect is difficult to predict.

In contrast, for fixed subpopulation size (and independent subpopulations), the effectiveness increases monotonically with processor count. In this case, the subpopulation size is chosen to guarantee effectiveness no worse than the best fixed global population size experiment.

The final minimum energies obtained in the fixed expected solution quality experiments are similar, but not identical. This is due partly to the approximate nature of the population sizing model, as well as the relatively small number of executions per case for these experiments, but it is mostly due to the unavoidable error in estimating the parameters for the model. Specifically, the exact signal to be detected is unknown. Importantly, the effectiveness is more consistent than that for the fixed global population size experiments.

The execution time (again averaged over 5 independent runs) is shown in Figure 3 as a function of processor count r for each scaling method. The results are as expected. For fixed global population size and the processor counts used in these experiments, subpopulation size and execution time are inversely proportional to processor count. For very large processor counts, communication overhead appears as a modest fraction of the total execution time. For fixed subpopulation size (and independent subpopulations), execution time is essentially constant. For fixed expected solution quality, the subpopulation size and execution time decrease roughly linearly with $\log r$.

Finally, the scaled efficiency (again averaged over 5 independent runs) is shown in Figure 4 as a function of processor count r for each scaling method. The scaled efficiency E is calculated using global population size N as the problem size:

$$E(r) = \frac{N(r)}{N(1)} \frac{T(1)}{r \cdot T(r)} \quad , \tag{6}$$

where $T(i)$ is the execution time per processor on i processors, and $N(i)$ is the global population size on i processors. Efficiencies greater than one are observed because of the $O(N^2 r^{-}2)$ complexity of the tournament selection algorithm (i.e. the time spent performing tournament selection decreases quadratically with subpopulation size). This effect is most dramatic in the fixed global population size experiments, for which the subpopulation size varies significantly. For more than 16 processors, the subpopulation size is small enough that the effect is diminished by the increased computational and communication overhead, so that scaled efficiency begins to decrease. Scaled efficiency is essentially constant for fixed population sizes, and increases slightly with processor count for the fixed expected solution quality experiments.

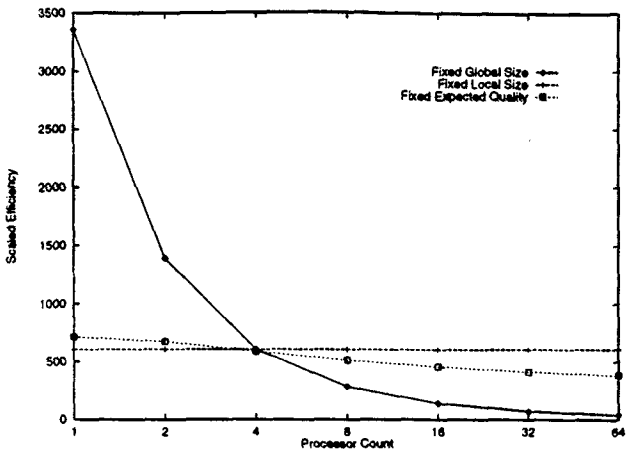

Figure 4: Scaled efficiency

6 CONCLUSIONS

The performance (effectiveness and scaled efficiency) of an MPI-based pfmGA for polypeptide structure prediction is determined experimentally on the IBM SP2. Three methods of scaling the algorithm are considered: fixed global population size, fixed subpopulation size, and fixed expected solution quality.

The population sizes for the latter experiments are determined using a population sizing model proposed recently by Cantú-Paz, et al. Because of the error in estimating the parameters for the population sizing model, the solution quality in the latter experiments is only approximately constant. The fixed expected solution quality method is shown to represent a compromise between the parallel efficiency of the fixed global population size method and the monotonically increasing effectiveness of the fixed subpopulation size method.

We wish to acknowledge the use of the DoD High Performance computing facilities of the Aeronautical System Center (ASC) Major Shared Resource Center (MSRC) at Wright-Patterson AFB in support of this research.

A PROTEIN STRUCTURE PREDICTION

This section discusses the objective function associated with the polypeptide energy minimization application, as well as the binary encoding scheme used in these experiments. The capability to predict a polypeptide's molecular structure given its amino acid sequence is important to numerous scientific, medical, and engineering applications [4]. The effort to develop a general technique for such structure prediction is commonly referred to as the *protein folding problem*, where the minimization of an energy function in conformational space is required [22].

A.1 Objective Function

For these experiments, conformations (i.e., the relative positions of the atoms comprising a molecule) are represented using *internal coordinates*. That is, the position of atom, i is specified by a *bond length*, a *bond angle*, and a *dihedral angle*, each with respect to an appropriate number of neighboring atoms. Based on physical insight, a subset of the dihedral angles are chosen as the independent variables for the optimization process.

The objective function, which is to be minimized, is based on the CHARMM [2] energy function

$$\begin{aligned} E = & \sum_{(i,j)\in\mathcal{B}} K_{r_{ij}}(r_{ij}-r_{eq})^2 + \\ & \sum_{(i,j,k)\in\mathcal{A}} K_{\Theta_{ijk}}(\Theta_{ijk}-\Theta_{eq})^2 + \\ & \sum_{(i,j,k,l)\in\mathcal{D}} K_{\Phi_{ijkl}}[1+\cos(n_{ijkl}\Phi_{ijkl}-\gamma_{ijkl})] + \\ & \sum_{(i,j)\in\mathcal{N}} \left[\left(\frac{A_{ij}}{r_{ij}}\right)^{12} - \left(\frac{B_{ij}}{r_{ij}}\right)^{6} + \frac{q_i q_j}{4\pi\varepsilon r_{ij}}\right] + \\ & \frac{1}{2}\sum_{(i,j)\in\mathcal{N}'} \left[\left(\frac{A_{ij}}{r_{ij}}\right)^{12} - \left(\frac{B_{ij}}{r_{ij}}\right)^{6} + \frac{q_i q_j}{4\pi\varepsilon r_{ij}}\right] \end{aligned} \quad (7)$$

where the five terms (which we denote $E_{\mathcal{B}}$, $E_{\mathcal{A}}$, $E_{\mathcal{D}}$, $E_{\mathcal{N}}$, $E_{\mathcal{N}'}$) represent the energy due to bond stretching, bond angle deformation, dihedral angle deformation, non-bonded interactions, and 1-4 interactions, respectively. Specifically,

- $\mathcal{B}$ is the set of bonded atom pairs,
- $\mathcal{A}$ is the set of atom triples defining bond angles,
- $\mathcal{D}$ is the set of atom 4-tuples defining dihedral angles,
- $\mathcal{N}$ is the set of non-bonded atom pairs,
- $\mathcal{N}'$ is the set of 1-4 interaction pairs,
- r_{ij} is the distance between atoms i and j,
- Θ_{ijk} is the angle formed by atoms i, j, and k,

- Φ_{ijkl} is the dihedral angle formed by atoms i, j, k, and l,
- q_i is the partial atomic charges of atom i,
- the $K_{r_{ij}}$'s, r_{eq}'s, $K_{\Theta_{ijk}}$'s, Θ_{eq}'s, $K_{\Phi_{ijkl}}$'s, γ_{ijkl}'s, A_{ij}'s, B_{ij}'s, and ϵ are empirically determined constants (taken from the QUANTA parameter files).

In Equation 7, internal energy is expressed as a function of both the internal coordinates and the inter-atomic distances. Thus, in order to calculate the energy (and hence the fitness) of the conformation encoded by an individual, it is necessary to calculate its Cartesian coordinates from its internal coordinates. We use the transformation method proposed by Thompson [21]. This method requires at most one 4×4 matrix multiplication per atom per conformation.

A.2 Encoding Scheme

Each individual is a fixed length binary string encoding the independent dihedral angles of a polypeptide conformation. The decoding function used is the affine mapping $D : \{0,1\}^{10} \rightarrow [-\pi, \pi]$ of 10 bit subsequences to dihedral angles such that

$$D(a_1, a_2, \ldots, a_{10}) = -\pi + 2\pi \sum_{j=1}^{10} a_j 2^{-j}. \qquad (8)$$

This encoding yields a precision of approximately one third of one degree. In these experiments, the 24 ϕ, ψ, ω, and χ dihedral angles of the pentapeptide [Met]-Enkephalin are the independent variables for optimization, hence the string length is 240. [Met]-Enkephalin is chosen because it has been used as a test problem for many other energy minimization investigations (e.g. [15, 20]), and its minimum energy conformation is known (with respect to the ECEPP/2 energy model)[16].

References

[1] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice.* Oxford University Press, New York, 1996.

[2] Bernard R. Brooks et al. CHARMM: A program for macromolecular energy, minimization, and dynamic calculations. *Jounal of Computational Chemistry*, 4(2):187–217, 1983.

[3] Erick Cantú-Paz and David E. Goldberg. Modeling idealized bounding cases of parallel genetic algorithms. In Koza et al. [14], pages 353–361.

[4] Hue Sun Chan and Ken A. Dill. The protein folding problem. *Physics Today*, pages 24–32, February 1993.

[5] Stephanie Forrest, editor. *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo CA, July 1993. Morgan Kaufmann Publishers, Inc.

[6] George H. Gates, Jr. Predicting protein structure using parallel genetic algorithms. Master's thesis, Graduate School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH 45433, December 1994.

[7] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Publishing Company, Reading MA, 1989.

[8] David E. Goldberg, Kalyanmoy Deb, and James H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6:333–362, 1992.

[9] David E. Goldberg, Kalyanmoy Deb, Hillol Kargupta, and Georges Harik. Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. In Forrest [5], pages 56–64.

[10] V. Scott Gordon and Darrell Whitley. Serial and parallel genetic algorithms as function optimizers. In Forrest [5], pages 177–183.

[11] Georges Harik, Erick Cantú-Paz, David E. Goldberg, and Brad L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. pages 7–12, New York, 1997. IEEE Press.

[12] Charles E. Kaiser, Jr., Laurence D. Merkle, Gary B. Lamont, George H. Gates, Jr., and Ruth Pachter. Case studies in protein structure prediction with real-valued genetic algorithms. In Michael Heath, Virginia Torczon, Greg Astfalk, Petter E. Bjørstad, Alan H. Karp, Charles H. Koelbel, Vipin Kumar, Robert F. Lucas, Layne T. Watson, and David E. Womble, editors, *Proceedings of the Eighth SIAM Conference on Parallel Processing for Scientific Computing,*, Philadelphia, PA, 1997. SIAM, Society for Industrial and Applied Mathematics.

[13] Hillol Kargupta. *SEARCH, Polynomial Complexity, and the Fast Messy Genetic Algorithm.* PhD thesis, University of Illinois, 1995. Also available as IllliGAL Report No. 95008.

[14] John Koza et al., editors. *Genetic Programming 1997: Proceedings of the Second Annual Conference*, San Francisco, 1997. Morgan Kauffman Publishers.

[15] Scott M. LeGrand and Kenneth M. Merz Jr. The application of the genetic algorithm to the minimization of potential energy functions. *Journal of Global Optimization*, 3:49–66, 1991.

[16] Zhenqin Li and Harold A. Scheraga. Monte carlo-minimization approach to the multiple-minima problem in protein folding. *Proceedings of the National Academy of Science USA*, 84:6611–6615, 1987.

[17] Laurence D. Merkle. *Analysis of Linkage-Friendly Genetic Algorithms*. PhD thesis, Graduate School of Engineering, Air Force Institute of Technology (AU),.Wright-Patterson AFB OH 45433, December 1996.

[18] Laurence D. Merkle, Robert L. Gaulke, George H. Gates, Jr., Gary B. Lamont, and Ruth Pachter. Hybrid genetic algorithms for polypeptide energy minimization. In *Applied Computing 1996: Proceedings of the 1996 Symposium on Applied Computing*, New York, 1996. The Association for Computing Machinery.

[19] Molecular Simulations, Incorporated. *CHARMm version 22.0 Parameter File*, 1992.

[20] Akbar Nayeem, Jorge Vila, and Harold A. Scheraga. A comparative study of the simulated-annealing and Monte Carlo-with-minimization approaches to the minimum-energy structures of polypeptides: [Met]-Enkephalin. *Journal of Computational Chemistry*, 12(5):594–605, 1991.

[21] H. Bradford Thompson. Calculation of cartesian coordinates and their derivatives from internal molecular coordinates. *The Journal of Chemical Physics*, 47(9):3407–3410, November 1967.

[22] Maximiliano Vásquez, G. Némethy, and H. A. Scheraga. Conformational energy calculations on polypeptides and proteins. *Chemical Reviews*, 94:2183–2239, 1994.