An Estimation of Distribution Algorithm Based on Maximum Entropy

Alden Wright[1], Riccardo Poli[2], Chris Stephens[3], W.B. Langdon[4], and Sandeep Pulavarty[1]

[1] Computer Science, University of Montana, Missoula, MT, 59812, USA
[2] Department of Computer Science, University of Essex, Colchester, UK
[3] Instituto de Ciencias Nucleares, UNAM, Mexico DF 04510
[4] Computer Science, University College, London, Gower Street, London, UK

Abstract. Estimation of distribution algorithms (EDA) are similar to genetic algorithms except that they replace crossover and mutation with sampling from an estimated probability distribution. We develop a framework for estimation of distribution algorithms based on the principle of maximum entropy and the conservation of schema frequencies. An algorithm of this type gives better performance than a standard genetic algorithm (GA) on a number of standard test problems involving deception and epistasis (i.e. Trap and NK).

1 Introduction

Genetic algorithms maintain a population of potential solutions to a problem. Each potential solution is assumed to have a fitness which is related to how well it solves the problem. The population undergoes selection alternated with crossover and/or mutation. Selection increases the frequency of high-fitness solutions and decreases the frequency of low-fitness ones, but does not add new solutions to the population. Crossover and mutation extend the sampling of the search space by creating new solutions from those currently in the population.

Estimation-of-distribution algorithms take a different approach to sample the search space. The population is used to estimate a probability distribution over the search space that reflects what are considered to be important characteristics of the population. A new population is then generated by sampling this distribution. This introduces diversity into the new population as well as preserving salient characteristics of the previous one. Thus, the sampling process usually replaces the crossover and mutation operators of the genetic algorithm. Estimation of distribution algorithms are reviewed in [1]. The MIMIC algorithm of [2] uses only second-order statistics and the Kullback-Leibler divergence.

The maximum entropy principle [3] has its historical roots in physics and is a widely used approach to estimating probability distributions from data. The key idea is that the associated probability distribution should agree with what is known but, apart from that, should express 'maximum uncertainty'. This allows the maximum possible freedom to adjust the distribution when new data become known. In practice observed data is used to derive a set of constraints on the

K. Deb et al. (Eds.): GECCO 2004, LNCS 3103, pp. 343–354, 2004.

estimated probability distribution. The entropy of this estimated distribution is then maximized subject to these constraints.

We assume a length-ℓ string representation, where the alphabet of symbols at any string position is an arbitrary finite set. A schema is a subset of the search space where the symbols at some string positions are defined (fixed) and at other string positions are arbitrary (variable). A schema is commonly denoted by a string of length ℓ where the special asterisk symbol $*$ is used to denote that any string symbol is possible at that position. The order of a schema is the number of defined (non-asterisk) positions. Given a frequency distribution over the search space and a schema, the corresponding schema frequency is just the sum of the relative frequencies of the elements of that schema.

One traditional way of "explaining" how genetic algorithms work is the building block hypothesis. This says that the GA builds high-fitness solutions by combining building blocks which are high-fitness, short, low-order schemata that can be thought of as representing partial solutions to the problem. This point of view came from an analysis of Holland's Schema Theorem [4] which, being an inequality, did not take into account the effects of schema construction. More recent exact schema theorems [5,6,7] have shown that, indeed, genetic algorithms "work" by combining building blocks. However, the relevant building blocks are dynamical, not static, of high "effective fitness" [5], not necessarily of high fitness, and, in general, are not short [8]. Although the traditional building block hypothesis is only a first order approximation, except under certain restricted circumstances, exact schema theorems can be used to understand the role of building blocks in a more rigorous way which does aid in our understanding of how these algorithms work.

Our approach is to use schema frequencies in a given population (after selection) to constrain the estimated probability distribution. The entropy of this distribution is subsequently maximized and the distribution sampled to produce a new population. The fitness of individuals in the new population is measured in the normal way. Schema frequencies are used because they are the most natural coarse-grained quantities associated with a population of strings. Since schema frequencies are sums of string frequencies, they should give more reliable estimates of properties of the fitness function than string frequencies. In addition, since the state of the algorithm can be completely described by schema frequencies, this should lead to tractable models of this class of algorithms.

On average, sampling from a maximum entropy distribution constrained by low-order schema frequencies preserves these schema frequencies in the new population. Therefore, we suggest that our algorithm will work well in problems where preservation of the building blocks corresponding the low-order schemata used in the algorithm is important. One might also imagine an extension of the algorithm so as to take into account the dynamical evolution of building blocks by changing dynamically the schemata whose frequency will be fixed.

We can now give an outline of the proposed algorithm.

1. Generate an initial random population.
2. Generate an initial collection of schemata.
3. Apply selection (as in a GA) to the population.

4. Calculate the frequencies of the schemata in the schema collection.
5. Sample the maximum entropy distribution to obtain a new population.
6. Go to step 3.

Steps 2 will be elaborated in section 3. Naturally, one should expect the performance of the algorithm to be sensitive to the schemata chosen in step 2.

2 Sampling from a Maximum Entropy Distribution Constrained by Schema Family Frequencies

In this section we give some basic definitions and then show how to sample the maximum entropy distribution constrained by the frequencies of an appropriate collection of schema families. We will use the term **maxent distribution** as shorthand for maximum entropy distribution. The results given in this section are proved in the appendix.

2.1 Entropy

If X is a finite set and $p(x)$ is a probability distribution over X, then the entropy of p is defined to be

$$S(X) = \sum_{x \in X} -p(x) \log(p(x))$$

where the base of the logarithm doesn't matter as long as one is consistent. We will assume base 2.

A **schema constraint**, (Q, a), on $p(x)$ consists of a set $Q \subset X$ and a real number $a \in [0, 1]$ such that $\sum_{x \in Q} p(x) = a$. A **family of schema constraints** will be given by a pair $(\mathcal{T}, g)$, where $\mathcal{T}$ is a collection of schemata of X and g is a function from $\mathcal{T}$ to the unit interval $[0, 1]$. Each schema constraint is of the form $(Q, g(Q))$ for some $Q \in \mathcal{T}$. The family $(\mathcal{T}, g)$ is **feasible** if there is some probability distribution that satisfies all of the schema constraints. If for each $Q \in \mathcal{T}$, $g(Q)$ is defined to be the frequency of Q in a given population, then $\mathcal{T}$ is feasible.

Lemma 1. Equal probability rule. *If $p(x)$ is the maximum entropy distribution on X subject to some schema constraints, and if x_1 and x_2 are indistinguishable with respect to these constraints (they are in the same schemata), then these points have equal probabilities, i.e., $p(x_1) = p(x_2)$.*

Given a single schema constraint on X that requires the sum of the probabilities of schema Q to be a, all points in Q will have the same probability $a/|Q|$ and all points in the complement will have probability $(1-a)/(|X|-|Q|)$. Where $|Q|$ is the number of strings which match Q and $|X|$ is the total size of the search space X. The following rule extends this observation.

Theorem 1. Disjoint sets rule[1] *Given a collection of schema constraints $(\mathcal{T}, g)$ such that the schemata of $\mathcal{T}$ are pairwise disjoint, the maxent distribution $p(x)$ subject to these constraints is given by:*

[1] This follows immediately from the equal probability rule.

$$p(x) = \begin{cases} \frac{g(Q)}{|Q|} & \text{if } x \in Q \in \mathcal{T} \\ \frac{1-\sum_{Q\in\mathcal{T}} g(Q)}{|X|-\sum_{Q\in\mathcal{T}} |Q|} & \text{if } x \notin Q \text{ for all } Q \in \mathcal{T} \end{cases}$$

A **schema family** is the collection of schemata which share a common set of defined positions. We will denote a schema family by a length-ℓ string over the alphabet $\{D, *\}$ where D denotes a position with a definite bit and $*$ denotes a position with a variable bit. Thus, if the alphabets are binary, $D*D**$ denotes the four schemata $\{0*0**,\ 0*1**,\ 1*0**,\ 1*1**\}$. Note that the schemata in a schema family are pairwise disjoint and thus the disjoint sets rule applies when all of the constraints are from a schema family.

We assume that the search space Ω can be written in the form:

$$\Omega = \mathcal{A}_1 \times \cdots \times \mathcal{A}_\ell$$

where each $\mathcal{A}_i$ is the finite alphabet of legal symbols at position i. Whenever we write Ω as a Cartesian product, such as $\Omega = X \times Y$ or $\Omega = X \times Y \times Z$, we will assume implicitly that each factor is a product of some of the $\mathcal{A}_i$. In other words, each factor corresponds to a set of string positions, and these sets form a partition of the set of string positions.

If $\Omega = X \times Y$, let Π_X and Π_Y denote the projections of Ω onto X and Y respectively. If Q is a schema, then $\Pi_X(Q)$ is the schema obtained by dropping all string positions corresponding to Y. For example, if X corresponds to the first three string positions and Y corresponds to the last three string positions, then $\Pi_X(1*0*01) = 1*0$. We will say that a schema or a schema family is **variable in** Y if all positions corresponding to Y are variable. I.e. all $*$. Note that if a schema Q is variable in Y, then $Q = \Pi_X(Q) \times Y$.

Suppose that $\Omega = X \times Y$ and that $p(x, y)$ is the maxent distribution on Ω, subject to a family of constraints $(\mathcal{T}, g)$, where each $Q \in \mathcal{T}$ is variable in Y. Since $\Pi_X(Q) \times Y = Q$, g can naturally be extended to the sets $\Pi_X(Q)$ by defining $g(\Pi_X(Q)) = g(\Pi_X(Q) \times Y)$. It follows from the equal probability rule (lemma 1) that $p(x, y_1) = p(x, y_2)$ for any $x \in X$ and $y_1, y_2 \in Y$. Thus, the maxent distribution on X subject to the constraints $(\Pi_X(Q), g)$ is also the marginal distribution $p(x)$ of $p(x, y)$. We will call this distribution the maxent distribution on X subject to $\Pi_X(\mathcal{T}, g)$.

Theorem 2. Non-overlapping schemata rule. *Let $(\mathcal{T}_X \cup \mathcal{T}_Y, g)$ be a collection of schema constraints on $X \times Y$ such that each $Q \in \mathcal{T}_X$ is variable in Y and each $Q \in \mathcal{T}_Y$ is variable in X. Then the maximum entropy distribution on $X \times Y$ is the independent product of the maximum entropy distribution $p(x)$ on X subject to $\Pi_X(\mathcal{T}_X, g)$ and the maximum entropy distribution $p(y)$ on Y subject $\Pi_Y(\mathcal{T}_Y, g)$. In other words, $p(x, y) = p(x)p(y)$.*

Example: Suppose that $\ell = 6$, and X corresponds to the first 2 string positions, Y corresponds to the third string position, and Z corresponds to the last 3 string positions. Let $\mathcal{E} = DD****$, $\mathcal{F} = **D***$, $\mathcal{G} = ***DDD$. Suppose that the frequencies of all of the schemata in these families are given by the function

g. We can first apply the non-overlapping schemata rule to schema families $\mathcal{E}$ and $\mathcal{F}$ to get the maxent distribution $p(x, y)$ on $X \times Y$ subject to $\Pi_{X\times Y}(\mathcal{E}, g)$ and $\Pi_{X\times Y}(\mathcal{F}, g)$. Then we can apply the rule again to the decomposition $(X \times Y) \times Z$ to get the maxent distribution on $X \times Y \times Z$ subject to all schema family constraints. For example, $p(101101) = g(10****)g(**1***)g(***101)$.

Theorem 3. Overlapping schemata rule. *Let $(\mathcal{T}_{X,Y} \cup \mathcal{T}_{Z,Y} \cup \mathcal{T}_Y, g)$ be a collection of schema constraints on $X \times Y \times Z$ such that $\mathcal{T}_{X,Y}$ is variable in Z, $\mathcal{T}_{Z,Y}$ is variable in X, and $\mathcal{T}_Y$ is variable in $X \times Z$. Let $p(y)$ be the maxent distribution on Y subject to $\Pi_Y(\mathcal{T}_Y, g)$, $p(x, y)$ be the maxent distribution on $X \times Y$ subject to $\Pi_{X,Y}(\mathcal{T}_{X,Y}, g)$, and $p(y, z)$ be the maxent distribution on $Y \times Z$ subject to $\Pi_{Y,Z}(\mathcal{T}_{Y,Z}, g)$. Assume that $p(y)$ is the marginal distribution of $p(x, y)$ and of $p(y, z)$. Then the maxent distribution on $X \times Y \times Z$ is given by*

$$p(x, y, z) = \frac{p(x, y)p(y, z)}{p(y)} \tag{1}$$

Example: Let $\mathcal{E} = DDD***$, $\mathcal{F} = *DD***$, $\mathcal{G} = *DDD**$. Suppose g is defined on all of the schemata of $\mathcal{E}$ and $\mathcal{G}$. Since all together probabilities must sum to 1 and since every schema in $\mathcal{F}$ is a disjoint union of schemata of $\mathcal{E}$, g is also implicitly defined on $\mathcal{F}$. Similarly, g is implicitly defined on $\mathcal{F}$ from the schemata of $\mathcal{G}$. In order to apply the overlapping schemata rule, these implicit definitions must agree. The overlapping schemata rule can be used to find the maxent probability on the schema family $DDDD**$ and hence on the whole search space. For example, $p(1001**) = g(100***)g(*001**)/g(*00***)$.

Now given constraints on the schema family $\mathcal{H} = **DDDD$, we can apply the overlapping schema rule again to incorporate these constraints assuming that the constraints on $**DD**$ defined implicitly from $\mathcal{G}$ agree with those from $\mathcal{H}$. (This would be the case as long as the constraints on $\mathcal{G}$ and $\mathcal{H}$ were derived from the schema frequencies of the same previous population.) For example, $p(100101) = g(100***)g(*001**)g(**0101)/(g(*00***)g(**01**))$.

Now suppose that the additional schema family was $\mathcal{J} = D**DDD$ instead of $\mathcal{H}$. In order to apply the overlapping schemata rule, the constraints on $D**D**$ defined implicitly from $\mathcal{J}$ would have to agree with the marginals of the probability distribution on $DDDD**$ derived above from $\mathcal{E}$ and $\mathcal{G}$. There is no reason to expect this to be the case even if all constraints were derived from the schema frequencies of the same previous population. The probability distribution on $DDDD**$ satisfies a conditional independence condition, whereas there is no reason to expect that the constraints on $D**D**$ derived from $\mathcal{J}$ and the schema frequencies of the previous population would satisfy this conditional independence condition.

3 Algorithms

In this section we give two algorithms based on maxent sampling.

3.1 The Order-1 Schemata Algorithm

This algorithm follows the general framework given in section 1 with the initial collection of schemata in step 2 being all order-1 schemata. Sampling from the maxent distribution is done using the non-overlapping schemata rule (theorem 3). It is not hard to see that this is just the UMDA (uniform multivariate distribution algorithm) of [9,10].

The UMDA algorithm is a simple estimation of distribution algorithm that preserves the frequencies corresponding of each variable independently of other variables. It works well [11] for problems without significant interactions between variables, but partial solutions of order more than one are disrupted and thus the algorithm may have difficulty solving problems where the fitness function has significant interactions between variables.

3.2 The Order-2 Contiguous Schemata Algorithm

A schema family is **contiguous** if the defined positions are sequential with no intervening variable positions. As a preliminary test of the idea of applying the idea of maximum entropy to estimation of distribution algorithms, we have the following algorithm that works with contiguous order-2 schemata. Since only contiguous schemata are used, one may expect that it will work best when interactions between variables tend to be between variables that are located close to each other on the string. The order of string positions could be rearranged to achieve this property using the techniques of [12] or [2], but we did not test this approach in this paper.

The order-2 contiguous algorithm follows the framework of section 1 with the collection of schemata in step 2 being all order-2 contiguous schemata. (The schemata may overlap.) The selection can be any selection method used in a GA. We have tested binary tournament and truncation selection—see section 4.

Sampling from the maxent distribution in step 5 of the algorithm is done by repeatedly applying the overlapping schemata rule (theorem 3). To create a new individual that is to be added to the new population, we initially start with the all $*$ schema. This is progressively refined, (using the overlapping schemata rule) with more and more bits being defined, until there are no $*$ left. When all positions are fixed the corresponding individual is added to the new population.

The first two positions are fixed by choosing a schema randomly from $DD*\ldots*$ using the schema frequencies as probabilities. The third position is fixed using the overlapping schemata rule using the schema family $*DD*\ldots*$. The fourth position is chosen using the schema family $**DD*\ldots*$, etc. (This will take $\ell - 1$ steps.)

For example, let us do this for $\ell = 4$. The new individual is initially represented by the schema $****$. A schema is chosen from the family $DD**$ according to the probabilities $g(00**)$, $g(01**)$, $g(10**)$, $g(11**)$. Suppose that $01**$ is chosen. The new individual schema is now $01**$. A schema is then chosen from $*DD*$. However, this must be a compatible schema, so one of $*10*$ or $*11*$ is chosen. (Two schemata are **compatible** if they agree wherever their defined positions overlap.) These two schemata are chosen using probability $g(*10*)/g(*1**)$

for $*10*$ and probability $g(*11*)/g(*1**)$ for $*11*$. Note that these probabilities add to 1. If $*10*$ is chosen, the new individual schema is then $010*$. One of the schemata $**00$ and $**01$ is then chosen in the same way.

One might ask why we are using order-2 schemata and not higher order schemata. Note that there is a chaining effect associated with the requirement that compatible schemata must be chosen on consecutive choices. For example, if both the schema $0000**$ and $1111**$ have high frequency, then our procedure is likely to choose either a sequence of zeros or a sequence of ones. In addition, using higher-order schemata can be a topic for further research.

If the frequency of an order-2 schema is zero in the population after the selection step, then it will be zero after step 5 as well. Thus, for the algorithm as described so far, such a schema will have zero frequency from this point on. This implies that the algorithm will eventually converge to a single individual. As this behavior may be undesirable we have added a mutation procedure to the algorithm. This is done by modifying the sampling procedure of step 5 as described above. The procedure is parameterized by a mutation rate. When a schema is chosen from an order-2 schema family, with a probability equal to the mutation rate, the schema frequencies are ignored and the schema is chosen from the set of compatible schemata using equal probabilities. A schema might be chosen which results in no compatible choices when processing the next schema family. In this case, the mutation procedure is forced when processing the next schema family. Note that, if mutation is applied, we are not sampling from the exact maxent distribution.

4 Empirical Results

The order-2 contiguous schemata algorithm for a binary alphabet was implemented in Java by Pularvarty and Wright. Individuals of populations were implemented as Java BitSets and schemata were represented as a pair of BitSets. For a schema, one BitSet represented the defined positions and the second represented the definitions of those defined positions. Then set operations (as implemented by computer bitwise operations) could be used to determine if an individual was an element of a schema. For example, if an individual is represented by the set A, the defined positions of schema Q by D, and the definitions by E, then the individual is in schema Q iff $D \cap A = E$.

For the order-2 contiguous algorithm we would expect that the algorithm would do well when there are building blocks that match the algorithm. The first class of test functions used was the concatenated trap function. Concatenated trap functions were designed with the building block hypothesis in mind [13,14].

A concatenated trap function of order k is a sum of trap subfunctions of order k. A trap function t of order k is defined by

$$t_k(x) = \begin{cases} |x| & \text{if } |x| \neq 0 \\ k+1 & \text{if } |x| = 0 \end{cases}$$

where $|x|$ is the number of ones in the binary string x. We used $k = 3$ and $k = 4$. The concatenated trap function is defined by dividing the string into

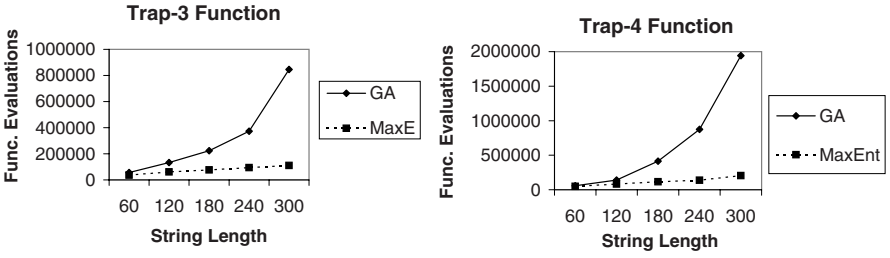

Fig. 1. On Trap benchmarks the GA scales much worse than maxent (order-2).

non-overlapping contiguous blocks of length k, and then defining an order-k trap function on each of these blocks. The value of the function is the sum of the trap functions.

Note that the optimum string is the all-zeros string, and that there are natural building blocks, namely the schemata with zeros for one block and all other positions variable. Since the building blocks are contiguous schemata, the structure of the problem matches the algorithm.

We compared the contiguous order-2 algorithm of this paper with a conventional genetic algorithm. For the experiments on the trap function and for both algorithms we used a mutation rate of 0.01, and truncation selection with a truncation parameter of 0.2. In other words, the best 1/5 of individuals are kept in the selection phase. The GA is sensitive to population size, and an attempt was made to choose a reasonably appropriate population for the GA. A population size of 5000 was used for the maximum entropy algorithm and for the GA, except when $k = 4$ and the string length was 240 or 300. In these two cases, population sizes of 10000 and 20000 were used. The genetic algorithm used one-point crossover with crossover rate 0.9. Algorithms were run until the optimum string was found. Figure 1 shows the average (over 20 runs) of number of fitness evaluations until the optimum was found. As the string length increases, the algorithm of this paper does much better than the one-point crossover GA.

The second class of fitness functions used are the NK-fitness functions [15]. These functions are also sums of simpler subfunctions. In this case, the simpler subfunctions are real-valued functions that depend on $K+1$ bits. Each subfunction is defined by a table mapping bit strings of length $K+1$ to real numbers, and the real numbers are chosen randomly using a uniform distribution from the interval $[0, 1]$. There are two versions of NK-fitness functions. For the adjacent version, the string is divided into ℓ overlapping contiguous blocks of order-$K+1$ each, and the function is a sum of random subfunctions defined on each of these blocks. For the random version, the string is divided into ℓ overlapping non-contiguous blocks of $K+1$ bits each. The ith block is guaranteed to contain bit i. The positions of the remaining bits are chosen randomly. We used $K = 3$ so that the blocks contained 4 bits each.

For a given string length and K, there are many NK fitness functions depending on the choice of the random numbers defining the subfunctions, and in the case of the random version, on the choice of the bits that make up the

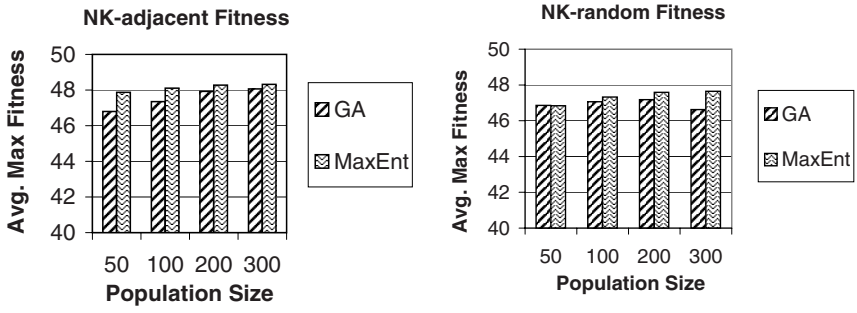

Fig. 2. Performance of GA v. maxent (order-2) for NK (N=64, K=3). Maxent is significantly better on both types of NK landscape.

block. The results given in Figure 2 are for string length 64, mutation rate 0.01, truncation selection with truncation parameter 0.2, and 5000 fitness evaluations. Each result is the average of 1000 runs.

The following procedure was used to statistically test whether the maxent algorithm was better than the GA. For each of 1000 trials, an NK fitness function was generated and then the GA and the Maxent algorithm were run with population sizes of 50, 100, 200. and 300. The fitness of the best individual over the 4 maxent runs was compared with the fitness of the best individual over the 4 GA runs. For NK-adjacent, the Maxent algorithm had better fitness 792 times, the GA had better fitness 198 times, and there was equal fitness 10 times. For NK-random, the Maxent algorithm had better fitness 646 times, the GA had better fitness 350 times, and there was equal fitness 4 times. Clearly these results are statistically significant.

5 Conclusion

A major advantage of the approach of this paper is the potential for modeling algorithms based on maximum entropy. The sampling step of the algorithm framework (step 5) depends only on the frequencies of the schemata in the schema collection (and on the outcome of random number generation). Thus, a model of the algorithm can be based on these frequencies rather than the frequencies of all strings. This means that the model is naturally coarse-grained.

This can be illustrated by existing models of algorithms based on gene pool recombination and linkage equilibrium, such as the UMDA algorithm. One analysis of the UMDA, based on quantitative genetics, appears in [11]. Fixed point analysis is used to explore the phenomenon of bistability in the UMDA with mutation in [16]. Both of these analyses use the fact that the population is in linkage equilibrium after gene pool recombination. When order-1 schema families are used, step 5 of our algorithm framework is gene pool recombination, and in this case, linkage equilibrium is equivalent to maximum entropy. In our algorithm that uses contiguous order-2 schema families, the state of the population

is reduced to the frequencies of the schemata in these families, and a model of the algorithm can be based on keeping track only of these frequencies.

In conclusion, we have presented a novel paradigm based on maximum entropy for constructing estimation of distribution algorithms. Preliminary experiments with an algorithm based on this idea using order-2 schemata on deceptive trap functions (Figure 1) and NK landscapes (Figure 2) are promising. Further work on using higher order schemata should be done.

Acknowledgements. CRS is grateful for support from DGAPA project IN100201 and Conacyt project 41221-F. AW, RP and WBL are grateful for hospitality of the Instituto de Ciencias Nucleares, where this work was initiated, and to the above grants for financial support.

References

1. Pedro Larrañaga. *A review on estimation of distribution algorithms*, chapter 3, pages 57–100. Kluwer Academic Publishers, 2002.
2. Jeremy S. de Bonet, Charles L. Isbell, Jr., and Paul Viola. MIMIC: Finding optima by estimating probability densities. In Michael C. Mozer et. al., editor, *Advances in Neural Information Processing Systems*, volume 9, page 424. MIT Press, 1997.
3. E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.
4. John Holland. *Adapdation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.
5. C. R. Stephens and H. Waelbroeck. Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124, 1999.
6. Ricardo Poli. Exact schema theory for genetic programming and variable-length genetic algorithms with one point crossover. *Genetic Programming and Evolvable Machines*, 2(2):123–163, 2001.
7. William B. Langdon and Riccardo Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.
8. C. R. Stephens, H. Waelbroeck, and R. Aguirre. Schemata as building blocks: does size matter. In *Foundations of Genetic Algorithms 5*, pages 117–133. Morgan Kaufmann, 1997.
9. Heinz Mühlenbein and Thilo Mahnig. Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1999.
10. Heinz Mühlenbein and Thilo Mahnig. FDA – a scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation*, 7(4):353–376, 1999.
11. Heinz Mühlenbein. The equation for the response to selection and its use for prediction. *Evolutionary Computation*, 5(3):303–346, 1997.
12. Robert E. Heckendorn and Alden H. Wright. Efficient linkage discovery by limited probing. In Erick Cantú Paz et al., editor, *Genetic and Evolutionary Computation – Gecco 2003*, LNCS 2724, pages 1003–10014. Springer Verlag, 2003.
13. Kalyanmoy Deb and David E. Goldberg. Analyzing deception in trap functions. In *Foundations of Genetic Algorithms - 2*, pages 93–108. Morgan Kaufmann, 1992.
14. Martin Pelikan, David E. Goldberg, and Erick Cantu-Paz. Linkage problem, distribution estimation, and bayesian networks. *Evolutionary Computation*, 8(3):311–340, 2000.

15. Stuart A. Kauffman. *The Origins of Order*. Oxford University Press, 1993.
16. A. H. Wright, J. E. Rowe, R. Poli, and C. R. Stephens. Bistability in a gene pool GA with mutation. In *Foundations of genetic algorithms-7*, San Mateo, 2003. Morgan Kaufmann.
17. Ameil Feinstein. *Foundations of Information Theory*. The Maple Press Company, York, PA, USA, 1959.
18. Silviu Guiasu and Abe Shenitzer. The principle of maximum entropy. *The Mathematical Intelligencer*, 7:42–48, 1985.

Appendix

In this section we give justification for the equal probability rule, the non-overlapping schemata rule, and the overlapping schemata rule.

Proof of the equal probability rule: Suppose that there are points $x_1, x_2 \in A$ such that $p(x_1) \neq p(x_2)$. Define the probability distribution q by $q(x_1) = q(x_2) = (p(x_1) + p(x_2))/2$ and $q(x) = p(x)$ for all $x \notin \{x_1, x_2\}$. Clearly, q satisfies all of the schema constraints. Since $-p \log p$ is a concave function of p, the entropy of q is greater than the entropy of p, contradicting the fact that p is the maximum entropy distribution. □

Following [17] can define the **conditional entropy** of a probability distribution $p(x, y)$ over $X \times Y$ by

$$S(X|Y) = \sum_Y p(y) S(X|y) = -\sum_Y \sum_X p(x, y) \log p(x|y)$$

The following is lemma 4 of chapter 2 of [17].

Lemma 2. *$S(X|Y) \leq S(X)$ with equality iff X and Y are probabilistically independent, i. e., if $p(x, y) = p(x)p(y)$.*

The following is taken from equation (5) of [18] and Lemma 3 of chapter 2 of [17].

Lemma 3. *$S(X, Y) = S(X|Y) + S(Y) \leq S(X) + S(Y)$, with equality if and only if X and Y are probabilistically independent.*

Proof of the non-overlapping schemata rule: We prove the rule for schema families $\mathcal{E}$ and $\mathcal{F}$ which correspond to the product decomposition $X \times Y$. The more general case follows by induction from this case.

Lemma 3 shows that $S(X, Y) \leq S(X) + S(Y)$ with equality iff $p(x, y) = p(x)p(y)$. It is sufficient to show that if $p(x, y)$ is defined to be $p(x)p(y)$, then $p(x, y)$ satisfies all of the schema constraints.

Let $Q \in \mathcal{T}_X$. Then

$$\begin{aligned}\sum_{(x,y)\in Q} p(x, y) &= \sum_{(x,y)\in \Pi_X^{-1}(\Pi_X(Q))} p(x)p(y) \\ &= \sum_{x\in\Pi_X(Q)} p(x) \sum_{y\in Y} p(y) \\ &= g(\Pi_X(Q)) \cdot 1 = g(Q)\end{aligned}$$

The proof for $Q \in \mathcal{T}_Y$ is similar. □

Lemma 4. $S(X,Z|Y) = S(X|Y,Z) + S(Z|Y)$

Proof.

$$\begin{aligned}
S(X|Y,Z) &= -\sum_X \sum_Y \sum_Z p(x,y,z)\log p(x|y,z) \\
&= -\sum_Y p(y) \sum_X \sum_Z p(x,z|y)\log p(x|y,z) \\
&= -\sum_Y p(y) \sum_X \sum_Z p(x,z|y)\log \frac{p(x,z|y)}{p(z|y)} \qquad \text{since } p(x|y,z) = \frac{p(x,z|y)}{p(z|y)} \\
&= -\sum_Y p(y) \sum_X \sum_Z p(x,z|y)\big(\log p(x,z|y) - \log p(z|y)\big) \\
&= -\sum_Y p(y) \sum_X \sum_Z p(x,z|y)\log p(x,z|y) + \sum_Y p(y) \sum_X \sum_Z p(x,z|y)\log p(z|y) \\
&= -\sum_Y p(y) \sum_X \sum_Z p(x,z|y)\log(p(x,z|y) + \sum_Z \sum_Y p(y,z)\log p(z|y) \\
&= S(X,Z|Y) - S(Z|Y)
\end{aligned}$$

The following lemma is a special case of lemma 6 of chapter 2 of [17].

Lemma 5. $S(X|Y,Z) \leq S(X|Y)$ *with equality iff*

$$p(x,z|y) = p(x|y)p(z|y),$$

or equivalently, if

$$p(x,z|y) = p(x)p(z)/p(y).$$

Proof of the overlapping schemata rule: In view of lemmas 4 and 5, it is sufficient to show that if $p(x,y,z)$ is defined by $p(x,y,z) = p(x,y)p(y,z)/p(y)$, then all of the constraints are satisfied.

First, let $Q \in \mathcal{T}_{X,Y}$. Recall that $Q = \Pi_{X\times Y}(Q) \times Z$. Then

$$\begin{aligned}
\sum_{(x,y,z)\in Q} \frac{p(x,y)p(y,z)}{p(y)} &= \sum_{(x,y)\in \Pi_{X\times Y}(Q)} \frac{p(x,y)}{p(y)} \sum_{z\in Z} p(z,y) \\
&= \sum_{(x,y)\in \Pi_{X\times Y}(Q)} \frac{p(x,y)}{p(y)} p(y) \quad \text{since } p(y) \text{ is the marginal of } p(y,z) \\
&= \sum_{(x,y)\in \Pi_{X\times Y}(Q)} p(x,y) = g(Q)
\end{aligned}$$

The cases when $Q \in \mathcal{T}_{Z,Y}$ and $Q \in \mathcal{T}_Y$ are similar. □