RESEARCH PAPERS

Acta Cryst. (1994). D**50**, 667–674

Using Genetic Algorithms for Solving Heavy-Atom Sites

BY GEOFFREY CHANG AND MITCHELL LEWIS

The Johnson Research Foundation, Department of Biophysics and Biochemistry, School of Medicine, University of Pennsylvania, Philadelphia, PA 19104, *USA*

(*Received* 8 *June* 1993; *accepted* 21 *January* 1994)

Abstract

A novel procedure has been developed for locating heavy-atom positions in crystals of macromolecules. This method used genetic algorithms (GA's) to search for heavy-atom sites that are consistent with an observed difference Patterson function. The procedure is straightforward to apply, space-group independent, and particularly powerful for cases involving non-crystallographic symmetry of multiple heavy atoms in the asymmetric unit. In this paper, we introduce how GA's are used for determining the heavy-atom positions and show how this method is more efficient than a sequential search.

Introduction

A crucial and often difficult step in the multiple isomorphous replacement method (MIR) is determining the heavy-atom positions needed for establishing an initial set of experimentally derived phases. The procedures most commonly used are direct inspection of a difference Patterson map or some search algorithm based upon vector superposition (Buerger, 1970). A potential solution is achieved when a set of atom positions produce a group of self and cross vectors that account for the larger non-origin peaks observed in the difference Patterson function. These signal peak heights will be proportional to the scattering power of the heavy atoms while the experimental noise will be roughly proportional to the number of these scatterers (Terwilliger, Kim & Eisenberg, 1987). The difficulty in deconvoluting the difference Patterson occurs when there are large numbers of heavy atoms in the asymmetric unit. In this case, the signal relative to the origin peak height will decrease dramatically and a solution by direct inspection will not be possible. In these instances, a more sophisticated search technique would be required. Genetic algorithms offer a new and more efficient approach for deconvoluting the Patterson function.

What are genetic algorithms?

A genetic algorithm (GA) is a powerful search-optimization procedure that is based on the principles of natural selection and genetics. The goal of the genetic algorithm is to find the maximum solutions(s) to multivariable functions spanning large search domains. GA's were first developed by John Holland and his students (Holland, 1975) to understand the processes of natural evolution. In his abstract work, he applied genetic mechanisms on binary strings of 0's and 1's which he called *chromosomes*. The process of evolution maintains that chromosomes that encode more viable characteristics survive and reproduce more often than those that are less fit. By applying selective pressure to successive generations, Holland found that chromosomes with the most favorable fitness emerged.

A schematic representation of the genetic algorithm is given in Fig. 1. As shown, the GA is composed of two unique modules: a genetic engine and an evaluation function. The role of the genetic engine is to perform gene manipulations on binary chromosomes. Although population genotypes are shaped by fitness criteria, the bit operations of the genetic engine are independent of the evaluation function. The evaluation function is responsible for decoding a chromosome into search parameters and returning a measure of performance with respect to the given problem. The evaluation function provides the link between the genetic engine, which performs the search optimization, and the problem to be solved, namely searching for heavy atoms in the asymmetric unit.

The initial GA population consists of randomly constructed chromosomes. Each chromosome is converted to a functional form, and assigned a level of fitness by the evaluation function. The fitness levels are then used by the genetic engine to calculate mating probabilities for generating a new population or *children*. A roulette wheel selection process is a common method used by GA's for randomly

choosing a parent. The number of slots (or area) allocated to each parent on the roulette wheel is directly proportional to the individual's fitness level. A parent with a relatively large fitness score will have a higher probability of mating when the roulette wheel is randomly 'spun'. The new chromosomes, or children, are created by making copies of the parental chromosome and applying genetic mechanisms, or bit operations.

The two most important gene operations performed by the genetic engine are point mutation and crossing over. The purpose of point mutation is to introduce small but consistent variation in the gene pool and refine a good chromosome to an even higher fitness. Point mutations involve the random 'flipping' of bits, that is changing $0 \Rightarrow 1$ or $1 \Rightarrow 0$, on a binary chromosome. The probability of a point mutation is controlled by a mutational-rate parameter which is inversely related to size of the chromosome and number of population. An example of point mutation is illustrated in Fig. 2. The high convergence rate of the GA is largely due to the cross-over mechanism. Crossing over interchanges segments of chromosomes to form new children with mixed parental genotypes. The mechanism introduces large gene variation from old genetic building blocks and also proliferates those genes that give rise to high fitness. The cross over allows population members to accommodate for large evolutionary change, *i.e.* escaping the local minima. Fig. 3 illustrates an example of a one-point and of a two-point cross-over event. As shown, the cross-over operator swaps a homologous region of parent 1 with a corresponding region in parent 2. The cross-over points are randomly determined and the rate is a specified parameter.

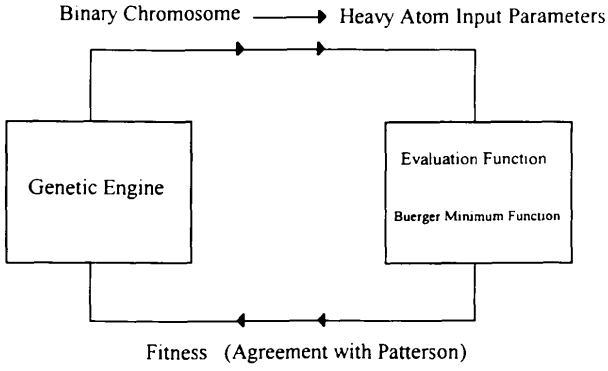


Fig. 1. Flow chart for the genetic algorithm adapted for solving heavy-atom positions. The GA generates a population of chromosomes that are decoded into heavy-atom input parameters. The Patterson vectors are then evaluated by a Buerger minimum function and the fitness score is passed back to the genetic engine. The genetic engine then uses this information to construct the next generation of chromosomes and a maximum solution will evolve reflecting a high agreement with the difference Patterson function.

Old Chromosome	Random Numbers	New Chromosome
0 1 0 1 0 0 0	.234 .444 .765 .234 **.005** .765 .676	0 1 0 1 1 0 0
1 1 0 0 1 1 0	.112 .013 .754 .345 .769 .337 .975	1 1 0 0 1 1 0
0 0 1 1 1 0 1	**.001** .766 .486 **.002** .323 .543 .324	1 0 1 0 1 0 1
0 0 1 0 0 1 1	.340 **.004** .769 .238 .987 .395 .344	0 1 1 0 0 1 1

Fig. 2. Effect of point mutation. The table shows four 7-bit chomosomes, a list of random numbers, and the resulting chromosome. When the random number is less than the mutational frequency of 0.006, as noted in bold-face type, the bit in the old chromosomes is mutated in the new chromosome.

The efficiency and effectiveness of the GA are heavily influenced by parameters such as mutation rate, cross-over rate, size of population and number of trials. Although optimal parameter values will depend on the particularities of the evaluation function, a set of general parameters has been developed by Grefenstette (1987) that are problem independent and fairly robust. These parameters were used for our heavy-atom calculations and shown to work quite well. The approximate number of trials for a 'generalized' GA to achieve convergence is the product of the population size, S, and the number of bits in the chromosome, N.

$$\text{Number of trials} = S \times N.$$

To verify that the GA has converged to a maximum solution, a number of experiments, E, are typically performed in parallel with different random seeds. This is analogous to setting up several Darwinian islands to ensure that independent populations do not fall into a local minima with respect to a particular random seed. The total number of trials is given by, therefore,

$$\text{Total number of trials} = S \times N \times E.$$

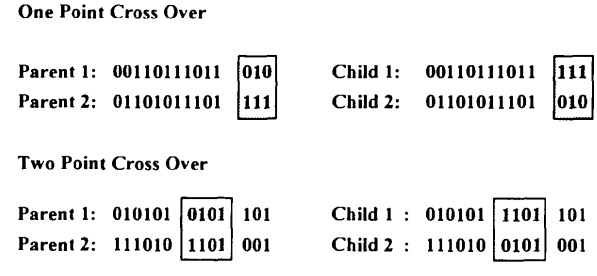


Fig. 3. Illustrates one-point and two-point cross-over events. The top part of the figure illustrates a one-point cross-over event. The children are produced by cutting the parental chromosome at the point indicated by the arrow and the parental bits in the box are exchanged. A two-point cross-over requires two cuts in the parental chromosome. The shaded area in parent 1 is given to child 2 and similarly the shaded area of parent 2 is given to child 1.

The number of chromosomes in the population is related to the chromosome size as given by Grefenstette (1987).

$$S = \exp[0.14 \times (N + 4)].$$

For a binary bit string of size N, there are 2^N possible combinations. The relative efficiency of the GA with respect to a sequential search is shown in Fig. 4. As illustrated, the GA is several magnitudes more efficient than a sequential search.

A population histogram of a typical GA search is illustrated in Fig. 5 as a function of gene distribution and generations. The maximum solution corresponding to the best agreement with the Patterson function is indicated by an arrow. The early generations of the GA are spent randomly exploring search space. During this stage, it is important that an appropriate population size be used. If the population is too small, there will be inadequate sampling of search space. The more population members used, however, the more generations (or trials) will be required for convergence. It is interesting that there are many evolutionary dead ends corresponding to local maxima. Convergence is achieved when there is homogeneity in the population distribution as illustrated by a sharp peak in the histogram function.

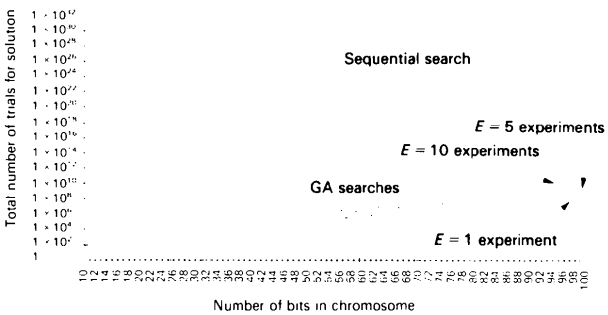


Fig. 4. Comparative semi-log plot of a sequential and GA search as a function of the number of bits in the chromosome. The estimated performance of the GA is conservative and the GA trials are shown with $E = 1$, 5 and 10 experiments. As illustrated, the genetic algorithm is significantly more efficient than a sequential search for large search space.

Methods and results

We have applied the genetic algorithm to solving the heavy-atom positions for a variety of test cases. As described above, there are two links between the genetic engine and the evaluation function: decoding the binary chromosome into useful parameters and a single number representing the fitness of that chromosome based on those evaluation parameters. The most simple and straightforward method of decoding a binary string is to directly convert from a binary number to an integer. This integer can then be scaled to a particular cell length or some parameter range. Unfortunately, this encoding scheme has several disadvantages because all binary digits flip, that is change from $1 \Rightarrow 0$, when going from $(2^N - 1)$ to 2^N where N is any integer. This phenomena is known as 'Hamming's Cliff' and can greatly inhibit the refinement of any parameter by point mutation or crossing over. A solution to this problem is to use gray-coded integers which have the property that any adjacent integer values differ by only one bit position. This makes a single point mutation more effective in parameter refinement. Shown in Fig. 6 is an example of gray-encoded integers and an example of a six-parameter decoding scheme.

A good fitness evaluation is the most difficult part of constructing any genetic algorithm. Again, the genetic engine is truly independent of the problem

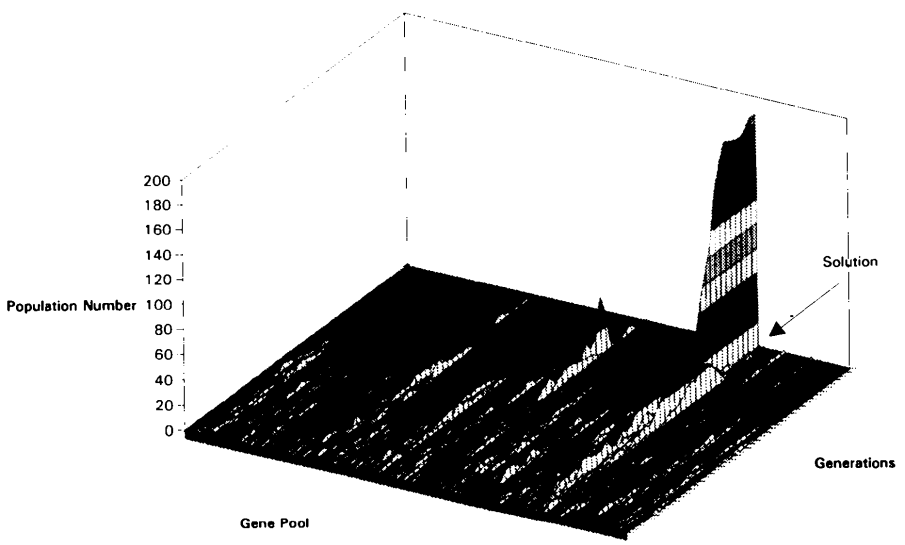


Fig. 5. Population histogram of GA evolution as a function of generations and gene pool. The maximum solution is indicated by an arrow. This illustrates the evolutionary history of a typical GA run. The early generations of the GA are spent randomly exploring space. During the intermediate stages, several local maxima are explored which result in many evolutionary dead ends. Convergence is noted by a peak in the histogram function which corresponds to a particular genotype with a maximum solution.

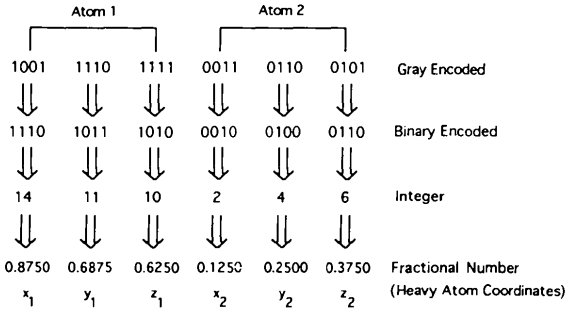
and will seek to find the maxima of any function. Clearly, if the maximum value of the evaluation function does not correspond to the correct solution, the GA search will certainly fail. Deconvoluting the Patterson function requires an evaluation procedure to assess how well the position of a heavy atom or atoms agrees with the observed difference Patterson function. A true heavy-atom position will yield self and cross vectors corresponding to strong peaks in the Patterson function. In practice, the search for a heavy-atom positions is most effectively accomplished by the vector superposition minimum function as suggested by Buerger (1970). This is the evaluation function used in our test cases. Below we describe heavy-atom searches and show that GA's are useful and highly efficient tools for deconvoluting the Patterson function.

Patterson calculation

Unfortunately, the observed diffraction data required for MIR phasing, F_{PH}, are usually not published and consequently difficult to obtain. Therefore, to determine the utility of the GA for solving heavy-atom sites, calculated diffraction data were used. In our efforts to simulate better experimental isomorphous replacement, the positions of heavy atoms were constructed by using coordinates of known water molecules from the first hydration shell and reassigning the scattering to that of an Hg atom. Structure factors were calculated with d spacing greater than 6 Å and only 80% of the diffraction data was used. Moreover, the magnitudes of the structure factors were randomly varied to simulate errors in the measurements ($R_{sym} = 8$–12%). A difference Patterson function was calculated and used in the search procedure.

Integer	Binary Coded	Gray Coded	Fractional Conversion 16:1
0	0000	0000	0.0000
1	0001	0001	0.0625
2	0010	0011	0.1250
3	0011	0010	0.1875
4	0100	0110	0.2500
5	0101	0111	0.3125
6	0110	0101	0.3750
7	0111	0100	0.4375
8	1000	1100	0.5000
9	1001	1101	0.5625
10	1010	1111	0.6250
11	1011	1110	0.6875
12	1100	1010	0.7500
13	1101	1011	0.8125
14	1110	1001	0.8750
15	1111	1000	0.9375

(*a*)



(*b*)

Fig. 6. The conversion of a string of bits to a functional form. (*a*) is a table of integers, the equivalent binary and gray-coded representations, and a fractional conversion value. The fractional conversion number is simply the integer divided by 16. Note that adjacent integers have gray-encoded numbers that differ by a one-bit position. (*b*) illustrates how a 24-bit chromosome is decoded into six fractional atomic coordinates using the gray-encoding scheme.

The single-site search

A series of calculations were performed using the GA for finding a single heavy-atom site. A 21-bit chromosome (3 parameters × 7 bits) was used to sample an interval of 1/128 of a cell edge. In cases where the Patterson space group had an invarient translation, a 14-bit chromosome was used appropriately. As shown in Fig. 4, the range of projected evaluations necessary for convergence was approximately 150–1000 and 1500–3000 for a 14- and 21-bit chromosome, respectively. The upper projected limits are very conservative. The evaluation ranges observed for the GA single heavy-atom searches are shown in Table 1 for ten single cases involving calculated and noisy data. As illustrated, the GA is 2–3 orders of magnitude faster than a sequential search.

The latter three cases in Table 1 provided an opportunity to use actual diffraction data. The crystal structure of interleukin-1α was solved by MIR and refined to 1.7 Å (Lewis, unpublished work). The protein crystallized in space group $P2_12_12_1$ with cell parameters $a = 43.3$, $b = 57.0$ and $c = 64.1$ Å. Three mercury compounds (ethyl mercury chloride, mercury potassium iodide and phenyl mercury acetate) provided single-site isomorphous derivatives. These experimental observations enabled us to explore more rigorously the benefits of the GA on real diffraction data. An effective sampling of better than 0.2 Å using a 21-bit chromosome (3 parameters × 7 bits) was used. In all three instances the GA found the single-site solution in a fraction of the total search space.

The GA single-site search can be easily modified to determine multiple heavy-atom positions in a singular fashion. The evaluation function is slightly modified to accommodate for the self and cross vectors created from previously determined heavy-atom positions. To ensure that the previous sites do not overlap with the test site, a distance check is performed. This eliminates the trivial solution of two atoms having the same coordinates. This method is repeated, or 'boot strapped', to build up N sites until

Table 1. *Single-site search results for* 11 *proteins*

The actual search space was the entire unit cell. Diffraction data corresponding to the heavy atoms are described in the text. Noise was added to the structure amplitudes to simulate experimental data more realistically. The R_{sym} for the noisy data was between 8 and 12%. The cases involving IL-1α involved real diffraction data. The last column is the range of evaluations that were necessary to obtain convergence. The range is based upon running a number of independent experiments with different random seeds. A minimum of five experiments were run for each protein. In all instances, the GA was able to find the correct solution but the number of trials varied. A sequential search at the same sampling interval requires 2^N, where N is the number of bits in the chromosome.

Protein (PDB file)*	Space group	Patterson symmetry	Perfect data iso-residual	Noisy data iso-residual	Chromosome length (Bits)	Approx. range of evaluations for convergence
1ALC	$P2_12_12$	*Pmmm*	0.213	0.231	21	1500–3000
1MBW	*P6*	*P6/m*	0.176	0.208	14	150–1000
1SGT	$C222_1$	*Cmmm*	0.216	0.246	21	1500–3000
2APR	$P2_12_12_1$	*Pmmm*	0.178	0.205	21	1500–3000
2FXB	*C2*	*C2/m*	0.250	0.271	14	150–1000
2LHM	$P2_12_12_1$	*Pmmm*	0.162	0.193	21	1500–3000
4FD1	$P4_12_12$	*P4/mmm*	0.236	0.267	21	1500–3000
4RXN	*R3*	$R\bar{3}$	0.303	0.311	14	150–1000
5CPA	$P2_1$	*P2/m*	0.112	0.155	14	150–1000
6LYZ	$P4_32_12$	*P4/mmm*	0.260	0.279	21	1500–3000
IL-1α $HgCH_3Cl$	$P2_12_12_1$	*Pmmm*	–	0.230	21	1500–3000
IL-1α HgKCl	$P2_12_12_1$	*Pmmm*	–	0.190	21	1500–3000
IL-1α Phe-Hg-$COCH_3$	$P2_12_12_1$	*Pmmm*	–	0.220	21	1500–3000

* Bernstein *et al.* (1977).

Table 2. *Comparison of different two-site search methods*

The actual search space was the entire unit cell. The isomorphous residuals for a two-site derivative are listed corresponding to the perfect and noisy diffraction data described in Fig. 1. As above, the IL-1α involved real diffraction data. The last two columns are the number of evaluations that were performed to obtain convergence. The range is based upon running a number of independent experiments with different random seeds. A minimum of five experiments were run for each protein. A sequential search at the same sampling interval requires 2^{35} or 2^{42} evaluations. The number of evaluations that were actually required to achieve convergence is a mere fraction of the total search space.

Protein (PDB file)*	Perfect data iso-residual	Noisy data iso-residual	Chromosome length (Bits)	Approx. range of evaluation for 'boot strap' convergence	Approx. range of evaluation for simultaneous two-site convergence
1ALC	0.307	0.282	42	500–5000	20000–40000
1MBW	0.242	0.253	35	300–2000	8000–20000
1SGT	0.283	0.285	42	500–5000	20000–40000
2APR	0.235	0.248	42	500–5000	20000–40000
2FXB	0.337	0.354	35	300–2000	8000–20000
2HLM	0.209	0.103	42	500–5000	20000–40000
4FD1	0.306	0.306	42	500–5000	20000–40000
4RXN	0.378	0.374	35	500–5000	20000–40000
5CPA	0.156	0.157	35	300–2000	8000–20000
6LYZ	0.344	0.343	42	500–5000	20000–40000
IL-1α $PtNH_4NO_3$	–	0.280	42	500–5000	20000–40000

* Bernstein *et al.* (1977).

weak self and cross vectors are encountered. At this point, the Patterson can be considered exhausted. The results of applying the 'boot-strap' technique for solving a second heavy-atom site are shown in Table 2. The case of the two-site platinum derivative ($PtNH_4NO_3$) of interleukin-1α involved real diffraction data. As a second example, the boot-strap approach was tested using a mercury derivative of glutathione peroxidase that had eight heavy-atom positions in the asymmetric unit (Ladenstein *et al.*, 1979). Again, all eight heavy atoms were found.

The 'boot-strap' approach is efficient because heavy atoms are found in a singular fashion. Although the systematic search is still plausible, the GA is still 1–2 orders of magnitude faster. For N heavy atoms, the increase in efficiency is to the power of N. The major disadvantage of the 'boot-strap' approach arises when there are errors in previously determined heavy-atom positions. Clearly, when initial sites are falsely chosen, all subsequential searches will naturally result in a consistent set of incorrectly placed heavy atoms. The fault for this

erroneous solution is not due to the genetic engine but rather the evaluation function. For multiple heavy-atom caes, there are a large number of peaks from which strong vectors can be constructed. The inevitable case of choosing a strong cross vector for a single-site self vector is entirely possible in the Buerger minimum function because the initial search is ignorant of difference Patterson peaks contributed by other heavy-atom positions.

Our results illustrate that GA's can be used as an alternative to the sequential procedures. The genetic algorithm simply provides an alternate method of searching the asymmetric unit that is less laborious and time consuming. As with all search routines, the ability to solve the Patterson function is directly related to the quality of the data and the occupancy of the heavy atom. When the heavy-atom derivatives are of poor quality and the information is buried in the noise it is not possible for any search procedure to unambiguously deconvolute the difference Patterson.

A simultaneous multiple-site search

The nemesis of the 'boot-strap' approach (choosing incorrect initial sites) can be greatly diminished by finding all sites simultaneously. This is accomplished by incorporating all search parameters (atomic coordinates) into the binary genome. A collision constraint ensures that the maximum number of signal peaks in the difference Patterson are explored. The multiple-site search is slower than the boot-strap approach because a larger bit string requires more population and generations to adequately sample the search space. The obvious advantage of the multiple-site search is that false heavy-atom solution sets are less likely to occur. The multiple-site search is still far more efficient than the equivalent sequential search. For example, let us consider a simultaneous three-site search (three coordinates per site) with a grid sampling of 1/128 of a cell edge. A complete sequential search will require a total of $(128^3)^3 = 9.22 \times 10^{18}$ trials. The GA chromosome of a length of 63 ($3 \times 3 \times 7$) bits can conservatively find the correct solution in under 1.5×10^6 trials (Fig. 4). This is an amazing 6.15×10^{12} times more efficient than a sequential search! The result of test cases involving multiple-site searches are shown in Table 2.

While the procedure described above is general and useful for finding multiple heavy-atom sites, the efficiency for monoclinic space groups can be greatly improved by taking advantage of Harker sections. The method begins by making a list of potential single-site (self-vector) solutions corresponding to peaks on the Harker section. In this search, the chromosome will encode: (1) integers which are used as indices pointing to potential heavy-atom positions in this single-site list and (2) floating translational components. The number of parameters will naturally depend on the number of heavy atoms being searched and also the space group. The role of the GA is to find those indices and floating translations that have good cross-vector agreement with the difference Patterson. The 'pointers to lists' method has been shown to greatly reduce the size of the chromosome and increase search speed considerably.

The 'pointers to lists' method can be easily illustrated in space group $C2$. The Harker section of this monoclinic space group provides information as to the x and z coordinates of all heavy atoms. The self vectors for any heavy-atom position will have a Patterson vector ($U = 2x$, $V = 0$, $W = 2z$). A precomputed list of potential self-vector solutions was accomplished by systematically peak picking the $V = 0$ section for all potential single-site solutions. This list is sorted by peak height and the stronger peaks are saved. For N heavy-atom positions, the genetic engine is responsible for passing N integers and $(N-1)$ floating parameters which are used to calculate the cross-vector solutions. The goal of the genetic algorithm is to find those indices which give rise to good cross-vector solutions. The 'pointers to list' method was applied to a number of two-site monoclinic Patterson functions. The number of trials required to find the correct solution ranged from 50 to 300 trials. The procedure was then repeated for three and four heavy atoms. Again, as in all cases, the GA was able to find all the heavy-atom sites.

Using non-crystallographic symmetry

A powerful feature of the GA is its ability to simultaneously refine multiple parameters. The GA is therefore ideally suited for those heavy-atom searches involving a non-crystallographic operator relating heavy-atom positions. For example, when molecules and presumably heavy atoms are related by non-crystallographic symmetry, there exists a simple transformation that allows one to compute the position of an atom at x' if one knows the position x,

$$x' = Cx + d. \tag{1}$$

In the above equation, C is a rotation matrix and d is a translation vector. When the non-crystallographic symmetry operator is accurately known the problem is trivial. In practice, however, the non-crystallographic operators C and d may only be known approximately. This will make a search difficult unless the rotation and translation operators are refined simultaneously while searching for the heavy-atom position x'. Below we provide two examples of how non-crystallographic symmetry may be incorporated in the heavy-atom search.

Glutathione peroxidase (1GP1, Ladenstein *et al.*, 1979) is a tetrameric selenoenzyme protein that crys-

tallized in space group $C2$ with a dimer in the asymmetric unit. A self-rotation function of the $\kappa = 180°$ section revealed two large peaks at $\varphi = 50\,(10)°$, $\psi = 90°$ and $\varphi = 150\,(10)°$, $\psi = 90°$ implying 222 symmetry with a twofold axis coincident with the crystallographic twofold. The absence of a pseudo-origin peak in the native Patterson map also suggested that there was a dimer in the asymmetric unit. In this crystal, the monomers in the asymmetric unit are related by a simple rotational operator which intersects the origin, *i.e.* $d = (0,0,0)$. The ambiguity in the cone angle, $\Delta\varphi = \pm 10°$, presents a difficulty in generating the exact non-crystallographic relationship relating the search atom and its non-crystallographic kin. The cone angle is, therefore, incorporated into the chromosome as an additional parameter. The search range of the $\Delta\varphi$ was confined to $[-10°, 10°]$ from the approximate angle of $\varphi = 50°$ and the sampling was 0.1°. In less than 10 000 evaluations, the genetic algorithm located the position of the two Se atoms and determined the direction of the twofold axis to be $\varphi = 48.7°$.

A second, and more complicated, example is based upon the crystal structure of the unliganded form of phosphofructokinase (2PFK, Rypniewski & Evans, 1989). Although the original protein coordinates were reported by Rypniewski & Evans in space group $C2_1$, the equivalent space group $C2$ will suffice for explanation. The non-crystallographic symmetry is considerably more complicated than that of glutathione peroxidase. The $\kappa = 180°$ section of the self-rotation function showed several peaks that were perpendicular to the unique axis (b axis) suggesting that a molecular twofold axis is parallel to the crystallographic twofold. The strongest peak normal to the crystallographic twofold axis is at $\varphi = 80\,(10)°$ and $\varphi = 170\,(10)°$ corresponds to the non-crystallographic symmetry that relates the two tetramers. Weaker peaks were observed at $\varphi = 60$, 150, 100 and 190° suggesting that dimers were related by non-crystallographic symmetry. The lack of a pseudo-origin peak in the native Patterson function further enforced the notion of a tetramer in the asymmetric unit. Fig. 7 illustrates the packing of the phosphofructokinase tetramers sitting on different crystallographic dyad axes. Each asymmetric unit contains two half-tetramers. The four heavy atoms, one from each monomer, are related by non-crystallographic symmetry and can be described in terms of the two transformations,

$$x_2 = Bx_1. \tag{2}$$

An atom at x_1 is related to an atom at x_2 where B is a rotation matrix that is calculated from spherical polar angles $\varphi_1 = 100\,(10)°$, $\psi = 90°$ and $\kappa = 180°$. The cone angle, $\Delta\varphi_1$, with a range of $[-10, 10°]$ is a GA parameter. To relate the two dimers within the asymmetric unit, a second transformation is needed that has a relative translational component, Δy,

$$x_3 = Cx_1 + d \quad \text{and} \quad x_4 = Cx_2 + d. \tag{3}$$

The matrix C is calculated from the polar angles $\varphi_2 = 80\,(10)°$, $\psi = 90°$, and $\kappa = 180°$ and the translation vector will have the form $d = (0.0, \Delta y, 0.5)$ where φ_2 and Δy are GA parameters with ranges $\Delta\varphi_2 = [-10°, 10°]$ and $\Delta y = [0.0, 0.75]$. The 1/2 in z is a centering operation. Again, if the parameters ($\Delta\varphi_1$, $\Delta\varphi_2$, Δy) are known precisely, the problem is trivial. Because of ambiguities, the GA search will require six parameters to fully specify the four heavy positions in the symmetric unit: three positions for x_1, two rotation angles ($\Delta\varphi_1$, $\Delta\varphi_2$) and one relative translation (Δy). An evalulation function was written to solve the problem by encoding the six 6-bit parameters (x_1, $\Delta\varphi_1$, $\Delta\varphi_2$, Δy) into a 36-bit chromosome. The first 18 bits encode the heavy-atom site, the next 12 bits represent the angles $\Delta\varphi_1$ and $\Delta\varphi_2$, and the final six bits encode the y component. Again we assumed that the values from the self-rotation function were sufficiently well known, so we could restrict the angular search. The GA was able to find the positions of all heavy atoms with a y translation of 0.31. This example illustrates the power of the GA and its ability to search and optimize several parameters simultaneously.

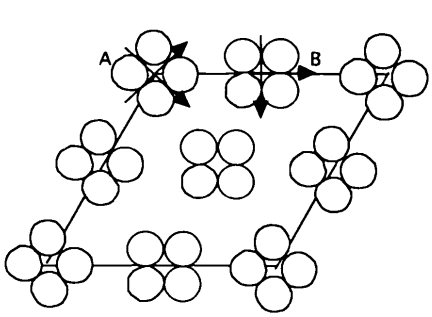


Fig. 7. Packing arrangement of molecules in the PFK cell. There are two tetramers in the unit cell that are not related by crystallographic symmetry. One tetramer is centered at the origin and the other is centered at (0.0, 0.31, 0.5). There are twofold axes that relate the monomers in each tetramer. There is also a twofold axis that relates the two tetramers.

Concluding remarks

Finding heavy atoms in difference Patterson functions can often be a difficult and laborious task. We have demonstrated that genetic algorithms are ideally suited for finding heavy-atom positions that occur as single sites, multiple sites and sites related by non-crystallographic symmetry. In all cases the GA search procedure was able to find the correct solution to the Patterson with an efficiency far greater than that of a sequential site search. The

limitations of the genetic algorithm depend only on the ability of the evaluation routine to make the correct solution the maximum solution. Terwilliger *et al.* (1987) referenced a samarium derivative in t-RNA that had vectors in the Patterson with negative values. Clearly, any vector superposition routine would fail to deconvolute this Patterson in this instance. The real advantage of the GA approach is that evaluation routines can be easily tailored to individual problems. In this paper, we have showed how GA's can be successfully applied to a variety of heavy-atom searches. The genetic algorithm is a novel procedure for finding the positions of heavy atoms in the Patterson map. It is similar in philosophy to other programs, such as *HASSP* (Terwilliger *et al.*, 1987), but is radically different in its search procedure. The utility of the GA approach is twofold: (1) the evaluation functions can be readily modified to incorporate additional information and (2) procedure is more robust than the conventional heavy-atom search technique in that many parameters can be varied simultaneously.

The genetic algorithms described in this paper may be obtained from the authors. This work was supported by grants NIH GM 44617, ARO DAAL-G-0173, and the Molecular Biophysics Training Grant 2-T32-GM082745.

References

BERNSTEIN, F. C., KOETZLE, T. F., WILLIAMS, G. J. B., MEYER, E. G. J., BRICE, M. D., RODGERS, J. R., KENNARD, O., SHIMANOUCHI, T. & TASUMI, M. (1977). *J. Mol. Biol.* **112**, 535–542.

BUERGER, M. J. (1970). *Contemporary Crystallography*. New York: McGraw-Hill.

GREFENSTETTE, J. (1987). In *Genetic Algorithms and Simulated Annealing*, edited by L. DAVIS, pp. 42–60. London: Pitman.

HOLLAND, J. H. (1975). *Adaption in Natural and Artificial Systems*. Ann Arbor: Univ. of Michigan Press.

LADENSTEIN, R., EPP, O., BARTELS, K., JONES, A., HUBER, R. & WENDEL, A. (1979). *J. Mol. Biol.* **143**, 199–218.

RYPNIEWSKI, W. R. & EVANS, P. R. (1989). *J. Mol. Biol.* **207**, 805–821.

TERWILLIGER, T. C., KIM, S.-H. & EISENBERG, D. (1987). *Acta Cryst.* A**43**, 1–5.