Protein Science (1996), *5*:1800–1815. Cambridge University Press. Printed in the USA.

Improved genetic algorithm for the protein folding problem by use of a Cartesian combination operator

ALFRED A. RABOW AND HAROLD A. SCHERAGA
Baker Laboratory of Chemistry, Cornell University, Ithaca, New York 14853-1301

(RECEIVED FEBRUARY 28, 1996; ACCEPTED JUNE 11, 1996)

Abstract

We have devised a Cartesian combination operator and coding scheme for improving the performance of genetic algorithms applied to the protein folding problem. The genetic coding consists of the C^{α} Cartesian coordinates of the protein chain. The recombination of the genes of the parents is accomplished by: (1) a rigid superposition of one parent chain on the other, to make the relation of Cartesian coordinates meaningful, then, (2) the chains of the children are formed through a linear combination of the coordinates of their parents. The children produced with this Cartesian combination operator scheme have similar topology and retain the long-range contacts of their parents. The new scheme is significantly more efficient than the standard genetic algorithm methods for locating low-energy conformations of proteins. The considerable superiority of genetic algorithms over Monte Carlo optimization methods is also demonstrated. We have also devised a new dynamic programming lattice fitting procedure for use with the Cartesian combination operator method. The procedure finds excellent fits of real-space chains to the lattice while satisfying bond-length, bond-angle, and overlap constraints.

Keywords: conformational search; dynamic programming; genetic algorithms; lattice fitting; Monte Carlo optimization; protein folding; protein structure prediction

Genetic algorithms and other evolutionary algorithms have been shown to be efficient and robust optimizers when applied to a variety of difficult optimization problems (Goldberg, 1989). The genetic algorithm optimization method was devised as an analogy to biological evolution (Holland, 1975). In the genetic algorithm scheme, an optimization problem is mapped to a computer simulation of the biological process of evolution. The cost function for the optimization process (e.g., an energy function) is identified with evolutionary fitness. The simulation consists of the biologically analogous events of mating, mutation, reproduction, and survival of the fittest. Through the evolutionary computer simulation, the fitness of the population increases. Hence, the quality of the solutions to the optimization task, which has been mapped to the biological process, also improves.

The key feature of the genetic algorithm method is its ability to use the information about the cost landscape that is contained in the genes of the various members of the population to create new low-cost candidates. This is achieved through the reproduction process, in which the genes (attributes) of the parents are combined and passed on to their children. The children undergo survival of the fittest, and then the surviving members become parents for the succeeding generation. Through this process, the fitness of the overall population and, hence, the quality of the solutions to the optimization task, is optimized. Thus, the crucial question for applying a genetic algorithm is how the attributes of the parents are passed on to the children to produce an efficient optimization method.

A number of applications of the genetic algorithm method have been made to the protein folding problem (Sun, 1993; Unger & Moult, 1993; Dandekar & Argos, 1994; Le Grand & Merz, 1994; Pedersen & Moult, 1995; Sun et al., 1995). To implement a genetic algorithm, it is necessary to encode the variables of the optimization problem into the genes. The genes of the parents are then operated on through recombination and mutation to produce the genes of the children. All of the above methods use a local direction measure as the genetic code: dihedral angles (Sun, 1993; Dandekar & Argos, 1994; Le Grand & Merz, 1994; Pedersen & Moult, 1995; Sun et al., 1995) or bond vectors (Unger & Moult, 1993). The encoding of the protein structure yields a genetic code of the form $\{(\phi 1,\psi 1), (\phi 2,\psi 2), (\phi 3,\psi 3), \ldots\}$ for the dihedral angles or $\{V^1, V^2, V^3, \ldots\}$ for bond vectors. The mode of reproduction in those methods is a crossover operator (a swapping of a subset of the genetic code of each parent, designated as Swap V) (see Fig. 1). This combination is suboptimal. What is passed on to the children through this encoding and recombination operator is local information. Specifically, the children retain the dihedral an-

Reprint requests to: Harold A. Scheraga, Baker Laboratory of Chemistry, Cornell University, Ithaca, New York 14853-1301; e-mail: has5@cornell.edu.

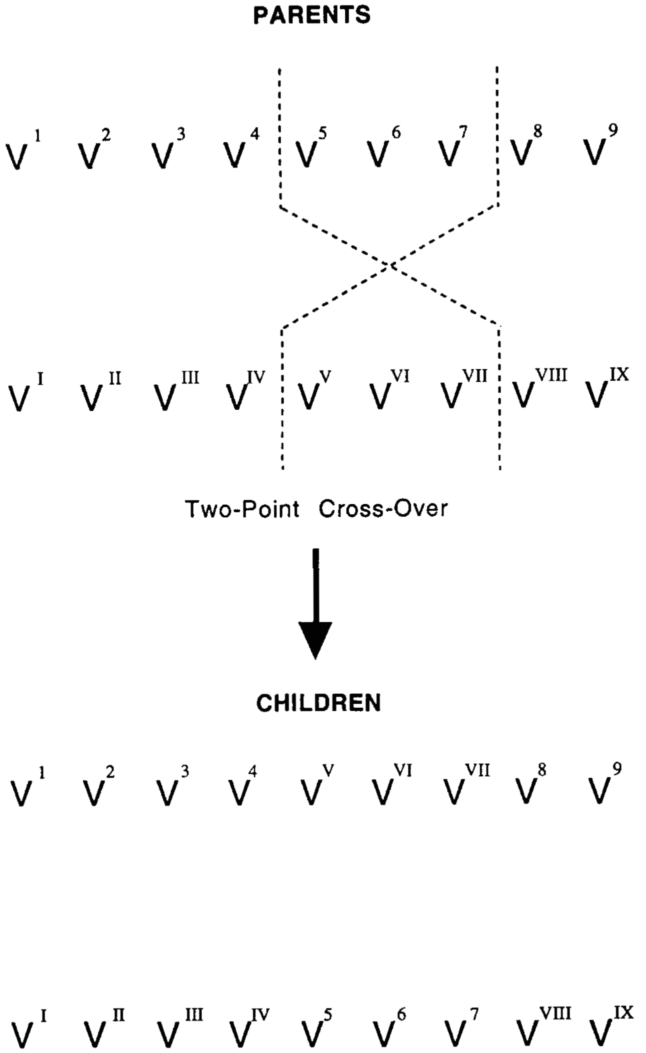
PARENTS
V1 V2 V3 V4 V5 V6 V7 V8 V9
VI VII VIII VIV VV VVI VVII VVIII VIX
Two-Point Cross-Over
CHILDREN
V1 V2 V3 V4 VV VVI VVII V8 V9
VI VII VIII VIV V5 V6 V7 VVIII VIX

Fig. 1. A schematic drawing showing the action of a crossover (swapping) operator on the genes of the parents. The two-point crossover produces the conformations of the children. The Vs represent the coordinates of the chains (bond vectors or dihedral angles). The Arabic numbers refer to parent (A) and the Roman numerals refer to parent (B). The children contain a subset of the coordinates of each parent. The crossover was carried out after coordinates 4 and 7 in the parent chains.

gles or bond vectors of their parents, and the long-range contacts that do not span a crossover point. In general, varying the local direction for a residue will have a drastic effect on the protein structure altering its topology and long-range contacts (Fig. 2). We propose a different encoding scheme (superimposed Cartesian coordinates) and a different reproduction operator (a linear combination operator), designated as a "Cartesian linear combination operator (Linco)." With this scheme, the topology and long-range contacts that both parents possess are retained in the children, although the local angles and bond lengths are not preserved. The locations where the chains of the parents differ are modified in the children by the recombination event (see Figs. 2, 3). In this way, the genetic algorithm can search the space of compact protein-like structures, and the children faithfully inherit the attributes of their parents.

Cartesian combination operator

Implementation of the Cartesian combination operator scheme involves two steps: (1) transforming the coordinates of the two parents into a mutually meaningful space and (2) producing the conformations of the children that retain and combine the attributes of the parents. Let us consider the relation between two conformations of a protein that will be serving as the parent chains. The Cartesian coordinates of these parent chains, in general, will have an arbitrary relation with respect to each other. A chain can be translated or rigidly rotated while keeping its conformation the same, but changing the values of its Cartesian coordinates, i.e., changing the location and orientation of the origin. To combine the Cartesian coordinates of the two parents, it is necessary to choose a meaningful origin, that is, a relative relation between the two

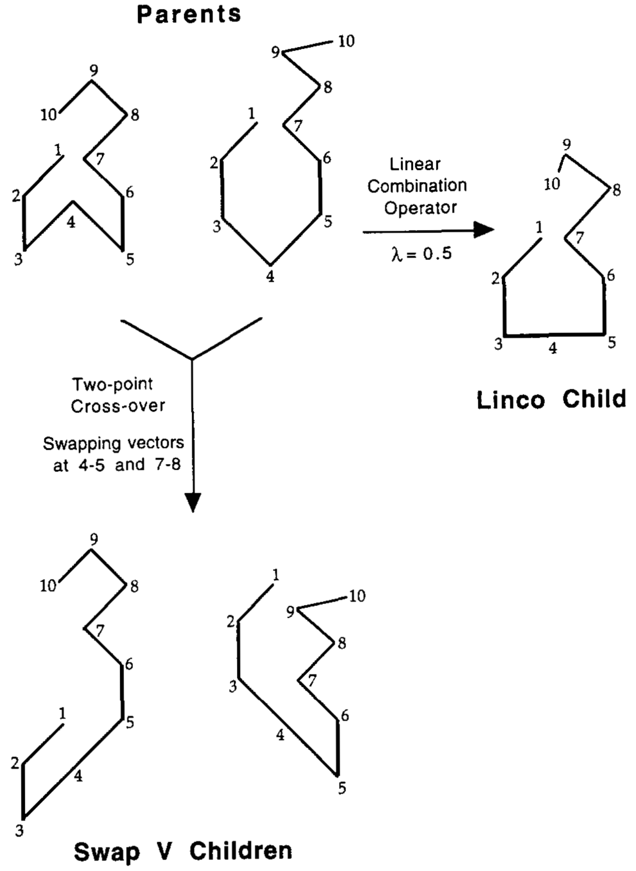

Fig. 2. Two-dimensional example of the action of two different combination operators that combine the genes (attributes) of the parents to produce the genes of their children. The children are created through the action of either the swapping crossover operator (Swap V, shown at the bottom under the down arrow) or the Cartesian linear combination operator (to the right). Even though the parents have similar topology and long-range contacts (e.g., 1 · · · 7), the children generated with the Swap V operator have very different conformations from their parents. The same behavior would be found if dihedral angles were used as the genetic encoding. In contrast, the child produced through the action of the Cartesian linear combination operator (Linco) retains some of its parents' attributes. The topology and long-range contacts that the parents have in common are passed to the child. The parts of their chains where the parents differ (e.g., at residues 4 and 10) are altered to the greatest extent. It should be noted that the Linco operator can produce distorted bond lengths in the children (e.g., the 9–10 bond in the Linco child of this figure). For details of the method, see "Cartesian combination operator" in the text.

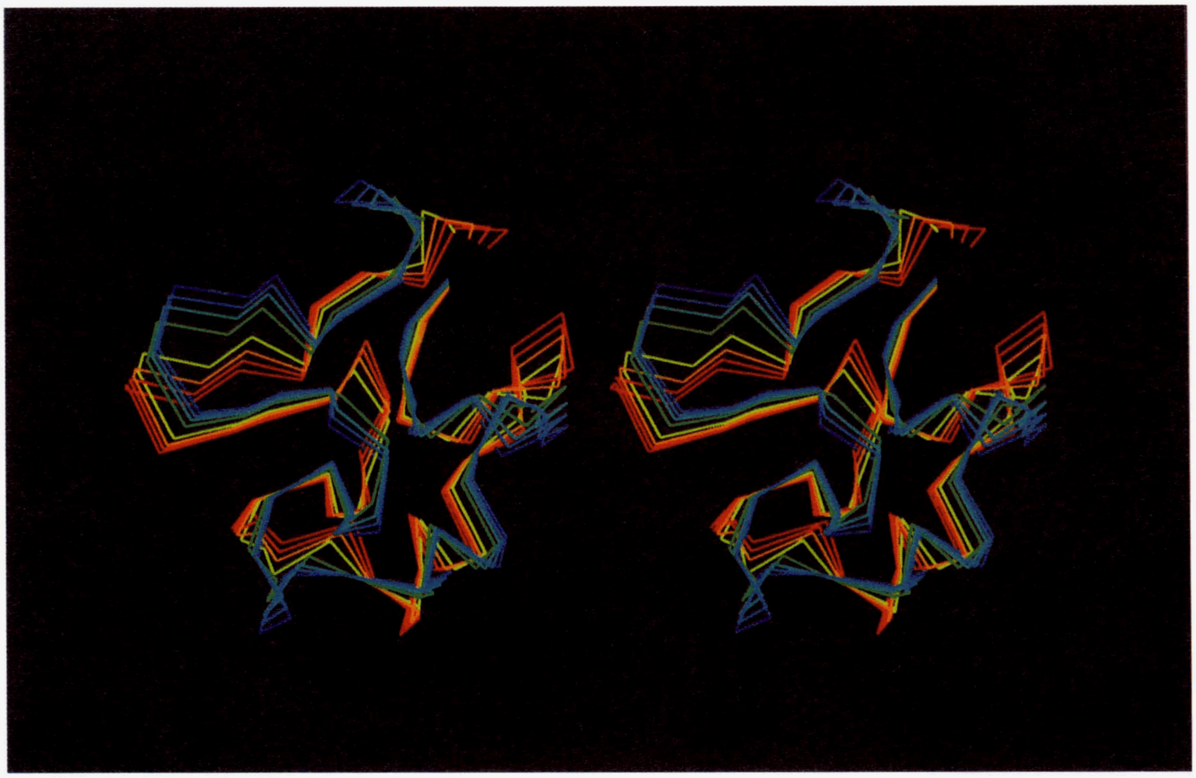

Fig. 3. Stereo diagram of two typical parents (on the outside in red and blue) and the six children (between the parents, colored with the visible light spectrum). The colors of the children change from red to blue in proportion to the λ value of the child. The coordinates of one of the parents have been optimally superimposed on the other. The children are then generated with the linear combination operator using the wide λ parameter set $\lambda \in \{0.1, 0.2, 0.35, 0.65, 0.8, 0.9\}$ as described in "Cartesian combination operator."

chains. There is some degree of freedom regarding the means of achieving this aim. The choice used in this paper is to superimpose the parent chains (A and B) rigidly to minimize the RMS deviation (RMSD) between them, where:

$$\mathrm{RMS} = \left\{ \frac{1}{N} \sum_i \left[(x_i^A - x_i^B)^2 + (y_i^A - y_i^B)^2 + (z_i^A - z_i^B)^2 \right] \right\}^{1/2}, \quad (1)$$

with x, y, and z being their Cartesian coordinates, which, in this way, are made mutually meaningful (i.e., the Cartesian coordinates of both chains now reflect the relation between them). The first step in production of the children from the parents is to superimpose the two parent chains to minimize the RMSD (Fig. 4A).

In the Cartesian combination operator scheme, the conformations of the children are then produced from a blending of the two parent chains. Again, there is much freedom in the method of achieving this aim. The simplest choice, and the one used in this paper, is the linear combination of the coordinates of the two parents, which is defined as follows. Let $\mathbf{q}^A$ be the Cartesian coordinates of parent A and $\mathbf{q}^B$ be the Cartesian coordinates of parent B with its origin and orientation found through the rigid superposition onto A. The coordinates of the child are:

$$\mathbf{q}^{Child}(\lambda, ParentA, ParentB) = \lambda(\mathbf{q}^B - \mathbf{q}^A) + \mathbf{q}^A,$$

$$\text{where } 0 \leq \lambda \leq 1 \ . \quad (2)$$

If $\lambda = 0$, parent A is recovered; if $\lambda = 1$, parent B is recovered. As λ increases from 0 to 1, the conformation of the child acquires progressively more of the attributes of parent B (see Fig. 3). In this paper, we have examined two λ sets: narrow, where $\lambda \in \{0.05, 0.10, 0.15, 0.85, 0.90, 0.95\}$; and wide, where $\lambda \in \{0.1, 0.2, 0.35, 0.65, 0.8, 0.9\}$. The narrow set examines the space close to each parent, whereas the wide set examines more of the area between the two parents. For these λ sets, each pair of parents produces six children with varying contributions from each parent. A complete description of the Cartesian combination operator genetic algorithm is shown in the flowchart of Figure 5 (also see Fig. 6A,B,C as another example, but one with a large RMSD between the parent chains).

In the current work, the space of protein conformations is a three-dimensional lattice. Because the linear combination operator will create off-lattice chains, and because it can produce distorted geometry, it is necessary to fit the children onto proper lattice chains (see Fig. 4B for an off-lattice child and Fig. 4C for the child fit to the lattice). A valid lattice chain has to occupy lattice sites and meet virtual bond length, virtual bond angle, and overlap constraints. Two alternative lattice procedures were developed to fit real-space chains to the proper lattice chains. For the complete description of the lattice fitting procedures, see "Lattice fitting by dynamic programming," below. For further details of the lattice used, see "Model system," below.

Although the method has been implemented here for the lattice space, it is readily adaptable to real space, e.g., for use with the Empirical Conformational Energy Program for Peptides (ECEPP/3) algorithm (Nemethy et al., 1992) in real space with fixed bond lengths and bond angles. The scheme to produce the children would then be as follows:

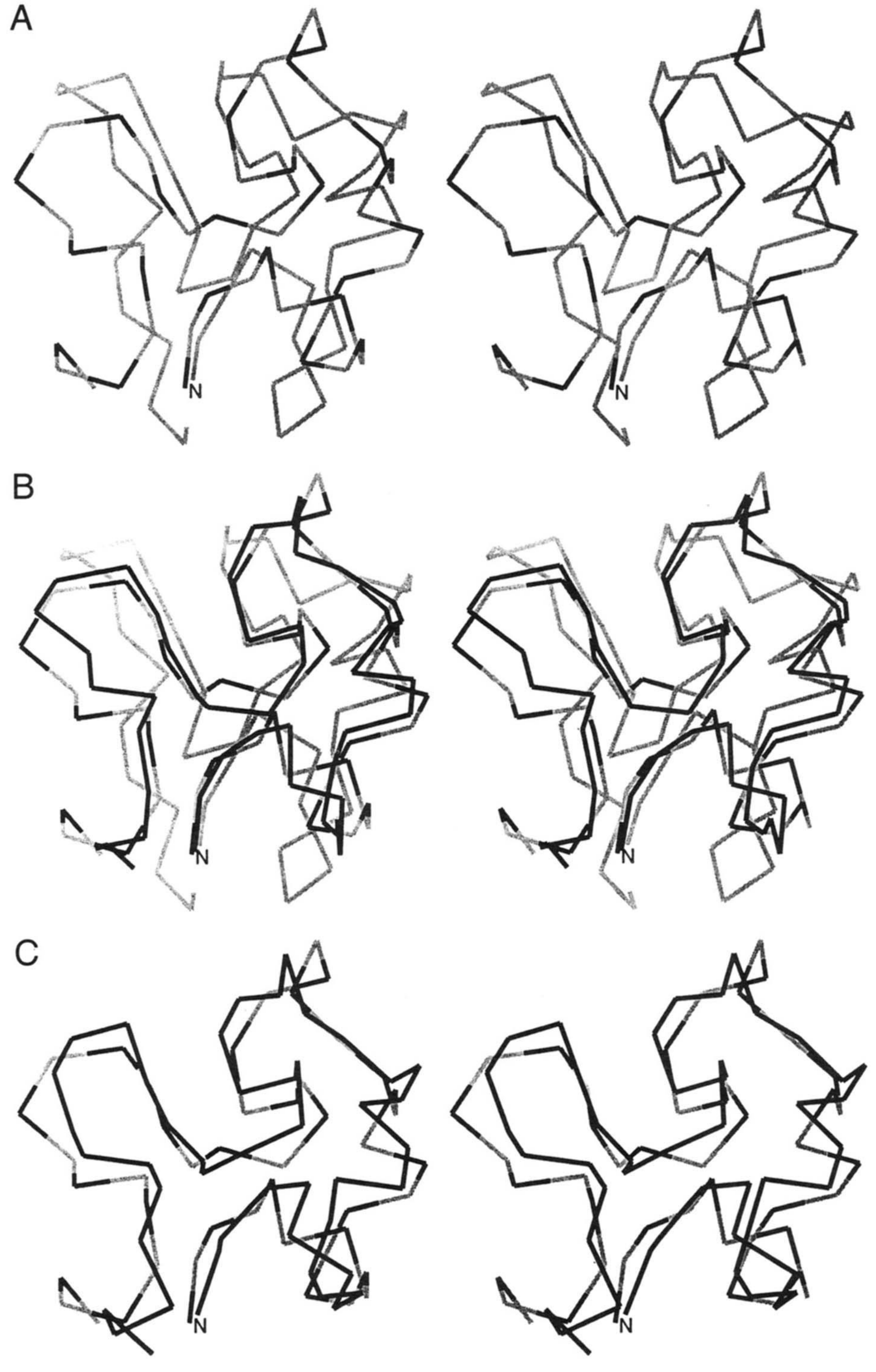

Fig. 4. Stereo diagrams of typical parent and child conformations. The amino terminus is mark with an "N." **A:** Parent A is shown as the alternating black and grey chain. Parent B is shown as the grey chain. The two parents were superimposed with a resulting RMSD of 4.7 Å. **B:** The two parents and an off-lattice child generated through the Cartesian linear combination operator with the parameter λ = 0.35. The child is shown in black and the parents are shown as in Figure 4A. The child is between its parents, proportionately closer to parent A, as seen, for example, at the C terminus. **C:** Parent A and the fitted proper lattice conformation for the λ = 0.35 child are shown. The relative conformations of the loops found in the child have been modified from parent A under the influence of parent B, as seen in the loops at the C terminus.

$$\mathbf{q}^{Child(\lambda,\, ParentA,\, ParentB)} \Leftarrow \lambda\|\mathbf{q}^{A} - \mathbf{q}^{Child}\| + (1-\lambda)\|\mathbf{q}^{B} - \mathbf{q}^{Child}\|, \qquad (3)$$

where $\mathbf{q}^A$ and $\mathbf{q}^B$ are the real-space Cartesian coordinates of the parents with B superimposed onto A, and the coordinates of the child are the set of dihedral angles ϕ, ψ, ω, and χ on the manifold of fixed bond lengths and bond angles that minimize an expanded definition of RMSD:

$$\mathrm{RMS}(\lambda) = \lambda\left\{\frac{1}{N}\sum_{i}\left[(x_i^A - x_i^{Child})^2 + (y_i^A - y_i^{Child})^2 + (z_i^A - z_i^{Child})^2\right]\right\}^{1/2} + (1-\lambda) \times \left\{\frac{1}{N}\sum_{i}\left[(x_i^B - x_i^{Child})^2 + (y_i^B - y_i^{Child})^2 + (z_i^B - z_i^{Child})^2\right]\right\}^{1/2}. \qquad (4)$$

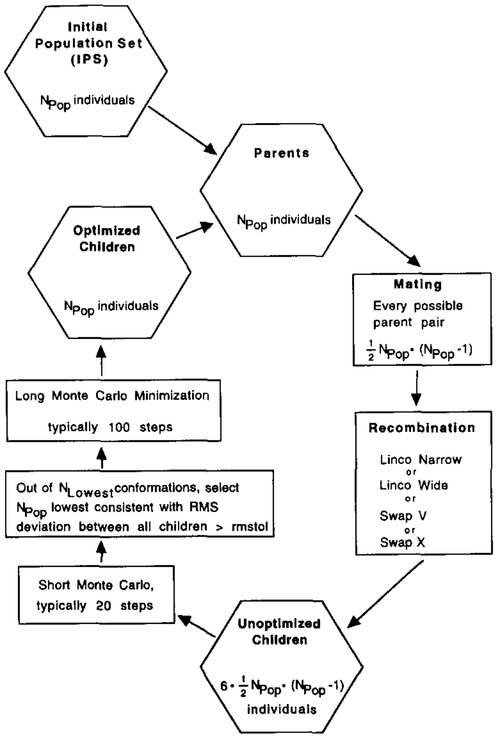
Initial Population Set (IPS)
N_{Pop} individuals
Parents
N_{Pop} individuals
Optimized Children
N_{Pop} individuals
Mating
Every possible parent pair
$\frac{1}{2} N_{Pop} \cdot (N_{Pop} - 1)$
Recombination
Linco Narrow
or
Linco Wide
or
Swap V
or
Swap X
Long Monte Carlo Minimization
typically 100 steps
Out of N_{Lowest} conformations, select N_{Pop} lowest consistent with RMS deviation between all children > rmstol
Short Monte Carlo, typically 20 steps
Unoptimized Children
$6 \cdot \frac{1}{2} N_{Pop} \cdot (N_{Pop} - 1)$ individuals

Fig. 5. Flowchart of the genetic algorithm methods. An initial population set (IPS) was created with Monte Carlo minimization of the energies of random starting conformations. An initial population set (IPS1) of ten conformations ($N_{Pop} = 10$) was generated in this manner. The energies of the conformations in this set were $\{-190, -117, -102, -99, -88, -72, -40, -39, -39, -36\}$ in arbitrary units. This set is used throughout this article unless otherwise noted. In the recombination event, $N_{Children} = 6$ children were produced for each of the $(1/2)N_{Pop}(N_{Pop} - 1)$ parent pairs. For the linear combination operator (Linco), each of the children has a different value for λ (see "Cartesian combination operator" in the text). For the bond vector swapping method (Swap V) and the Cartesian swapping method (Swap X), a two-point crossover was performed, yielding two children. This procedure was repeated three times to produce the six children per parent pair. To serve as the new parent set, N_{Pop} individuals must be chosen from the set of all children. During the generation of the children, the N_{Lowest} conformations are saved. The value of N_{Lowest} was typically 50. Of these conformations, N_{Pop} lowest-energy conformations were chosen with the proviso that their RMSD with respect to all other chosen children was greater than a tolerance (rmstol). The value for rmstol was 3.4 Å. If not enough conformations under rmstol were found, then rmstol was reduced by 67% and the process repeated. The selected children were subjected to a longer Monte Carlo minimization, resulting in the Optimized Children set. These children served as the new parents for the next generation. The overall procedure was halted if the population converged (the average RMSD among all the children was less than 0.2 Å) or if the procedure had run through a prespecified number of generations.

However, in this paper, we have investigated only the lattice as the conformational space. The children are produced by using the linear combination operator in real-space, and then fitting to proper lattice chains.

For comparison with the Linco method and, as an analogy to the dihedral or bond vector swapping method (Swap V) described in the Introduction, a scheme for the direct combination of the superimposed Cartesian coordinates was also investigated. A two-point crossover is made at random locations in the parent chains, producing two children that have a subset of the Cartesian coordinates of each parent. This method will be referred to as Swap X (for more details of the algorithm, see Fig. 5).

Model system

The system used to analyze the Cartesian combination operator method consisted of a protein chain on a lattice and a knowledge-based potential. The conformation of the protein chain was specified by its C^{α} coordinates on the lattice; side chains were not included in the computations in this paper. However, the amino acid sequence information is contained in the interaction centers located at the C^{α} coordinates of the protein chain. The lattice used here is identical to that of Kolinski and Skolnick (1994). It is a cubic lattice with 1.7-Å lattice spacing. The virtual bond vectors for the C^{α} coordinates are constrained by the virtual bond distance and virtual bond angle. The allowed bond vectors are the cyclic permutations (and negations) of the canonical bond vectors (2, 1, 1), (2, 1, 0), and (1, 1, 1), resulting in a set of 56 vectors. The virtual bond angles are restricted to the values between 78.5 degrees and 141.1 degrees. In this article, protein chains are represented only by their C^{α} coordinates. There is a hard-core repulsion between the C^{α} coordinates of non-neighbor residues; they must be separated by at least three lattice units (5.1 Å).

The knowledge-based potential consisted of four terms: a radius of gyration penalty, a disulfide-bond penalty, a contact profile term, and a pairwise contact energy term. The radius of gyration and disulfide-bond penalty terms were formulated as Gaussian functions that have a value of 0 at the conformation of the native protein and a value of 1 far from the conformation of the native protein.

The contact profile is the number of residues within a sphere around the residue that is being considered (Nishikawa & Ooi, 1986). The radii examined by Nishikawa and Ooi ranged from 8 to 18 Å. A sphere of radius 10 Å was used for this article. The term used for the energy function is the correlation coefficient between the native conformation and the conformation for which the energy is being calculated. Therefore, it ranges from 1 (complete agreement with the native conformation) to -1 (complete disagreement). It would have an average value of 0 for randomly distributed contact numbers.

The above three terms have the virtue of having their minimum value at the native conformation. At the native conformation, the penalty terms would have a value of 0 and the contact profile coefficient would equal 1. This is helpful in the analysis of optimization methods because it identifies the global minimum energy. For a more difficult optimization test of the methods, a pairwise contact energy term was also included. The contact energy term used is that of Kolinski et al. (1993). It is a pairwise energy term based on the relative contact frequencies of two amino acids found in a subset of published protein conformations. It should be noted that the simple potential described above was created to test the Cartesian combination genetic algorithm method. It was designed as a reasonable approximation for a protein folding energy function and as a representation of a typical knowledge-based potential. Ultimately, a more complete potential, especially one that includes explicit side chains, would be needed to attempt to predict the native structures of proteins. For further details of the energy function used in this article, see the Test potential appendix.

Monte Carlo minimization was performed as part of the genetic algorithm methods for local minimization. For comparison with the genetic algorithms, longer Monte Carlo simulations were also conducted as a global minimization method. Details of the Monte Carlo scheme were the same throughout. The Monte Carlo method used is identical to that of Kolinski and Skolnick (1994). The move

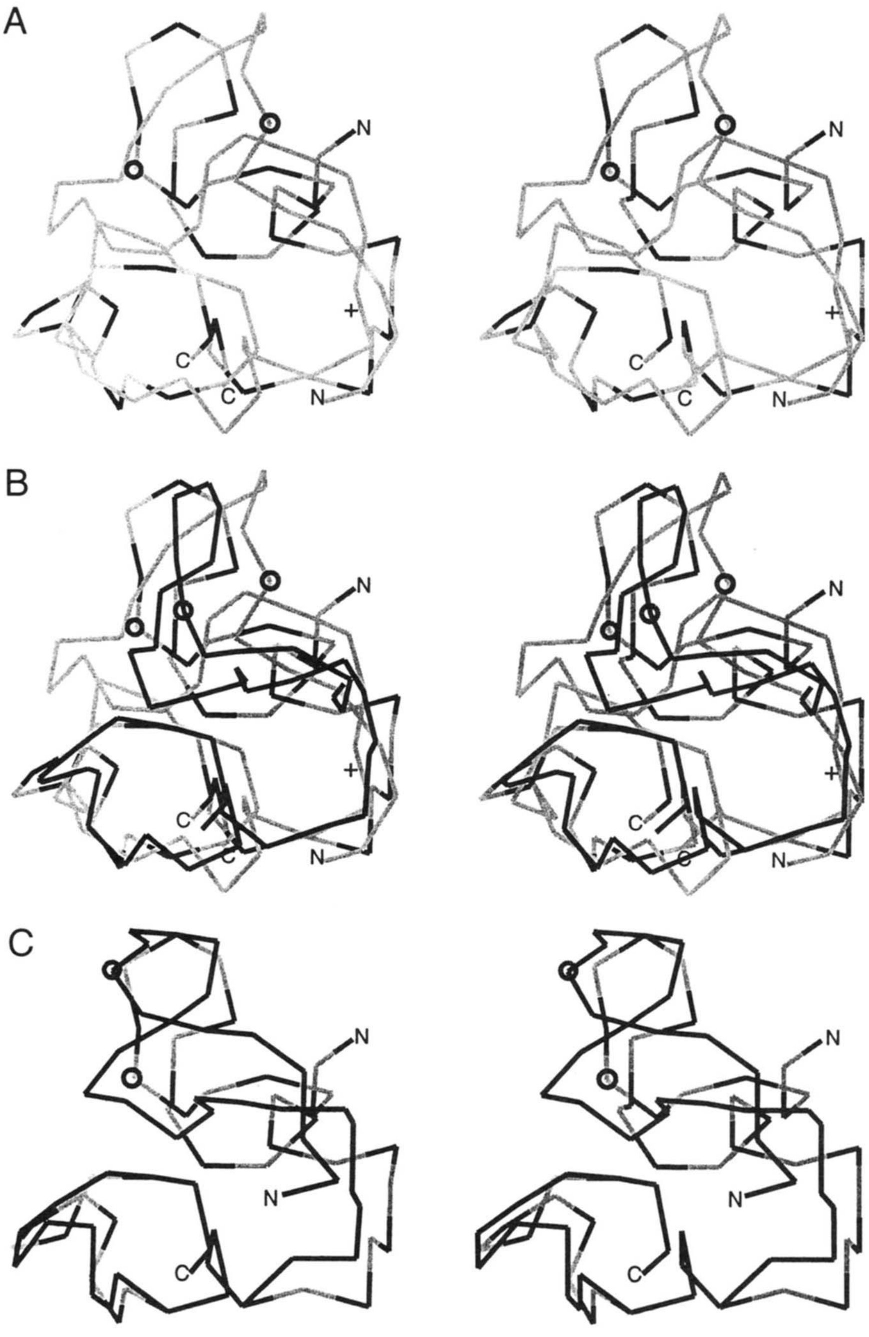

Fig. 6. A: Stereo diagrams of parent A (alternating black and grey) and parent B (grey) for two typical conformations. It should be noted the parent chains are different conformations from those of Figure 4A. The parents in this figure have very different topology from each other, with the resulting large RMSD of 8.9 Å. The amino terminus is mark with an "N" and the carboxyl end is marked with a "C." The conformations are similar at the carboxyl end, diverging at the residue marked with a + symbol. At this point, the topologies diverge. Parent A (alternating black and grey) comes forward and, at the point denoted by a circle, the chain has gone back so that the N-terminal region has crossed behind its chain. Conversely, at the point marked with the + symbol, the chain of parent B (grey) goes back and, at the circled residue, the N-terminal region passes in front of its chain. **B:** The child (black) formed in real space by the Cartesian linear combination operator method with $\lambda = 0.35$. Its conformation is in between its parents. **C:** Parent A (alternating black and grey) and the fitted proper lattice conformation for the $\lambda = 0.35$ child (black). The conformation of the child is relatively similar to parent A from the C terminus until it reaches the turn proceeding the circled residue. Under the influence of parent B, the child forms a different topology. At the circled residue, the N-terminal region of the child passes in front of its chain, whereas the N-terminal region of parent A passes behind. The linear combination operator method has made a dramatic change in the topology of the child while retaining the overall contacts and compactness of its parent.

set consisted of two-bond moves, four-bond moves, eight-bond moves, end moves, and a rigid-chain rotation.

The model protein used in all the simulations was crambin (1crn), which is an $\alpha + \beta$ protein with 46 residues and 3 disulfide bonds.

Results and discussion

Linco versus Monte Carlo

Several computer experiments were conducted to test and analyze the performance of the Cartesian combination operator genetic algorithm. The Cartesian combination operator method was compared with the standard Metropolis Monte Carlo algorithm. All implementations of the genetic algorithms proved far superior to Monte Carlo minimization. Figure 7 shows a plot of the Cartesian combination operator and of Monte Carlo minimization where both methods used the same CPU time. Of the 240 conformations generated and minimized with Monte Carlo, the lowest energy conformation found was −232, whereas the Cartesian combination operator method, after only one generation, found a conformation with an energy of −263, and eventually found conformations with energy less than −360. Furthermore, for use as starting conformations in *other* simulations, it was necessary to generate more than 500 conformations. Each conformation was minimized with 2,000 Monte Carlo steps (2.58×10^5 elemental steps). The lowest energy found by the Monte Carlo method for this additional set

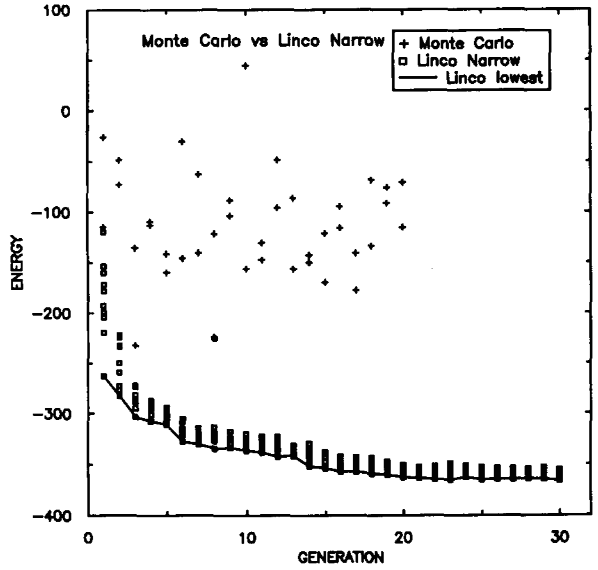

Fig. 7. Comparison of the results obtained from the Cartesian combination operator and from the standard Metropolis Monte Carlo minimization method. Each method used the same CPU time. The Monte Carlo scheme consisted of generating a starting conformation at random, then minimizing with 2,000 Monte Carlo steps (2.58×10^5 attempted elemental moves). At each generation, 12 independent conformations were Monte Carlo minimized and the two lowest found are plotted (+). The Cartesian combination method used the linear combination operator with the narrow λ set (Linco Narrow) as described in Figure 5 and had a population size (N_{Pop}) of 10. At each generation, the energy for the N_{Pop} optimized children are plotted (□) with a line connecting the lowest energy conformation.

was only −258. Any variant of the Cartesian combination operator genetic algorithm found conformations of energy below that value in only one or two generations.

Linco versus Swap V

The Cartesian combination operator was compared with the vector swapping operator for use in genetic algorithm optimization. The Cartesian combination operator scheme demonstrated a large improvement in efficiency over the traditional vector swapping method. The Cartesian combination operator scheme achieved a rapid exponential type of decrease in energy, whereas the vector swapping method shows a more gradual decline (Fig. 8A). This is illustrated further by a plot of the difference in energy between the two schemes at each generation (Fig. 8B). The same behavior of the two methods was found for all variants of the two methods and different energy potentials examined; thus, it appears to be a general property of the two methods.

Linco versus Swap X

The direct exchange of the superimposed Cartesian coordinates was also investigated. This method is analogous to the bond vector swapping method, but Cartesian variables are used as the information stored in the genes of the parents. To serve as an acceptable site for the recombination cross, a cut-off distance between the corresponding residues of the two parents was used so that the geometry of the conformation of the child would not be too distorted. A complete description of the algorithm is found in Figure 5, and the method is referred to as Swap X.

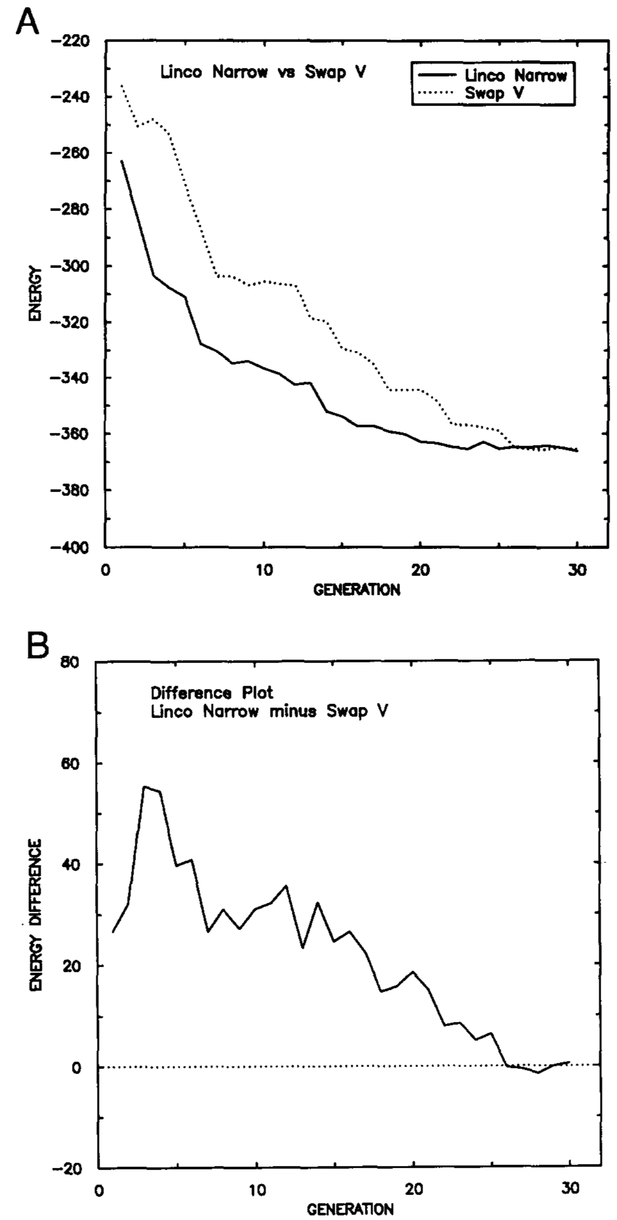

Fig. 8. A: Comparison of the results obtained from the Cartesian combination operator (Linco Narrow) and from the standard vector swapping method (Swap V). Each method used the same initial population set IPS1, which consisted of ten conformations ($N_{Pop} = 10$) with an average energy of −82.3, and −190 as the lowest-energy conformer. This data set is used throughout the computations of this article and is the initial population set used unless otherwise noted. The initial population set was generated through Monte Carlo minimization from random starting conformations. Each method (Linco Narrow and Swap V) generated six children for each parent pair, as described in Figure 5. The lowest-energy conformation for each generation is plotted. The narrow λ set was used in the Cartesian combination scheme. **B:** Comparison of the energy difference found between Cartesian combination and the vector swapping operator method at each generation. The vector swapping method (Swap V) linearly approaches the energy found with the Cartesian combination (Linco narrow) method.

The behavior of the Swap X method is a rapid decrease in energy comparable to the Linco method in the first few iterations (~7). In the subsequent generations, there is a very noisy oscillation around the −330 energy value (Fig. 9A). This behavior has

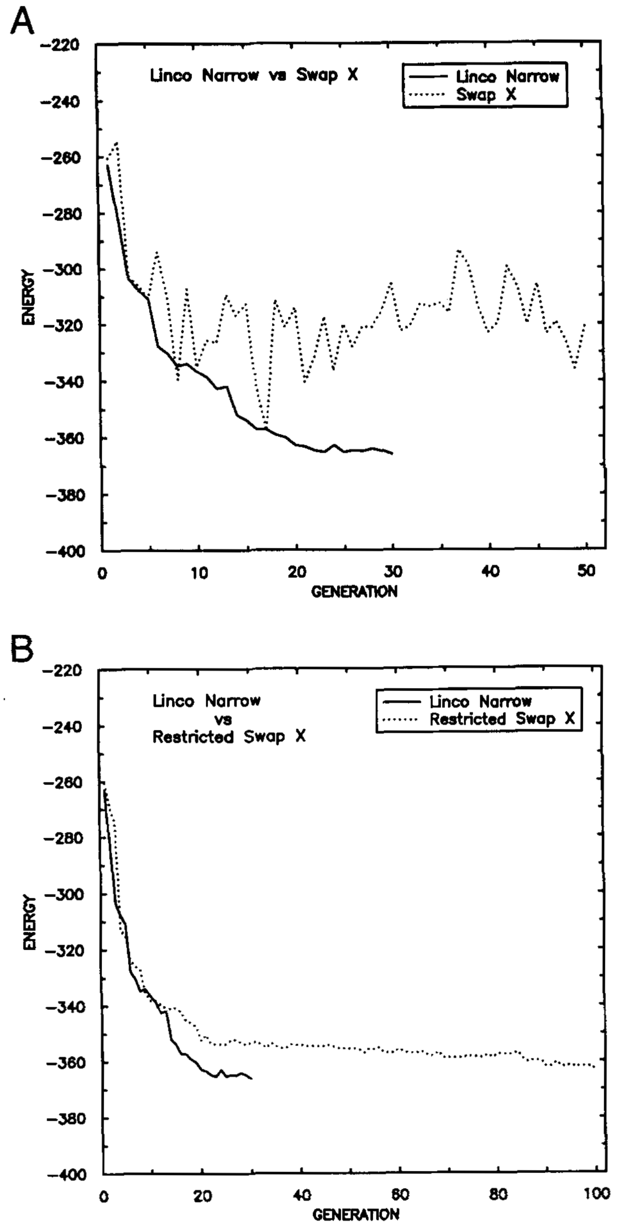

Fig. 9. **A:** Comparison of the linear combination operator (Linco) and the direct Cartesian swap operator (Swap X). The lowest energy conformation found at a given generation is plotted for both methods. In the Swap X method for each parent pair, three random two-point swaps of the superimposed Cartesian coordinates were performed. This creates six children per parent pair. The maximum allowed deviation at a splice point was 4 Å. Various cut-off distances were tested, all showing similar behavior. The Linco narrow method is as described in Figure 7. **B:** The same plot shown above except that the Swap X method now retains the lowest energy structures even if they were from the parent set. It also uses a value of $N_{Lowest} = 20$. These parameter changes serve to lessen the perturbation from the parent chains.

not been investigated fully, but is consistent with a large perturbation from the parental conformation that leads to a rapid lowering of the high-energy starting conformations, but that fails to improve good low-energy structures further. By adjustment of parameters to restrict the perturbation in the parent structure, the Swap X method can be made to reach the lower energy levels (~ −360) (Fig. 9B). This restricted search is slow, however, perhaps because it now more resembles a local search procedure than that of a genetic algorithm.

Diversity experiments

An attempt was made to characterize and investigate the behavior of the Cartesian combination method. The efficiency of the method with varying λ parameter sets was investigated. The two sets examined were: narrow, where $\lambda \in \{0.05, 0.10, 0.15, 0.85, 0.90, 0.95\}$; and wide, where $\lambda \in \{0.10, 0.20, 0.35, 0.65, 0.80, 0.90\}$. Overall, the behaviors of the two sets were similar (Fig. 10A). In the initial generations, the narrow set decreases the energy more rapidly than the wide set. After 20 iterations, the wide set has found conformations with lower energy (Fig. 10B). This behavior can be probed by switching from the narrow parameter set to the wide set at generation 15 and vice versa in the runs that produced the aforementioned results. The results show an interesting dichotomy. If an evolution started with the narrow set and is switched to the wide set, no more improvement in the energy is found (Fig. 11). Conversely, if the evolution is started with the wide set, and then is switched to the narrow λ set, a marked improvement is seen (Fig. 11). Between generation 16 and generation 21, the energy decreases 22 units to −372. This is lower than the run with Linco Wide used throughout, although it is certainly within the variability between runs with different initial populations or random seeds (see following sections). It seems that the narrow set can carry out the optimization task more efficiently, but loses genetic diversity, whereas the wide set still retains enough diversity to improve the conformations in subsequent generations. For genetic algorithms, the term "diversity" refers to the differences contained in the genetic code of the members of the population. In the specific protein folding context, it relates to the conformational space covered by the protein chains and will be measured by the average RMSD among all the parent chains.

The difference between the two λ sets can also be seen by examining how the average RMSD within the population changes as the run proceeds (Fig. 10C). The narrow set shows a rapid decrease in RMSD in the first few generations and then a slow decrease to a value of ~2 Å. The wide set shows a much more gradual decrease until it levels off at a value of ~3.5 Å. At this stage of evolution, the wide set still retains enough diversity in the population pool to improve the conformations further, lowering their energy by 5.4 units from generation 23 to the end of the evolution run.

The effect of population diversity on the ability of the genetic algorithm to improve the fitness of the population was investigated further. A scheme has been developed to reintroduce diversity into the population as the diversity is lost through the evolution. The method consists of replacing a fraction of the population with independent low-energy conformations if the average RMSD within the population decreases below a threshold. It should be noted that, in general genetic algorithm methods, some diversity is maintained by mutation operators, which randomly introduce genetic diversity. In this study, a mutation operator was not included to make the examination of the Cartesian combination operator method more understandable, i.e., foregoing the introduction of another random noise factor to the method. In a full-blown Cartesian combination operator genetic algorithm, inclusion of an appropriate mutation operator might be worthwhile.

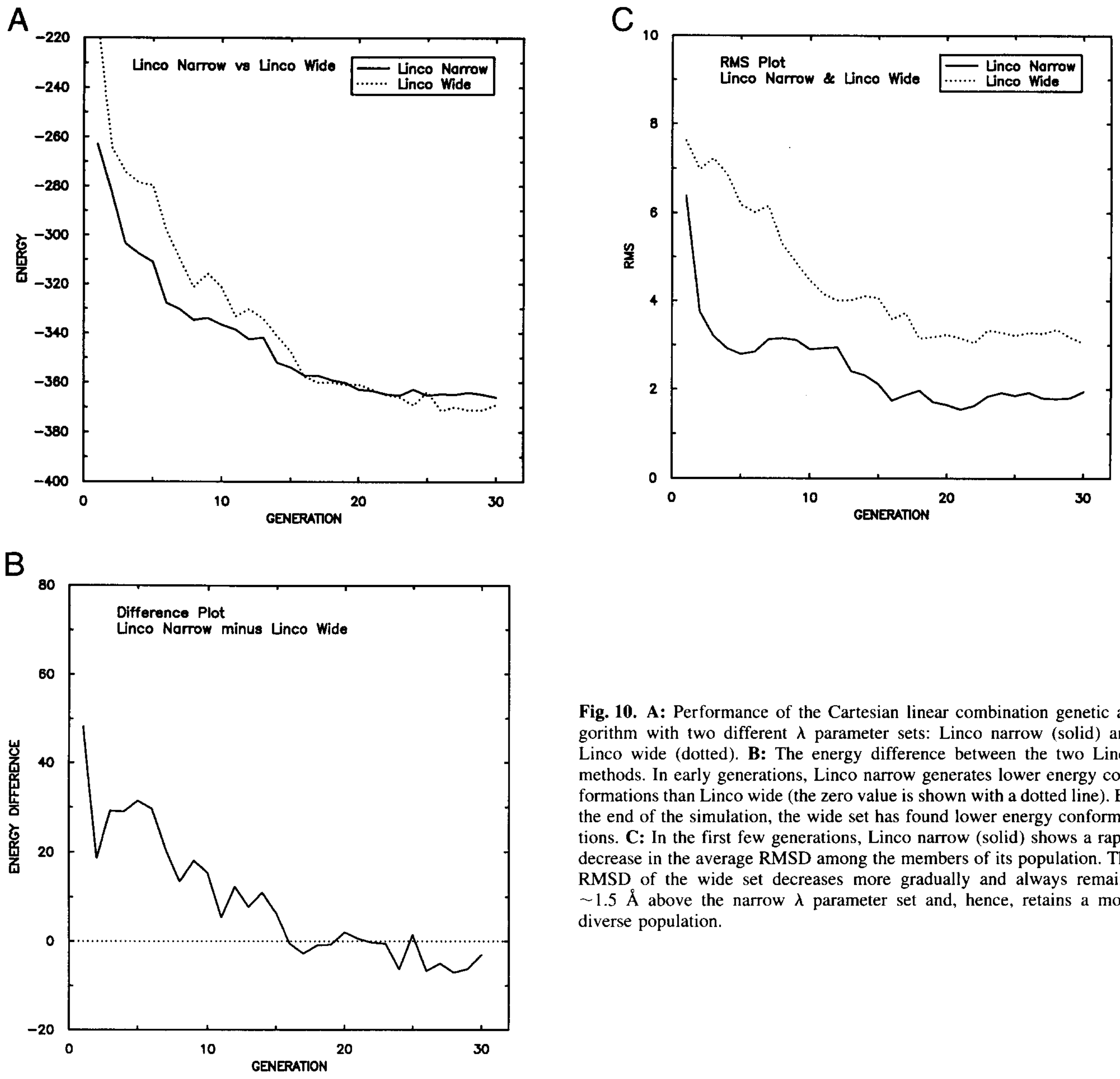

Fig. 10. **A:** Performance of the Cartesian linear combination genetic algorithm with two different λ parameter sets: Linco narrow (solid) and Linco wide (dotted). **B:** The energy difference between the two Linco methods. In early generations, Linco narrow generates lower energy conformations than Linco wide (the zero value is shown with a dotted line). By the end of the simulation, the wide set has found lower energy conformations. **C:** In the first few generations, Linco narrow (solid) shows a rapid decrease in the average RMSD among the members of its population. The RMSD of the wide set decreases more gradually and always remains ~1.5 Å above the narrow λ parameter set and, hence, retains a more diverse population.

The re-introduction of diversity was observed to improve the solutions to a small extent, consistent with what was observed with the λ parameter above. In the re-introduction scheme, half of the population was replaced with independent low-energy conformations (~ −75) if the average RMSD among the set of optimized children was below 2.55 Å. The effect was small, but the introduction of diversity improved the lowest energy conformer found by ~ 3 units to −373.8 (Fig. 12A). The plot of the average RMSD shows that, although half of the population was replaced repeatably, the genes of the higher energy-independent conformations (of energy ~ −75) were quickly overwhelmed by the low-energy conformers (Fig. 12B). Hence, the RMSD of the population (after recombining and selecting for fitness) remained close to the value before the introduction of the random independent conformations at ~2.5 Å.

In the scheme described in Figure 5, each new generation replaced the old population. No parents were retained even if they possessed very low energy. The effect of retaining the N_{Pop} lowest energy conformers regardless of whether they were children or parents was investigated. The results show a somewhat faster decrease in energy, but a premature convergence to higher energy clusters (Fig. 13A,B show the typical behavior).

We also investigated the effect of the population size on the quality of optimization. The experiments were repeated with $N_{Pop} \in \{5, 10, 15, 20, 25, 30\}$. A set of 30 independent conformations was generated from many Monte Carlo runs with energies less than −100 with an average of ~ −150 units. The results show the expected behavior: an increase in the population size allows more of the space to be searched and, hence, a more rapid decrease in the energy (Fig. 14A).

In this case, each of the initial populations was a subset of the larger group. This couples the effect of a particular set of initial conformations to that of population size. To probe the effect of different sets of initial populations, three new initial population

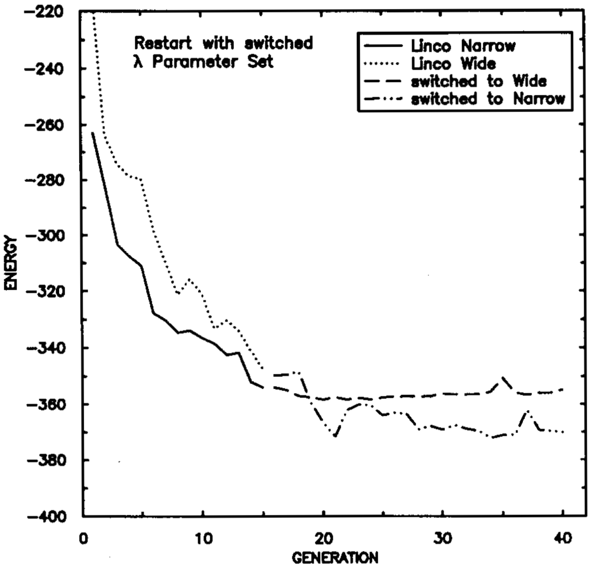

Fig. 11. Evolutionary runs for Linco narrow and Linco wide were identical to that of Figure 10 for generations 1–15. Starting from the next generation (16) until the end of the runs, the populations were evolved using the other λ parameter set.

sets were constructed and the above experiments repeated. Two new initial populations, which include an especially low-energy conformation, were constructed for N_{Pop} = 5. These are designated as 5L1 and 5L2, respectively. The first (5L1) was a subset of the population (for the 30) described above, and it contained an initial conformer with energy −246. The other new set (5L2) was the same as the N_{Pop} = 5 set above, except for replacing the highest-energy conformer (−100) with the lowest-energy conformer found through Monte Carlo with a value of −258. The third new initial population set was made for N_{Pop} = 10 (10H). The 10 highest conformers were chosen from the set of 30 initial conformations described above (their energies ranged from −124 to −100). These experiments show that there are variations caused by the initial population; however, a lower starting energy does not dictate a lower final energy (Fig. 14B). One of the low-energy initial populations led to a low-energy solution (5L2), whereas the other low-energy set (5L1) did not. Even though the initial population of 10H was a high-energy set, it behaved in much the same manner as the other N_{Pop} = 10 set.

One final attempt was made to elucidate the behavior of the Cartesian combination genetic algorithm method and the characteristics of the energy landscape. Five runs with independent low-energy (~ −150) initial populations sets were performed (Fig. 15A). The results obtained for all the sets were similar to each other, as were the N_{Pop} = 10 sets described above. For each set, the lowest-energy conformation found at generation 40 was pooled for use as the starting population for an N_{Pop} = 5 run. The energy increases initially, then slowly decreases back to its level for the initial population after 20 generations (Fig. 15A,B). At generations ~44 and ~123, the lowest-energy conformation found jumps rapidly to a lower energy level (Fig. 15B). The behavior of the method after the first ~30 generations is reminiscent of local minimization. The RMSD never converges, but remains around 4 Å, and the energy jumps from a plateau to a new plateau at a lower energy. Through this phase, the energy is reduced by 10 units to the presumed global minimum level of −386.5 units.

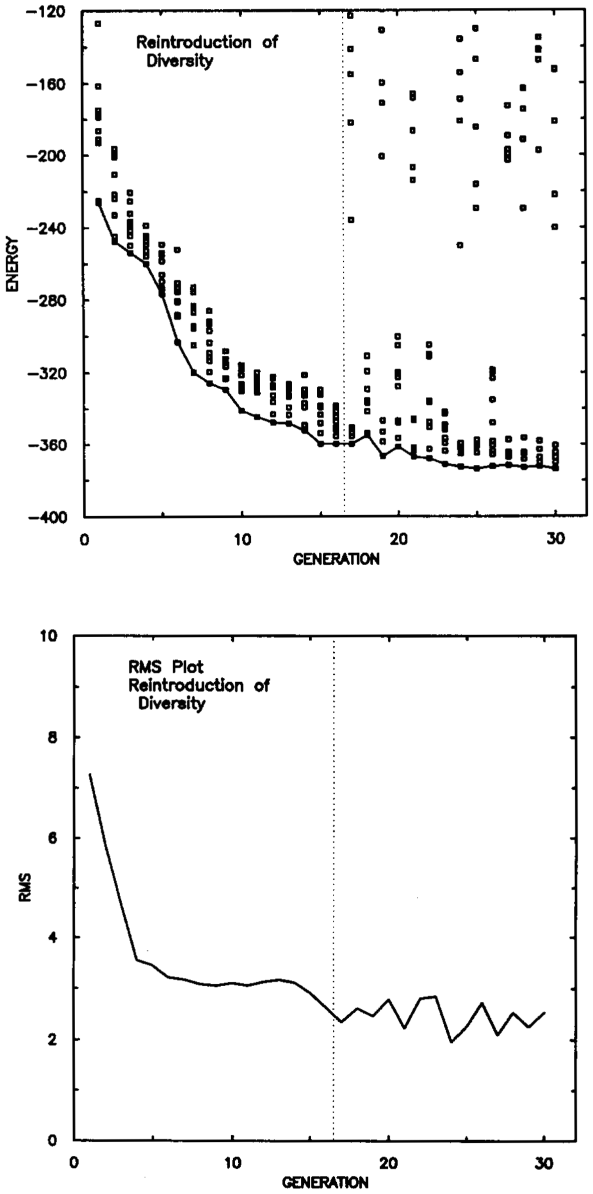

Fig. 12. **A:** Plot based on the re-introduction of diversity. The simulation was as outlined in the flow chart of Figure 5, except that it contained an additional procedure to reintroduce diversity if the average RMSD of the optimized children set fell below a set threshold parameter. In the evolution shown in this figure, this occurred after the 16th generation (vertical dotted line). At this point, half the population was replaced with the (1/2)N_{Pop} lowest-energy conformations from a set of 3 × N_{Pop} random conformations that were minimized with 2,000 steps of Monte Carlo minimization (energies average ~ −75). This replacement occurred whenever the RMSD was lower than the threshold throughout the rest of the evolution. All of the N_{Pop} conformations are plotted at each generation (□). A solid line connects the lowest-energy conformation at each generation. The narrow λ set was used as the Linco operator. **B:** The RMSD oscillates as half the population is replaced with the independent Monte Carlo generated conformations, and the RMSD repeatedly decreases below the threshold.

The above diversity results help to characterize the requirements of the Cartesian combination operator method. We have found that the optimal parameters consist of the wide λ set, a population size of N_{Pop} ~ 20, and the replacement of the par-

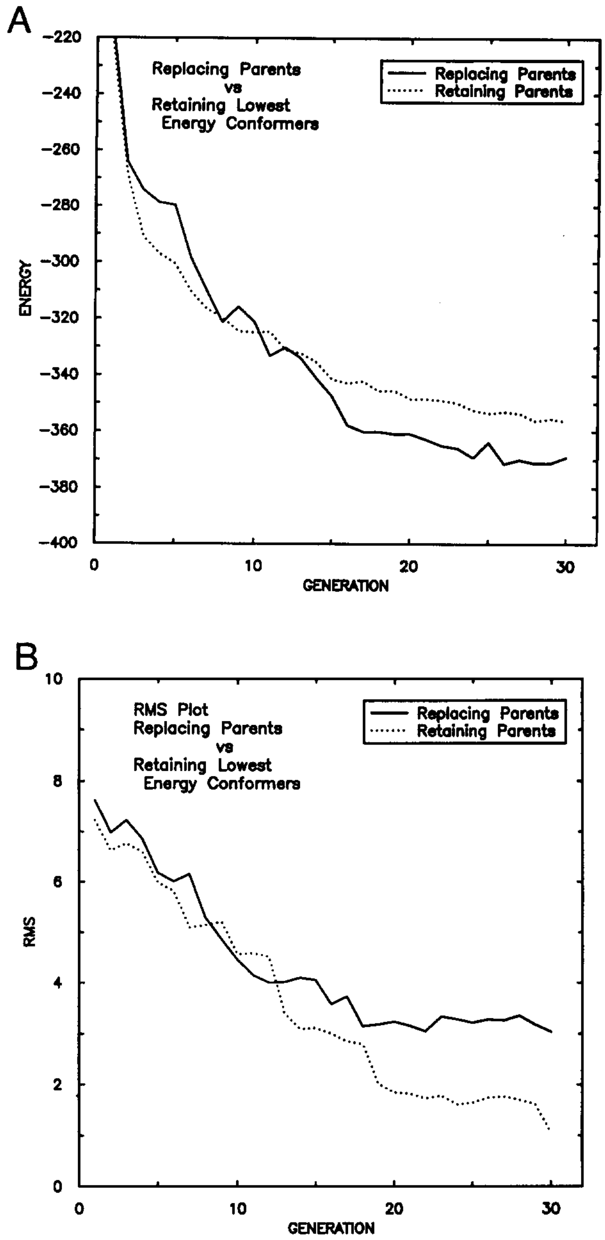

Fig. 13. **A:** Scheme that retains the lowest-energy conformations, regardless of their status as parents or children, decreased in energy more rapidly in early evolution, but converges prematurely to a higher energy level. The Linco wide method was used for both schemes. **B:** Premature convergence when retaining the lowest-energy conformers (dotted line) can be seen by examination of the average RMSD among the population. After generation 13, the Replacing Parents scheme converges rapidly to a single structure at generation 30 (~1.0 Å).

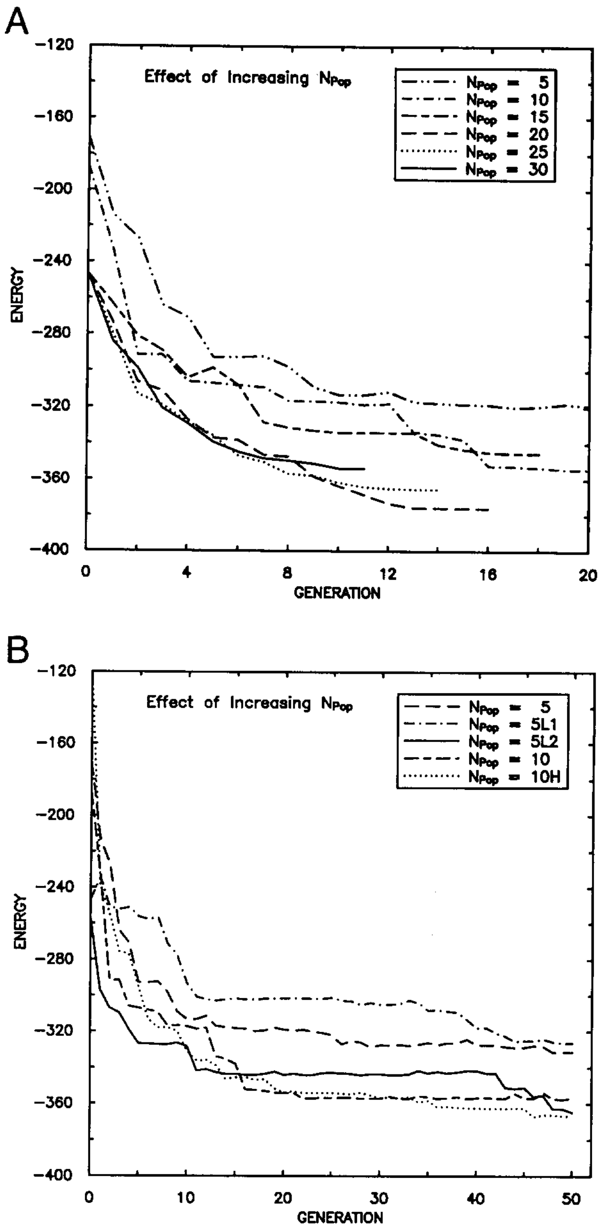

Fig. 14. **A:** Plot showing the effect of increasing population size (N_{Pop}). The Linco wide operator was used. The lattice fitting procedure was the hybrid *i*th vector/dynamic programming method (see "Lattice fitting by dynamic programming"). **B:** Plot using the identical scheme but with inclusion of different initial population sets. Sets 5L1 and 5L2 are sets that contain a very low-energy initial conformer. The 10H initial population was a high energy set made from the highest initial energy conformers in the N_{Pop} = 30 set for A. The energies are {−124, −123, −112, −111, −105, −104, −103, −101, −100} in arbitrary units.

ents at each generation. These parameters will be useful in designing a full-blown genetic algorithm, which may include all the recombination types (Linco, Swap V, and Swap X), a mutation operator, and other genetic algorithm procedures. This optimized genetic algorithm would then be used to locate low-energy conformations with a more complete knowledge-based potential and, hence, would be used to predict the native structure of proteins.

Lattice fitting by dynamic programming

The fitting of a chain in real space to a discrete set of points is an important topic in polymer science in general; for example, fitting an X-ray structure to a lattice. Here, as part of the Cartesian combination operator genetic algorithms, it is necessary to convert the children produced by recombination to proper lattice chains. A proper lattice chain must meet three requirements: it must contain

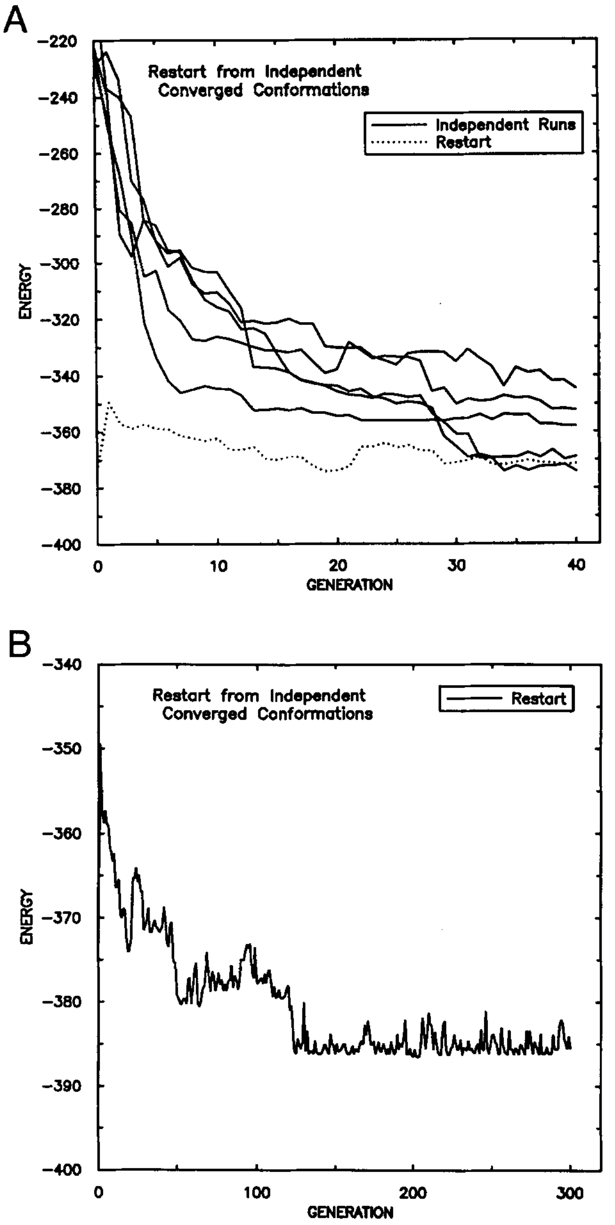

Fig. 15. **A:** Five independent initial population sets were generated with Monte Carlo minimization. Many Monte Carlo runs were conducted to generate conformations with energies less than -100. The five populations were evolved by the Linco wide operator, with a population size $N_{Pop} = 10$, and used the hybrid *i*th vector/dynamic programming lattice fitting procedure (see text). All the sets showed similar behavior. The lowest-energy conformer found at generation 30 for each run was pooled to form an $N_{Pop} = 5$ initial population with energies of $\{-374, -369, -358, -352, -345\}$ in arbitrary units. Starting with this initial population, the first 40 generations of its evolution under the same Cartesian combination method are shown (dotted line). **B:** Full evolutionary run of the very low energy $N_{Pop} = 5$ initial set shown in A. The energy jumps up to -350 after the first generation, and then decreases back to -374 at generation 20. The energy then decreases in two steep jumps at generations ~44 and ~123, finally reaching the presumed global minimum level of -386.5 units. It should be noted that the scale of the plot is greatly expended over plot A, making the changes appear magnified.

proper virtual bond lengths and virtual bond angles, and it must not contain overlaps. Specifically, it should meet the following constraints.

Let $\mathbf{Q}_i$ be the coordinates of the *i*th residue of the lattice chain.

Virtual bond constraint:

$$2.94\ \text{Å} \leq \|Q_i - Q_{i+1}\| \leq 4.16\ \text{Å} \quad \text{for } i = 1, n - 1. \tag{5}$$

Let $\gamma(Q_{i-1}, Q_i, Q_{i+1})$ be the virtual bond angle.

Virtual bond angle constraint:

$$78.5^\circ \leq \gamma \leq 141.1^\circ \quad \text{for } i = 2, n - 1. \tag{6}$$

Overlap constraint:

$$\|Q_i - Q_j\| \geq 5.1\ \text{Å} \quad \text{for } |i - j| > 1 \quad i, j = 1, n. \tag{7}$$

The specific values used for the constraint terms above are taken from the paper of Kolinski and Skolnick (1994).

The requirement for the lattice fitting procedure is that it be fast and accurate. It needs to be fast because every child chain that is generated needs to be fitted to the lattice. It must be accurate so that the attributes of the parents are transmitted faithfully to the children. It should be noted that the lattice fitting needed for the Cartesian combination method is a more demanding task than fitting the X-ray structure of a protein to the lattice. The children produced by the linear Cartesian combination operator may have very distorted geometry; the chains may have distorted virtual bond lengths and virtual bond angles, and may contain many overlaps.

For clarity throughout the remainder of this section, we will use the terminology "*i*th city" for the real-space C^α coordinates for residue *i*, and "satellite" for the lattice site chosen to correspond to that city. The term "*i*th stage" refers to a consideration of which lattice site will be chosen for the *i*th city.

A few algorithms for fitting real chains to the lattice have been published (Godzik et al., 1993; Rykunov et al., 1995). In the work of Godzik et al., three methods were used. The first was the simple procedure: start at one end of the chain and choose the proper lattice site that minimizes the local deviation between the lattice-fit chain and the real-space chain, and then proceed in this fashion to the other end. At each stage, one satellite is chosen and then is fixed for the succeeding stages. In the Cartesian combination operator method, we have used this procedure, and it will be referred to as the "*i*th vector" method. It is very fast, but can yield large deviations from the real chain. Hence, it is useful, in the Cartesian combination operator method, to follow this by a more accurate fitting procedure described below. The other two methods in the paper of Godzik et al. use Monte Carlo simulated annealing to improve an initial lattice conformation. These two methods generate a random initial conformation, then use simulated annealing to bring the lattice chain closer to the real chain according to either the interresidue distances or RMSD criteria. These methods are too slow for use with the Cartesian combination method. For example, for $N_{Pop} = 10$ and $N_{Child} = 6$, for 30 generations the lattice fitting procedure would have to be used 8,100 (i.e., 45 × 6 × 30) times.

The procedure of Rykunov et al. (1995) uses the dynamic programming algorithm to fit the real chain to the lattice. It is unsuitable for use here because it does not include virtual bond angle constraints, and it uses an approximate overlap penalty function that does not guarantee that the fitted chain will not contain overlaps, and hence violates the overlap constraint. We have devised a different dynamic programming lattice fitting procedure that guarantees that the lattice chain will meet all of the proper lattice constraints described above.

The objective for lattice fitting is to minimize the cost function:

$$F = \sum_{i=1}^{n} \|q_i - Q_i\|^2 , \tag{8}$$

where q_i refers to the coordinates of residue i in the real chain (cities) and Q_i refers to the coordinates of the lattice site chosen for the ith residue in the lattice chain (satellites).

For the moment, for simplicity, let us examine the properties of the global minimum fit (Q_i^* for $i = 1, n$) of the cost function if we exclude the virtual bond angle and overlap constraints. If we choose an arbitrary city i, then the part of the globally optimal chain from the first lattice point Q_1^* to the lattice point Q_i^* is also the optimal chain for the first i cities among all those on-lattice chains that start in Q_1^* and end in Q_i^*. It should be noted, however, that there may be lattice chains from 1 to i that have a lower cost, but they will not pass through lattice point Q_i^* and will have a higher total cost as they are extended to the end. Hence, the dynamic programming algorithm (which guarantees the optimal fit) proceeds from residue 1 to n at each stage, storing only the optimal paths to each satellite around the city under consideration. To proceed from city i to city $i + 1$, one calculates the cost of extending the optimal paths from each of the satellites of i to all the satellites of $i + 1$, storing only the optimal paths (one per satellite) of $i + 1$ (Fig. 16). It should be noted that any connections that violate the virtual bond length constraint are excluded. These optimal paths to the satellites of city $i + 1$ serve as the basis to extend the chain to city $i + 2$ and so on until the end of the chain.

We have extended the dynamic programming algorithm to include the virtual bond angle constraint in the lattice fitting procedure. Now what is needed to be stored is not the optimal path to each satellite of city i, but the optimal path for each satellite and the arriving direction (see Fig. 17). The analogous logical statement about optimally fit chains mentioned above holds here: if we choose an arbitrary city i, then the part of the globally optimal chain from the first lattice point Q_1^* to the lattice point Q_i^* is also the optimal chain for the first i cities among all those on-lattice chains that start at Q_1^* and reach lattice point Q_i^* arriving at that direction ($Q_i^* - Q_{i-1}^*$).

In contrast to the virtual bond constraint, the overlap constraint is nonlocal. With the inclusion of the overlap constraint, the subchains of the global minimum chain need not be optimal paths. An optimal path from the first lattice point Q_1^* to satellite Q_i^* may be excluded from the global minimum path by an overlap even though the global minimum path passes though satellite Q_i^*. Hence, the dynamic programming algorithm (an algorithm that stores the locally optimal paths) cannot be used. The algorithm that we use guarantees that the chain will not have any overlaps, but at the loss of guaranteeing that the global minimum cost chain is attained. The global minimum will be missed only if a particular set of circumstances occurs. These circumstances, and the specific algorithm for our dynamic programming method, which finds the optimal valid lattice chain, can be found in the Lattice fitting appendix.

The quality of our dynamic programming procedure was compared with the simple ith vector method. The ith vector method is the greedy algorithm that chooses the best local vector for each residue, proceeding from one end of the chain to the other (as described above). The dynamic programming method finds much better fits than the ith vector method (Fig. 18). In cases where the

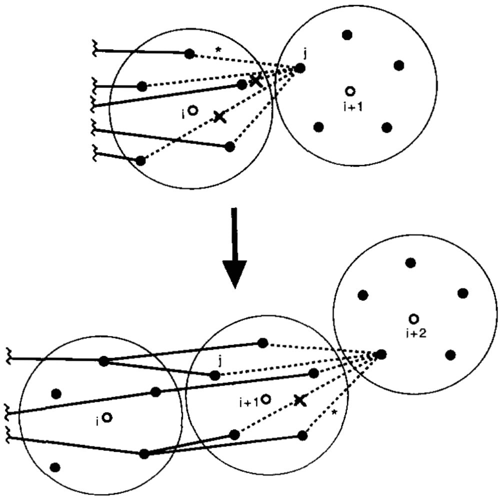

Fig. 16. Schematic diagram of the dynamic programming algorithm for only virtual bond length constraints. Solid lines indicate the optimal paths to arrive at each of the lattice sites (solid circles) in the neighborhood of city i (open circle). The neighborhood is denoted by the large circle enclosing the lattice satellites for each city. Dashed lines are the considered paths to satellite j around city i + 1. An X through the dashed lines denotes that the proposed path violates the virtual bond length constraint. An asterisk denotes the hypothetical optimal path to satellite j around city i + 1. The figure below the arrow indicates the continuation of the method to city i + 2. It should be noted that at most only one optimal path arrives at each satellite for the city being extended.

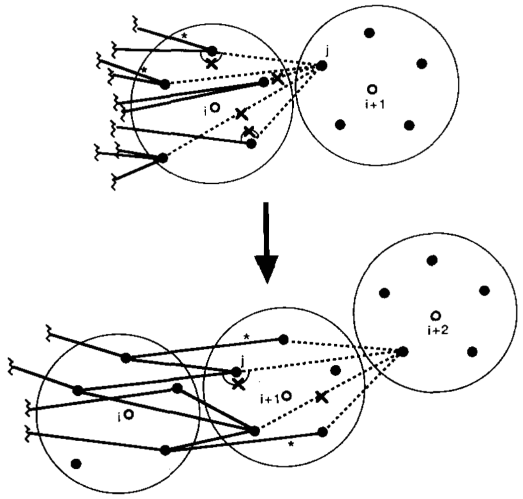

Fig. 17. Schematic diagram of the dynamic programming algorithm including both the virtual bond length and virtual bond angle constraints. Solid lines indicates the optimal paths to arrive at the lattice sites in the neighborhood of city i from *various* directions. Dashed lines are the considered paths to satellite j around city i + 1. An X through the dashed lines denotes that the proposed path violates the virtual bond length constraint. An X through the angle indicators denotes that the proposed path violates the virtual bond angle constraint and, hence, that path is rejected. The asterisks denote the optimal paths to satellite j around city i + 1 coming from the indicated directions. The figure below the arrow indicates the continuation of the method to city i + 2. It should be noted that multiple optimal paths arrive at each satellite (i.e., from different directions) for the current city (i + 1) being extended, but at most only one path arrives at each satellite for the previous city (i).

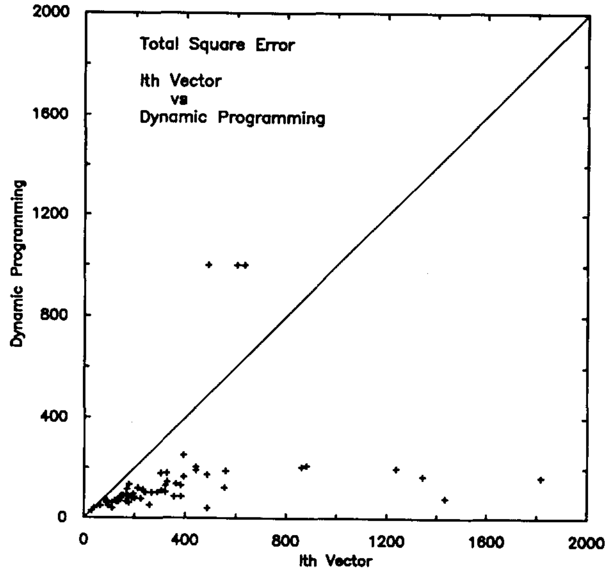

Fig. 18. Plot of the errors in fitting found with the dynamic programming and the *i*th vector methods. The dynamic programming algorithm finds fitted lattice chains superior to the *i*th vector method (all the points fall below the $y = x$ diagonal). The values on the axes are the total square errors for each respective method in Å^2. The three points above the diagonal are the cases where the dynamic programming method fails to find a solution as a result of choosing the size of the neighborhood around the cities that was too small. Those points are arbitrarily placed at 1,000 Å^2 on the dynamic programming axis for this plot. The conformations were generated with the Cartesian combination operator method using the wide λ data set.

*i*th vector method gives very large deviations (total square error greater than 800 Å^2), the dynamic programming algorithm finds good fits (less than 200 Å^2).

In three cases, for the data plotted in Figure 18, the dynamic programming method found no solution because the lattice satellite neighborhood around the cities was chosen to be too small. This illustrates a feature of the method. Choosing the neighborhood size (i.e., the number of lattice sites) around a city sets a maximum deviation around that city. This forces the lattice fit to be within a prescribed distance from each city. This allows the fit to be faithful to all parts of the chain even if it degrades the overall fit. This is desirable for use with the Cartesian combination operator genetic algorithm because it allows faithful inheritance of the attributes of the parents. Of course, the neighborhood size can be increased sufficiently to guarantee the overall best fit. It should be noted that the computer time for the dynamic programming method is very dependent on the size of the neighborhoods. For the neighborhood size chosen for the data in Figure 18, the dynamic programming method was ~300 times slower than the *i*th vector method.

As expected, the fitting by either method was found to be best for λ at the ends of the interval [0, 1] and worst for the middle values (Fig. 19). This is especially true for the *i*th vector method. The dynamic programming method still finds good solutions even for λ = 0.35 and 0.65.

The use of dynamic programming fitting improves the efficiency of the Cartesian combination operator genetic algorithm (Fig. 20). The price, however, is very slow execution time. The lattice fitting now takes as much CPU time as all the rest of the steps (Monte Carlo minimization, etc.). To take advantage of the improvement to the Cartesian combination operator method with use of dynamic programming, a dual approach is used. The fitting first uses the *i*th vector method and then, if the deviation with the

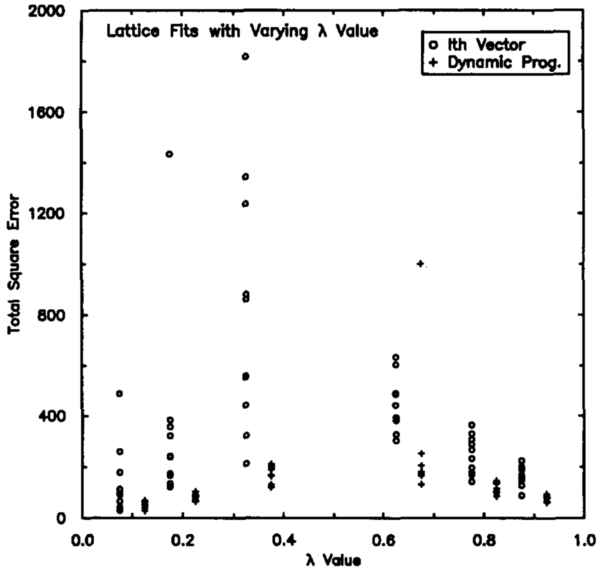

Fig. 19. Total square error (Å^2) for fitting the real chains generated with the Cartesian combination operator method to the lattice with $\lambda \in \{0.10, 0.20, 0.35, 0.65, 0.80, 0.90\}$. For clarity, the data shown on the plot have been shifted slightly to the left (O, *i*th vector) or to the right (+, dynamic programming). Three cases in which the dynamic programming method fails to find a solution have been assigned arbitrarily a value of 1,000 Å^2 and appear as a single + symbol at λ = 0.65.

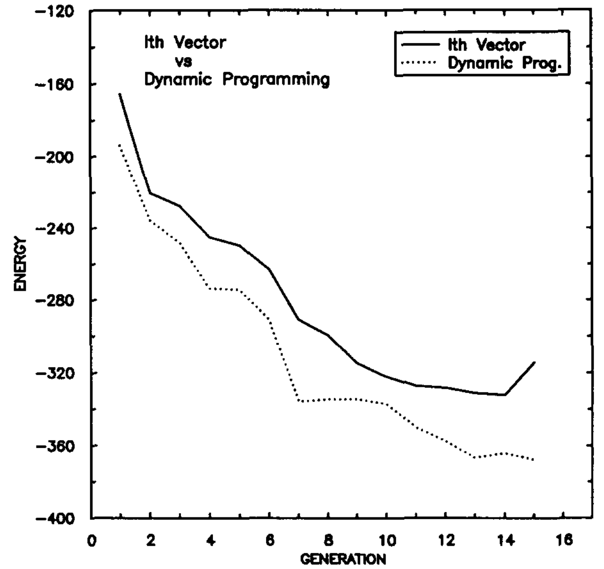

Fig. 20. The dynamic programming method markedly improves the efficiency of the Cartesian linear combination genetic algorithm. The initial population set was a subset of the conformations in the set IPS1, with N_{Pop} = 5. The Linco wide operator was used. For the dynamic programming method, a neighborhood radius of 3.75 Å was used. The dynamic programming fitting CPU time was ~700 times slower than the *i*th vector method.

off-lattice chain is too large, the dynamic programming algorithm is used.

Conclusion

The results presented here demonstrate that the Cartesian combination operator genetic algorithm can efficiently generate low-energy children that retain the topology and pairwise contacts of their parents. This illustrates the advantage that the Cartesian combination operator method has over traditional bond-swapping genetic algorithms. Traditional crossover operators retain only the local bond vectors or dihedral angles, disrupting some of the long-range contacts and the topology of the parent chains. Algorithms, such as Monte Carlo methods, can optimize local angles efficiently. The major problem in protein folding has been the global optimization of the long-range contacts. The potential used in this article consists of long-range contacts and topology constraints. The Cartesian combination operator method was devised to optimize the global folds of proteins, and the results demonstrate its ability to locate low-energy structures that retain native-like topology.

The Cartesian combination operator genetic algorithm is ideally suited for testing and optimizing knowledge-based potentials. The method efficiently generates good solutions to the knowledge-based potentials by searching the space of compact protein-like structures. This allows investigation of the effect of perturbing the knowledge-based potential or different parameterizations by generating a set of conformations that are of low energy for the potential. The potential can be changed in an iterative process to yield a potential function optimized for locating the native conformations of proteins. This task is being investigated currently.

The Cartesian combination operator method is likewise suited to the general task of searching for the native structure. By their very nature, knowledge-based potentials are a "mean field" type of potential. That is, the potential assigns a low energy to structures that are similar to those found in the database on which the potential was parameterized. It is likely that no single knowledge-based potential will ever be able to identify the native structure accurately for all proteins, i.e., assigning the global minimum energy to the native conformation and higher energies to all other alternate conformations. The best use of knowledge-based potentials may be to generate an ensemble of likely candidates for the native structure of a protein. A more physically based potential can then be applied to the search restricted to the neighborhood of the knowledge-based potential candidates. This hybrid approach may ultimately be the way to solve the protein folding problem. The Cartesian combination operator genetic algorithm is ideally suited to provide the initial candidates.

Acknowledgments

We thank J. Kostrowicki for providing the original outline of the dynamic programming scheme and many helpful suggestions on its implementation. This work was supported by grant no. GM-14312 from the National Institute of General Medical Sciences of the National Institutes of Health. A.A.R. was supported by a National Research Service Award from the Public Health Service. The computations were performed on the IBM Scalable Powerparallel supercomputer of the Cornell National Supercomputer Facility (CNSF), a resource of the Cornell Theory Center, which receives major funding from the National Science Foundation and the IBM corporation, with additional support from New York State, members of its Corporate Research Institute, and from the National Institutes of Health.

References

Dandekar T, Argos P. 1994. Folding the main chain of small proteins with the genetic algorithm. *J Mol Biol 236*:844–861.

Godzik A, Kolinski A, Skolnick J. 1993. Lattice representations of globular proteins: How good are they? *J Comp Chem 14*:1194–1202.

Goldberg DE. 1989. *Genetic algorithms in search, optimization and machine learning.* New York: Addison-Wesley.

Holland JH. 1975. *Adaptation in natural and artificial systems.* Ann Arbor, Michigan: University of Michigan Press.

Kolinski A, Godzik A, Skolnick J. 1993. A general method for the prediction of the three dimensional structure and folding pathway of globular proteins: Application to designed helical proteins. *J Chem Phys 98*:7420–7433.

Kolinski A, Skolnick J. 1994. Monte Carlo simulations of protein folding. I. Lattice model and interaction scheme. *Proteins Struct Funct Genet 18*:338–352.

Le Grand SM, Merz KM Jr. 1994. The genetic algorithm and the conformational search of polypeptides and proteins. *Molecular Simulation 13*:299–320.

Nemethy G, Gibson KD, Palmer KA, Yoon CN, Paterlini G, Zagari A, Rumsey S, Scheraga HA. 1992. Energy parameters in polypeptides. 10. Improved geometrical parameters and nonbonded interactions for use in the ECEPP/3 algorithm, with application to proline-containing peptides. *J Phys Chem 96*:6472–6484.

Nishikawa K, Ooi T. 1986. Radial locations of amino acid residues in a globular protein: Correlation with the sequence. *J Biochem 100*:1043–1047.

Pedersen JT, Moult J. 1995. Ab initio structure prediction for small polypeptides and protein fragments using genetic algorithms. *Proteins Struct Funct Genet 23*:454–460.

Rykunov DS, Reva BA, Finkelstein AV. 1995. Accurate general method for lattice approximation of three-dimensional structure of a chain molecule. *Proteins Struct Funct Genet 22*:100–109.

Sun S. 1993. Reduced representation model of protein structure prediction: Statistical potential and genetic algorithms. *Protein Sci 2*:762–785.

Sun S, Thomas PD, Dill KA. 1995. A simple protein folding algorithm using a binary code and secondary structure constraints. *Protein Eng 8*:769–778.

Unger R, Moult J. 1993. Genetic algorithms for protein folding simulations. *J Mol Biol 231*:75–81.

Appendix 1: Test potential appendix

The potential used to test the Cartesian combination operator method consists of four terms: radius of gyration and disulfide-bond topology constraints, a contact profile term, and a pairwise contact energy term.

Radius of gyration

$$E_{Gyr} = a\left\{1 - \exp\left[-\frac{1}{2b^2}(G - G^\circ)^2\right]\right\}, \qquad (9)$$

where G, G° are the radii of gyration (in Å) for the test and native conformations. The constants a and b were set to 0.226 energy units and 3.0 Å, respectively.

Disulfide-bond constraints

$$E_{Disul} = \sum_{i}^{pairs} a\Bigg\{1 - \exp\Bigg\{-\frac{1}{2b^2}(D_i - D_i^\circ)^2\Bigg]\Bigg]\Bigg\}, \qquad (10)$$

where D_i, D_i° are the squared distances between C^α coordinates of each disulfide pair, i, for the test and native conformations, respectively. The sum runs over all disulfide pairs. The constants a and b were set to 3.99×10^{-3} energy units and 170 $Å^2$, respectively.

Contact profile

$$E_{Profile} = \rho(P, P^\circ), \qquad (11)$$

where ρ is the correlation coefficient for profiles:

$$P(i) = \sum_{j=1}^{n} \delta_{ij}, \qquad (12)$$

where $\delta_{ij} = 1$ if residue j is within 10 Å of residue i, and 0 otherwise. P° refers to the native profile.

Pairwise contact energy

$$E_{Pairwise} = \frac{1}{2}\sum_{i}^{n}\sum_{|i-j|>1}^{n}\delta_{ij}e_{ij}. \qquad (13)$$

$\delta_{ij} = 1$ if the distance between residues i and j is less than 7.6 Å, and 0 otherwise; e_{ij} is the pairwise interaction energy, the parameters being taken from Kolinski et al. (1993).

The total energy is a sum of the terms above:

$$E_{Total} = c_1 E_{Gyr} + c_2 E_{Disul} + c_3 E_{Profile} + c_4 E_{Pairwise}. \qquad (14)$$

The coefficients were each adjusted independently. Each coefficient, in turn (setting all other coefficients to zero), was chosen so that the fraction of conformations accepted in the Monte Carlo two-bond move was 15% at a temperature of 1.0. The coefficients obtained in this manner were c_1, c_2, c_3, c_4 = 6.00×10^6, 1.50×10^6, -200, 3.75. For the Monte Carlo simulations conducted in this article, a temperature of 0.4 (corresponding to a smaller fraction of accepted conformations) was found to be the best for minimization and was used throughout. It should be noted that for the native conformation of a protein the terms E_{Gyr} and E_{Disul} equal 0, and the term $E_{Profile}$ equals 1. For the native conformation of the protein crambin (1crn), $E_{Pairwise}$ was -10.1 units. Therefore, for the native conformation of crambin, the total energy equals -237.8 units [i.e., $c_1(0.0) + c_2(0.0) + c_3(1.0) + c_4(-10.1)$].

The simple potential described above was devised to test the Cartesian combination operator method. A more complete potential, which includes explicit side chains, would be needed to attempt to predict the native structure of proteins. The need for a more complete potential to predict native structures can be seen from existence of non-native structures with lower energy than the native conformation. For example, the crambin conformations found with the Cartesian combination genetic algorithm in Figure 15A have energies of -374, -369, -358, -352, -345 units. These conformations have energies below the native value of -237.8 units, and their RMSDs were 8.9, 8.1, 6.2, 8.7, and 8.4 Å, respectively. This illustrates an important use of efficient minimization methods, such as the Cartesian combination genetic algorithm; they can be used to test and optimize potentials to create energy functions that can then be used to predict the native structure of proteins.

Appendix 2: Lattice fitting appendix

This appendix provides the details of our dynamic programming algorithm, which fits a real-space chain to a lattice, subject to bond length, bond angle, and overlap constraints (see Equations 5–8). Our algorithm proceeds from one end of the chain to the other. At each stage, only the optimal paths needed to reach each lattice site Q_i^j from direction v^k for all necessary js and v^ks are stored. More explicitly: let f_i^{jk} be the "global minimal" cost of arriving at lattice point Q_i^j from direction v^k when building the lattice chain from the first to the ith city (i.e., the ith real-space residue). In order to proceed to the next city, the matrix f_{i+1}^{jk} needs to be calculated for all js and ks. The connections to the satellites of city $i + 1$ (i.e., the lattice sites in the neighborhood of residue $i + 1$) are explored. Let us consider the lattice site Q_{i+1}^j because it is connected to a lattice site of the previous city ($Q_i^{j'}$) that meets the virtual bond length constraint. This defines the direction of arrival v^k. The algorithm proceeds through the sorted list (in order of increasing cost) of optimal paths $\{f_i^{j'1}, f_i^{j'2}, f_i^{j'3}, \ldots\}$ until it finds the first direction $v^{k'}$ (arriving at $Q_i^{j'}$) that meets the virtual bond angle constraint. The satellite Q_{i+1}^j is checked to see if it overlaps any of the satellites for that chosen path, and that path is rejected if violations are found. The path found in this manner is the "global minimal" path for arriving at lattice site Q_{i+1}^j from direction v^k. Then the additional cost $f_{i+1}^{jk} = f_i^{j'k'} + \| q_i - Q_i^{j'} \|^2$ is calculated. The algorithm proceeds in this manner for all satellites around city $i + 1$ connected to all satellites around city i. The vectors $\{f_{i+1}^{j1}, f_{i+1}^{j2}, f_{i+1}^{j3}, \ldots\}$ are sorted in increasing order of cost. The algorithm can now proceed to the $i + 2$ stage. At the last city, all the optimal paths for arriving at each lattice satellite of city n are compared. Of these paths, the lowest cost path is chosen, and it is the overall "global minimum" cost fitting.

In the above algorithm, global minimum appears in quotes. This is because the inclusion of the overlap constraint prevents the guarantee of generating the optimal cost path. The optimal path may be missed only if: a path (A) arriving at satellite j from direction v^k is not stored in favor of having saved a lower-cost path (B), arriving at the same satellite j from the same direction v^k, *and* path (B) is eventually discarded because of an overlap at a later stage, *and* the global minimum solution contains path (A). The likelihood of this occurrence needs to be investigated. For the needs of the Cartesian combination operator method, it is sufficient that the "dynamic programming" method generates very good fits without overlaps and ones that meet the virtual bond length and virtual bond angle constraints.