The ineffectiveness of recombination in a genetic algorithm for the structure elucidation of a heptapeptide in torsion angle space. A comparison to simulated annealing

A.H.C. van Kampen, L.M.C. Buydens *

Laboratory for Analytical Chemistry, Catholic University Nijmegen, Toernooiveld 1, NL-6525 ED Nijmegen, The Netherlands
Received 6 September 1996; accepted 20 December 1996

Abstract

Genetic algorithms comprise a family of stochastic optimization strategies, which are often applied to solve complex optimization problems. The combination of population based search and a recombination operator distinguishes the genetic algorithm from other global optimization techniques that often only comprise a (sophisticated) mutation-selection scheme. Investigations were conducted that suggest, however, that recombination is not always effective, i.e., crossover was unable to recombine the so-called building blocks that should produce improved trial solutions. In this research the contribution of the crossover operation to the performance of the genetic algorithm was examined for a structure elucidation problem of a heptapeptide. In addition, the performance of the genetic algorithm was compared to the alternative simulated annealing strategy. It was shown that the current design of the genetic algorithm did not promote the recombination of building blocks, and was therefore easily outperformed by simulated annealing. The strategy presented to reveal the effectiveness of recombination is straightforward, and can easily applied to other genetic algorithm applications.

1. Introduction

Genetic algorithms (GAs) [1–4] comprise a family of stochastic global optimization strategies that are applied to solve complex optimization problems. These problems are characterized by search spaces that contain multiple optima, and in which the problem parameters are non-linearly correlated. GAs are loosely based upon the principles of natural selection and population genetics, and although the basic algorithm is straightforward, application may sometimes be accompanied with pitfalls and difficulties. One of these pitfalls is the subject of this paper.

The crossover operation is the mechanism that distinguishes the GA from other (stochastic) global optimization methods (e.g., simulated annealing) that often comprise a sophisticated mutation and selection scheme [5–7]. The basic idea of crossover is to recombine trial solutions such that improved solutions emerge. More precisely, the crossover operator establishes the communication between strings via the exchange of the so-called building blocks (sub-solutions), which are highly fit schemata of low order and short defining length. [1,8]. A schema is a similarity template describing a subset of strings with similarities at certain string positions. Schemata of low order
and short defining length basically represent similarity templates with only few specified bits that are encoded close together on the string. Consequently, they are not easily disrupted by recombination [1,9]. According to the building block hypothesis, these building blocks are sampled, recombined, and resampled to form strings of potentially higher fitness. In a way by working with the building blocks, the complexity of the optimization problem is reduced; instead of building high-performance strings by trying every conceivable combination, the strings are constructed from the best partial solutions of past samplings. Holland argued that recombination of trial solutions may be beneficial, and necessary, because mutation alone may not properly exploit the search space, and therefore, the efficiency of the GA should basically emerge from recombination [3].

Deceptive optimization problems [1,10,11], in which recombination of low order building blocks lead to incorrect (sub-optimal) higher order building blocks can, however, also be solved by the GA. The reason behind this is that the dynamics of the GA ensure that building blocks are not only systematically propagated and recombined during succeeding generations, but are also destroyed and created due to accidental changes by crossover and mutation. These changes may also result in fit building blocks that finally lead to high quality solutions.

As stated by Whitley [12], deceptive problems are the only interesting problems for application of GAs. However, a problem occurs when the deceptiveness of the optimization problem becomes too severe, and none or only few combinations of fit building blocks lead to improvements. Consequently, the search degrades to an inefficient process in which trial solutions are only improved by accidental changes.

For this reason it is very important that constructive building blocks, which can be processed by the GA, exist. Accordingly, the combination of the fitness function, encoding of the problem parameters (e.g., real or binary), and type of crossover (e.g., uniform or n-point) must ensure the presence and processing of these building blocks. For complex nonlinear problems these building blocks will not necessarily have a physical interpretation, and may therefore be difficult (if possible at all) to identify, which can make a directed design of a GA a laborious process. For most real-world GA applications the existence and processing of building blocks is rarely checked, and therefore, little knowledge is available about whether or not the GA actually proceeds as is suggested by theory.

The non-linear correlations between the problem parameters are the cause for the deceptiveness of the optimization problem. If the GA is well-designed and promotes the processing of building blocks, then, in a way, the complexity of the optimization problem is reduced. Instead of combining the problem parameters of highly fit strings directly, which does not necessarily lead to improved solutions due to the correlation of the parameters, the GA is adding building blocks that are supposed to lead to improvements. However, for complex optimization problems it seems unlikely that such a transformation can be imposed by the GA design. Even if such a transformation exist, it will probably be quite difficult to find a suitable design. Therefore, one should at least check the contribution of the crossover operator.

To this end two experiments are available. The first one comprises a comparison of a CMS scheme (GA with crossover, mutation and selection) to a MS scheme (GA without crossover). If this experiment reveals no significant contribution of recombination, then the GA design should be changed. If, on the other hand, crossover does contribute, then a second experiment is needed to test whether solutions are really constructed from the recombination of building blocks. For this experiment the so-called random crossover operator [13] can be used. If these experiments reveal that recombination is without the desired effect, the resulting algorithm is left with only mutation and selection, and consequently, will probably be easily outperformed by a more sophisticated mutation selection scheme like simulated annealing.

This paper presents a study for the effectiveness of recombination when the GA is applied to the elucidation of the spatial structure of a heptapeptide in torsion angle space. Different operators and values for the GA control parameters were tested, and the best found parameterization was compared to the performance of a standard simulated annealing approach. The paper does not claim to make any generalization of its conclusions to other GA applications, but emphasizes an additional pitfall in the GA design. The functionality of the recombination operator should not be taken for granted.
2. Methods

2.1. Software

The GA application described in this paper was written in standard ANSI C, with use of the toolbox GATES (genetic algorithm toolbox for evolutionary search), version 2.0 [14,15].

2.2. Problem description

During the last few years the GA [16–21] and simulated annealing (SA) [22–25] methods have extensively been applied to the structure elucidation problem of biological macromolecules from NMR data. Knowledge about the spatial structures is considered important as it may contribute to a more thorough understanding of their biological function. Multidimensional NMR spectroscopy has become the state-of-the-art method for the structure determination of biological molecules in solution [26–28]. Distance restraints obtained from these experiments, complemented with knowledge about the covalent structure of the molecule may lead to a description of its spatial structure.

In the present investigation the molecule was represented by its internal coordinates (torsion angles), which were the subject of optimization, and preserves the covalent structure during optimization. The optimal set of torsion angles was defined as a conformation that did not violate the experimental distance restraints, and simultaneously was free of atom overlap. Accordingly, the fitness function was defined as:

\[
\text{Fitness}^{-1} = \text{Error} = A \times \sum \text{(restraint violations)} + B \times \sum \text{(atom overlap)}^2
\]

where \( A \) and \( B \) were weight factors (which were set to 2.0 and 1.0 respectively), ‘restraint violation’ is the magnitude of a single violation of a distance restraint, and ‘atom overlap’ is the magnitude of the overlap between two atoms.

For the experiments presented in this paper the natural heptapeptide optoid, \( \mu \)-selective dermorphin (Fig. 1), was taken as the molecule of interest [29–31]. This molecule consists of 7 amino acids (Tyr–dAla–Phe–Gly–Tyr–Pro–Ser–NH\(_2\)), and was described by 31 torsion angles. A set of 19 distance constraints was available from [32], and used in the experiments presented in this paper. Since heptapeptides are generally flexible in solution, and because only a limited set of distance restraints was used, it was expected that more than one conformation, con-

Fig. 1. The natural heptapeptide optoid \( \mu \)-selective dermorphin (Tyr–dAla–Phe–Gly–Tyr–Pro–Ser–NH\(_2\)).
sistent with the input data, could be found, i.e., the optimization problem was degenerated. Since degeneration occurs in a range of optimization problems, this specific system may, to some extent, be representative for these kind of optimization problems, and therefore, presented a suitable test case for testing the effectiveness of recombination. It was not the aim of this paper to reveal the ‘true’ conformation of this peptide, because this is already known from other publications. However, such a study would certainly require additional input constraints. In this paper only the final error values, and number of function evaluations were considered.

2.3. Simulated annealing

Simulated annealing (SA) [6,33,34] derived its name from the annealing process of metals that results in a crystalline configuration (low energy state). The principle of simulated annealing is simple. Starting from randomly initialized problem parameters \( p \) (torsion angles), a trajectory through space is generated by making (small) modifications to these parameters. For the present SA application, the parameters were modified by increasing or decreasing their current value with a amount \( \xi \), generated from a Cauchy distribution [35]:

\[
P_{\text{new}} = P_{\text{current}} \pm \xi
\]

If evaluation of the new set of parameters results in a decrement of the error value \( (E) \), then the new parameter values are accepted, otherwise they are only accepted with a probability \( P_{\text{accept}} = e^{(-E_{\text{new}} - E_{\text{old}})/\xi} \). The control parameter \( c_{i} \) is initialized such that most detrimental steps are accepted, and consequently, the early stage of the optimization process resembles a random search. After making \( N \) steps through the search space, the value of \( c \) is decreased according to \( c_{i+1} = c_{i} \times \alpha \) (\( \alpha \) was taken 0.95 for the experiments described in this paper), giving rise to a lower acceptance probability. Subsequently, a new trajectory through space is generated. With decreasing \( c_{i} \), the chance for accepting detrimental steps decreases, which finally forces the algorithm to a local search where only improvements are accepted.

The SA algorithm in this research was parameterized such that the \( c \) was decreased when \( N = 50 \) steps were accepted, or a maximum of 250 proposed moves was exceeded. The initial value of \( c \) was chosen such that the initial ratio between accepted moves and proposed moves was about 0.9.

2.4. Statistical evaluation of results

The results (error values and number of function evaluations) obtained in the experiments were evaluated by the Kruskal–Wallis non-parametric analysis of variance (ANOVA) [36–39]. The \( p \)-values that result from such a calculation represents an increasing index of reliability that is involved in accepting the hypothesis that the tested groups of observations were sampled from an identical population. Typically, in many sciences, results that yield \( p \leq 0.05 \) are considered borderline statistically significant. Results that are significant at the \( p \leq 0.01 \) are commonly considered statistically significant, and \( p \leq 0.005 \) or \( p \leq 0.001 \) are called highly significant. These categories were used for the interpretation of the data in this paper.

Part of the results were visualized with Box–Whisker plots [40]. The box borders were defined by the 25% and 75% percentiles. The median value of the observations was indicated by a square inside the box, and the minimum and maximum values were indicated by the Whiskers. Outliers were depicted by an open circle, and extreme values by a star (‘*’). A value was rated to be an outlier if the distance from the box border was larger than 1.5 times the length of the box. If the distance was larger than 3.0 times the length of the box, then that value was regarded as an extreme value. The values 1.5 and 3.0 were taken from [40], and only served the visualization of the results.

2.5. The random crossover operator (RXO)

The RXO operator [13] was developed to examine whether building blocks are recombined productively by the crossover operator. Fig. 2 depicts the function of this operator. A parent string is selected from the

![Diagram of the random crossover operator (RXO).](image)
population, and recombined with a string that was generated at random. Due to the application of this operator, communication between strings of the population is no longer present, i.e., no building blocks can be exchanged. Thereupon, no knowledge is available about possible building blocks on the random string because it was not evaluated by the fitness function. Consequently, this type of crossover is equivalent to a (macro)mutation operator. Suppose the following RXO operation (with a 2-point crossover. The $\uparrow$'s denote the breaking points):

$$\text{random string} \ 100\{001\}1100$$

$$\text{parent string} \quad 110\{010\}1001 \xrightarrow{\text{crossover}} 1100011001$$

The same result could, however, be obtained by a macro-mutation in which only the bits between the breakpoints were mutated:

$$\text{parent string} \quad 110\{010\}1001 \xrightarrow{\text{mutate}} 1100011001$$

Therefore, if RXO performs comparable to a regular crossover operator, it can be concluded that also the latter acts like a mutation, and solutions do not emerge from building block recombination (which should lead to a superior performance).

It must be remarked that the mutation imposed by RXO is not precisely identical to a possible mutation imposed by a regular crossover operator. This is due to the fact that the strings in the GA population become more similar as optimization proceeds, which diminishes the effective mutation rate caused by crossover. The mutation rate imposed by RXO remains constant because before every application a new random string is generated. Accordingly, interpretation of results obtained with RXO should be done carefully.

In order to test the principle of the RXO operator, an experiment was conducted that comprised a comparison between a GA with the RXO operator, and a GA that used uniform crossover for the minimization of the function $y = \sum_{i=1}^{20} x_i^2$. For this simple minimization problem the final solution is a string with every bit set to zero. The problem contained no nonlinear interactions between the problem parameters, and consequently every combination of zeros in the strings comprised a building block that is easily processed by uniform crossover. The performance of RXO should therefore be inferior to these regular types of crossover. The result of this experiment is shown in Fig. 3, which denotes the number of function evaluation needed to derive at the solution of the problem ($y = 0.0$). This figure clearly demonstrates that the processing of building blocks by uniform crossover easily outperforms a (macro)mutation operator (RXO).

3. Results and discussion

3.1. Optimization of the genetic algorithm

Before establishing the GA performance, the effect of crossover, and comparing the GA to SA, the parameterization of the GA was optimized. Experimental designs were setup to determine the main effects of several control parameters and operators. The interactions between the GA parameters were neglected because their determination would require too many experiments consequently, the final parameterization was at most near optimal if it is assumed that the interaction effects were small compared to the individual main effects.

The ‘number of function evaluations’ for the GA experiments was calculated from the population size and the number of generations after which no further improvement of the best string in the population was observed. A maximum of 1000 generations was imposed on each GA run.

The first experiment was set up to determine a
suitable population size. Population sizes of 50 and 200 strings were tested with 10 experiments at each level, starting from different initial populations. During these experiments rank-based selection [41] was used with the threshold defining the best 25% of the population from which strings were selected at random. Elitism selection was used to copy the best 5% of the strings to the next population. These strings remained unaffected by crossover and mutation. The crossover and mutation probabilities were set to 0.70 and 0.05 respectively. Uniform crossover was used, and exchanged 30% of the bits between two strings on application. A normal point mutation was used. Each problem parameter was encoded with a 11 bit Gray coding. The results are depicted in Fig. 4, which denotes the Box–Whisker plots for both levels. An ANOVA calculation revealed no significant difference ($p = 1.000$) between the two population sizes for the error values. Consequently, the smaller population size was chosen for the remaining experiments, because it was about four times faster.

Subsequently, the threshold fraction for rank-based selection, the elitism percentage, crossover probability ($P_c$), crossover type, number of bit exchanges for uniform crossover, the number of encoding bits for each problem parameter, and the type of parameter encoding were optimized with the use of several experimental designs. Table 1 depicts the levels at which each parameter was tested, and the number of experiments that were conducted at those levels.

An ANOVA revealed that none of these factors were highly significant, i.e., both the error values and the number of function evaluations were comparable for all tested levels. Table 2 depicts the $p$-values for

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Level</th>
<th>#Exp</th>
<th>Parameter</th>
<th>Level</th>
<th>#Exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elitism percentage</td>
<td>5%</td>
<td>9</td>
<td>type crossover</td>
<td>uniform</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>10%</td>
<td>9</td>
<td></td>
<td>1 point</td>
<td>6</td>
</tr>
<tr>
<td>Threshold fraction</td>
<td>0.10</td>
<td>6</td>
<td></td>
<td>2 point</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.25</td>
<td>6</td>
<td>$P_c$</td>
<td>0.5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>0.40</td>
<td>6</td>
<td></td>
<td>0.7</td>
<td>6</td>
</tr>
<tr>
<td>#Bits</td>
<td>7</td>
<td>6</td>
<td></td>
<td>0.9</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>6</td>
<td>encoding</td>
<td>regular binary</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>6</td>
<td></td>
<td>gray binary</td>
<td>12</td>
</tr>
<tr>
<td>Bit exchange fraction</td>
<td>0.1</td>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.3</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 2

<table>
<thead>
<tr>
<th>Parameter</th>
<th>$p$-value</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>error</td>
</tr>
<tr>
<td>Elitism percentage</td>
<td>0.05</td>
</tr>
<tr>
<td>Threshold fraction</td>
<td>0.19</td>
</tr>
<tr>
<td>#Bits</td>
<td>0.72</td>
</tr>
<tr>
<td>Bit exchange fraction</td>
<td>0.92</td>
</tr>
<tr>
<td>Type crossover</td>
<td>0.58</td>
</tr>
<tr>
<td>$P_c$</td>
<td>0.72</td>
</tr>
<tr>
<td>Encoding</td>
<td>0.02</td>
</tr>
</tbody>
</table>

Each factor for the error values and function evaluations. Only an elitism fraction of 10%, and the use of a 7 bit regular binary encoding were slightly better, and consequently, were used in the remaining experiments.

Next the effect of the mutation probability ($P_m$) was determined. Five different levels were tested with 10 experiments at each level. The levels ranged from $P_m = 0.0$ (only crossover, i.e., CS scheme) to $P_m = 0.3$. The results are depicted in Fig. 5, which clearly demonstrates the impact of the mutation probability. The performance of the CS scheme is, as expected, very poor from which it is concluded that mutation is necessary to introduce new (lost) information in the population. A mutation rate that is too low or too large also produced poor results. A comparison of the levels $P_m = 0.01$, $P_m = 0.05$, and $P_m = 0.1$ revealed that a mutation rate of 0.05 produced better results ($p = 0.007$ for the error value, $p = 0.040$ for the number of evaluations).

3.2. Performance of the optimized genetic algorithm

The final parameterization of the optimized GA is depicted in Table 3. To test the performance of this GA, 84 experiments were carried out in which only the initial random population was varied. Fig. 6 depicts the Box–Whisker plots, and the actual distributions for the error values and number of function evaluations (the actual values are depicted in Table 4). It is clear that there is a large deviation for both quantities ranging from poor to well defined structures. From this it was concluded that the GA was not always able to converge to a well-defined structure despite the existence of multiple conformations. The histograms show that neither the error values or the number of function evaluations were normally distributed (which justified the use of the non-parametric ANOVA). Further investigations revealed that there was no correlation between the error values and the number of function evaluations.

3.3. Effectiveness of recombination

In order to reveal the effect of crossover on the search performance, 10 experiments were carried out without crossover (MS scheme), and compared to the results of the CMS scheme. The results are shown in Fig. 7. An ANOVA revealed no significant differ-

Fig. 5. Effect of the mutation probability. A mutation probability of $P_m = 0.0$ denotes the CS scheme. At each level 10 experiments were conducted.
ence between the CMS and MS scheme for the error values \( p = 0.740 \), or number of function evaluations \( p = 0.880 \). From this it seems fair to conclude that crossover did not contribute to the performance of the GA. One could argue that the CMS scheme produced one solution \( E = 2.30 \) that was slightly better than the best solution obtained with the

![Graphs](image)

Fig. 6. Box–Whisker plots and distribution of error values and function evaluations resulting from 84 experiments with the optimized CMS scheme.

### Table 3
Parameterization of optimized GA

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range of each parameter</td>
<td>(-180^\circ - +180^\circ)</td>
</tr>
<tr>
<td>Encoding type</td>
<td>regular binary</td>
</tr>
<tr>
<td>Number of bits per parameter</td>
<td>7</td>
</tr>
<tr>
<td>Selection method</td>
<td>rank-based selection</td>
</tr>
<tr>
<td>Threshold fraction</td>
<td>0.25</td>
</tr>
<tr>
<td>Elitism fraction</td>
<td>10%</td>
</tr>
<tr>
<td>Crossover type</td>
<td>uniform</td>
</tr>
<tr>
<td>Crossover probability</td>
<td>0.7</td>
</tr>
<tr>
<td>Number of bit exchanges (fraction)</td>
<td>0.3</td>
</tr>
<tr>
<td>Mutation type</td>
<td>point mutation</td>
</tr>
<tr>
<td>Mutation probability</td>
<td>0.05</td>
</tr>
</tbody>
</table>

### Table 4
Comparison of the error values and function evaluations for the GA (optimized CMS scheme) and SA

<table>
<thead>
<tr>
<th></th>
<th>GA</th>
<th>SA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>error</td>
<td>evaluations</td>
</tr>
<tr>
<td>Mean</td>
<td>3.39</td>
<td>40734</td>
</tr>
<tr>
<td>Median</td>
<td>3.56</td>
<td>44100</td>
</tr>
<tr>
<td>( P_{25} )</td>
<td>2.69</td>
<td>37875</td>
</tr>
<tr>
<td>( P_{75} )</td>
<td>4.03</td>
<td>47775</td>
</tr>
<tr>
<td>Minimum</td>
<td>2.30</td>
<td>7100</td>
</tr>
<tr>
<td>Maximum</td>
<td>5.95</td>
<td>49850</td>
</tr>
</tbody>
</table>

MS scheme \( E = 2.54 \), and that this might be an effect of crossover. It is, however, more likely that this is an effect only of the initial population, and consequently, repeating the MS scheme more often is likely to result in an equally good solution. Even if one wishes to conclude that the CMS scheme gave slightly better results, this would not be of much use if one is not prepared to repeat the CMS scheme for
many times in order to increase the probability that such a solution is included.

An additional experiment with the RXO operator strengthens the conclusion that recombination was not effective. Fig. 8 depicts the results of 40 experiments with the RXO operator in comparison to the CMS scheme. This figure indicates that RXO is capable of producing equally well-defined structures, but on average is inferior to uniform crossover (\( p = 0.005 \)). Since the comparison of the CMS scheme to the MS scheme revealed no significant difference, it is likely that this is caused by the constant mutation rate imposed by RXO (see method section). Furthermore, the processing of building blocks is likely to accelerate the convergence rate of the algorithm, which is not the case in the present experiment. From this it may be concluded that no building blocks are recombined productively. Consequently, one is left with a simple MS scheme (selection and point mutation) that is likely to be outperformed by a more advanced MS scheme like SA.

3.4. Simulated annealing

To test this hypothesis, thirty SA experiments were performed, and compared to the CMS scheme (Fig. 9). There was no significant difference between the error values for the two search strategies (\( p = 0.200 \)). Table 4 depicts the actual values for the Box plots. The number of function evaluations for SA was, however, about a factor 3 times less than the number of evaluations required by the GA. Again one could argue that the GA produced one better solution, but again this is probably an effect of the initial starting point in the search space, and repeating SA more often is likely to result in a comparable solution. Further analysis is necessary to investigate whether the SA is indeed a more effective optimization method.
thermore, a simple and straightforward instance of the SA algorithm was applied, and not optimized like the GA.

3.5. Diversification scheme

Since it may be expected that crossover has the largest impact at the beginning of the optimization process (because then the strings are highly dissimilar), a diversification scheme that delays convergence may help to enhance the contribution of recombination. A crowding scheme [1] was implemented and 20 experiments were conducted. The similarity measure used by the crowding scheme was based on the Euclidean distance between the parameters on the strings. A maximum of 5000 generations was allowed. The results of this experiment were very poor, i.e., the error values were significant higher ($p = 0.000$) than for the CMS scheme, and the required number of function evaluations was about 4 times larger. Due to these results, a diversification scheme was not longer considered.

3.6. Small population size

In order to investigate if the number of function evaluations needed by the GA could be reduced, while maintaining the quality of the solutions, 40 GA experiments with a population size of 25 strings were conducted, and compared to the optimized CMS scheme. The resulting solutions for this small population size were inferior to those found with a population size of 50 strings ($p = 0.001$). Thereupon, the number of function evaluations needed to derive at the final solutions was still significantly larger than for SA. From this it was concluded that decreasing the size of the population did not lead to results that were more comparable to SA, and the present choice for the population size is adequate.

The results of the present study are hard to generalize to other problem domains, or even to the same type of problems. Penersen and Moult [42] gave a brief overview of GA applications for protein structure determination, and discuss a comparison to molecular dynamics and Monte Carlo methods. They suggest that the GA may be preferable in some situations, but no details were given on the effectiveness of recombination, and no remarks were made about simulated annealing. Sun [43] also performed a comparative study on GAs and SA for protein structure determination, and showed that the GA was about a factor 100–200 times faster than SA. Again no details were given on the effectiveness of recombination.

From this research it should be evident that every GA application should at least be tested for the effectiveness of crossover. This provides more knowledge about applications in which the crossover operator does contribute to the search performance, and thereby may provide useful information for the design of new applications. The presented strategy for testing the effectiveness of recombination is straightforward, and may easily be applied for other GA applications.

It should be noted that evolutionary algorithms [44] that rely more strongly on mutation have been investigated for many years in the fields of evolutionary programming [45,46], and evolutionary

![Box plots comparing GA and SA](image_url)
strategies [47]. These techniques have been applied various problems with equal success as genetic algorithms [48–50]. Recombination might enhance the search but a comparative studies of GAs in which building blocks are exchanged are necessary.

In a way this research is related to the ongoing discussion about real-coded GAs. For these GAs schemata may be identified [51], which provide evidence that binary encoding is not the only possible approach. A comparison between binary and real-coded GAs must incorporate a test for the effectiveness of recombination. Without such a test, comparing the performances of both instances is only of limited use.

4. Conclusion

This research demonstrated that for the presented problem of structure elucidation in torsion angle space, the crossover operator was not able to contribute significantly to performance of the genetic algorithm. The combination of fitness function, parameter encoding, and crossover type did not promote the processing of building blocks. The resulting genetic algorithm was therefore basically based on selection and point mutation, and outperformed by the more sophistication mutation-selection scheme imposed by simulated annealing. More precisely, the conformations found by the genetic algorithm were of comparable quality to the conformations found by simulated annealing. However, simulated annealing converged about a factor three times faster than the genetic algorithm. Although many possibilities exist to make changes to the genetic algorithm design, it remains unclear how to change the design in order to promote the exchange of building blocks. The presented strategy to test the effectiveness of recombination is straightforward and can easily be adopted for other genetic algorithm applications.

Acknowledgements

F.F.A. Hollander is acknowledged for his contribution in writing a large part of the software to manipulate structures in torsion angle space.

References
