Journal of Mathematical Chemistry Vol. 28, No. 4, 2000

Optimization of transition state structures using genetic algorithms *

Sharene D. Bungay [a,**], Raymond A. Poirier [a] and Richard J. Charron [b]

[a] *Department of Chemistry, Memorial University of Newfoundland, St. John's, NF, Canada*
E-mail: sharene@math.mun.ca; rpoirier@mun.ca
[b] *Department of Mathematics and Statistics, Memorial University of Newfoundland, St. John's, NF, Canada*
E-mail: rcharron@math.mun.ca

Received 23 October 2000

Geometry optimization of transition state structures (first order saddle points) has proven to be a challenging problem in theoretical chemistry. Despite many attempts, no method has been developed that can guarantee convergence to a transition structure. The well-known method of genetic algorithms (GA's) was adapted for this problem, and designed to seek points on a potential energy surface with a zero gradient norm and one negative eigenvalue in the Hessian. A description of genetic algorithms and the software written to optimize first order saddle points is given. The software developed was tested on a mathematical function having minima, maxima, and first order saddle points. The method was capable of finding all of the saddle points, as the results presented demonstrate. Optimization of various transition state structures was then attempted. Although the current genetic algorithm software requires long run times, the algorithm will preferentially seek first order saddle points, weeding out any other stationary points. Thus, the initial guess at the optimum is not as critical as with other methods, and as well, multiple saddle points can be found.

KEY WORDS: genetic algorithms, optimization, transition state structures, fitness value

1. Introduction

Geometry optimization of chemical structures has long been an issue in theoretical chemistry. In particular, transition state structures, which are energy maxima along the minimum energy path connecting reactants and products in a chemical reaction, have posed fundamental difficulties. A computational approach for optimizing such structures is important since they have a fleeting existence and are therefore impossible to isolate experimentally. Various approaches exist for optimizing transition state structures, but no methods exist that can guarantee convergence to a transition state structure. For example, methods such as Broyden–Fletcher–Goldfarb–Shanno (BFGS) [1–4], and

* This research was supported by the Natural Sciences and Engineering Research Council of Canada.
** Present address: Department of Mathematics and Statistics, University of Guelph, Guelph, ON, Canada.

0259-9791/00/1200-0389$18.00/0

Optimally Conditioned (OC) [5] by Davidon are specifically designed to locate minima since the force constant matrix (Hessian) is forced to remain positive definite. In addition, most of these methods require "chemical intuition" in the form of a good initial guess to achieve convergence to a transition state structure. A method developed recently by Anglada et al. [6] called Transition-State-BFGS (TS-BFGS) was designed specifically for optimizing transition state structures. However, this method requires a good initial structure, as well as a good initial Hessian. This paper explores the use of the well known technique of genetic algorithms [7] to optimize the geometry of transition state structures [8].

Genetic algorithms are a type of evolutionary computing used in many disciplines for optimization problems. Genetic algorithms probabilistically search the problem's solution space by manipulating a population of possible solutions (*individuals*). Each individual consists of a string of binary bits, known as a *chromosome*, which encode its values of the problem variables. Genetic algorithms evolve better solutions to the problem being solved by applying various genetic operators to parents selected from the population. These operators combine and mix the genetic information (bits) of the parents to form offspring. The goal is to create offspring which are *more fit* than the parents. The *fitness* of an individual is a numerical value representing how well its variable values solves the problem. Simulation of reproduction via the application of genetic operators such as crossover and mutation continues for a number of generations, until a convergence criteria is satisfied.

Previous use of genetic algorithms for chemical structures include energy minimization of molecular clusters by Mentres and Scuseria [9], as well as various conformational searches [10–12]. A review of the use of genetic algorithms in chemistry was written by Judson [13].

2. Chemical features

The critical points on a general surface include minima, maxima, and saddle points of various order. All of these points are characterized by a zero gradient but are distinguishable by the number of negative eigenvalues of the second derivative matrix (Hessian). For chemical reactions, different structures represent points on a potential energy surface, where the coordinates are the bond lengths, bond angles, and dihedral angles. On this surface, minima represent the reactants, products, and intermediates in the reaction. First order saddle points represent transition state structures, and higher order saddle points are of no chemical interest. Thus, when looking for transition state structures we are interested in points with a zero gradient and one negative eigenvalue in the Hessian matrix. The construction of a genetic algorithm for a particular problem requires problem specific information, used to bias the evolution toward the desired optima. Thus, the known characteristics of first order saddle points will be used for this purpose.

3. Implementation of genetic algorithms

Similar to other geometry optimization methods, the genetic algorithm implemented begins with an initial approximation of the optimum structure sought. The encoded variables are the internal Z-matrix coordinates of the structure, bond lengths, bond angles, and dihedral angles (torsions). Since this data is in real number space, an encoding scheme is required to convert these real numbers into binary strings. Two encoding schemes were used. The simplest was *multiplicative* encoding where each variable was multiplied by $10^{accuracy}$ where *accuracy* is the number of accurate decimal places required in the optimum. We chose a value of 6. The value resulting from the multiplication was truncated to an integer and the binary representation of this integer was used as the encoded variable. The binary strings of each of the variables were then concatenated to form a longer binary string which formed the individual's chromosome. A more common encoding scheme which was also used is *interval* (or *range*) encoding. A domain $[a_i, b_i]$ is specified for each of the problem variables. The number of bits, n_b, used to represent each variable is chosen to divide the domain up into $2^{n_\mathrm{b}} - 1$ intervals. The integer representation of each variable x_i is given by

$$D = \frac{x_i - a_i}{b_i - a_i}\left(2^{n_\mathrm{b}} - 1\right) \tag{1}$$

The binary representation of the resulting integer is used as the encoded variable and the binary strings of each variable are concatenated as before.

In addition to these encoding schemes, Gray encoding was also implemented. Gray encoding is an alternative to standard binary and has the feature that the Hamming distance is constant. That is, consecutive integers encoded in Gray differ by only one bit flip.

The initial population of individuals (of size μ) was generated by adding small random perturbations to the initial approximation in the case of multiplicative encoding. These perturbations were restricted to a user defined region. For the interval encoding case, individuals were generated randomly and mapped to lie within specified intervals, with the initial guess included in the population unmodified.

The genetic operators act on the encoded bit strings of the individuals, whereas the fitness evaluation uses the real valued data to assign numerical fitness values. The variables are decoded by reversing the procedure used to encode them, and a problem dependent fitness function is evaluated for the variable set. Since the fitness function is the mechanism for directing the evolution towards the desired objective, it generally incorporates any *a priori* knowledge of the optimum sought. For finding a first order saddle point, the defining characteristics of the optimum are a zero gradient length and a Hessian matrix with one negative eigenvalue. Therefore, the fitness function used was

$$f = \frac{1}{\|\vec{g}\| + \varepsilon} \cdot \frac{1}{(n-1) + \varepsilon} \tag{2}$$

where $\|\vec{g}\|$ is the L_2-norm of the gradient vector divided by $\sqrt{k}$ (where k is the number of variables), n is the number of negative eigenvalues of the Hessian matrix, and ε was chosen small to prevent division by zero. As a result of applying this fitness function, individuals close to a first order saddle point will have large fitness values and, hence, will dominate the evolution after a number of generations.

Following the assignment of fitness values to all of the individuals in the initial population the reproduction process of selecting parents, and applying the genetic reproduction operators begins. Various methods exist for selecting parents, all of which bias the more fit individuals in some way. Two of these methods that were implemented are *roulette wheel* selection, and *tournament* selection. Roulette wheel selection is equivalent to assigning individuals a slice of a circle with the size of the piece proportional to that individual's fitness value. A random number is generated and the individual whose slice this number falls in, is chosen as a parent. Therefore, the more fit individuals are more likely to be chosen. Tournament selection involves randomly selecting a number of individuals to take part in a tournament. The individual in this pool with the highest fitness is chosen as a parent. As a result, the larger the tournament size the more likely that an individual with a high fitness value will be chosen. The tournament size (t_{size}) is variable and the user can specify a probability (t_{prob}) with which to choose the individual with the highest fitness. If this probability condition is not satisfied a random individual is chosen from the current tournament pool.

The genetic operators used include crossover and mutation. Crossover is a means to generate better individuals than those present in the previous generation. Two parents are selected as above, their chromosomes are aligned and a random crossover point is chosen. The bit strings are then crossed by exchanging the bits to the right of the crossover point, forming two offspring. This type of crossover is called *single-point crossover* and is performed with a user defined probability, p_{c}. Mutation is used to maintain diversity in the population. Each bit of each individual is examined for mutation, and is flipped with a user defined probability, p_{m}.

Following the reproduction stage the fitnesses of the offspring are evaluated and the generation is complete. Individuals are now chosen to take part in the next generation. This surviving population can consist of all of the offspring produced, or can be a combination of offspring and parents. In the latter case, the best individual from the previous generation is added to the surviving population; this is called *elitism*. The surviving population proceeds to the reproduction stage as before. This cycle continues until convergence is achieved, at which point the algorithm terminates and reports the optimum found.

Convergence criteria can very widely between GA implementations. A maximum number of generation (G_{max}) can be performed, after which convergence is assumed, and the best individual found is reported. Another possibility is to place a tolerance on the gradient norm such that the algorithm evolves until $n = 1$ and $\|\vec{g}\| < \delta$ where δ is user specified.

4. The software

The current implementation of the genetic algorithm includes several input parameters used in the setup and operation of the algorithm, many of which were mentioned in the previous section. These parameters are summarized in table 1. The parameters M_{init} and M_{sub} are used with the multiplicative encoding scheme to ensure that individuals representing physically infeasible solutions are not included in the population. For the generation of the initial population, all individuals are within M_{init} of the initial guess provided. Individuals in all subsequent generations are restricted to within M_{sub} of this initial guess.

The method for forming the surviving population that consists of both parents and offspring involves creating a pool of individuals whose fitness values are above the average fitness. If this pool has enough individuals (greater than or equal to μ) then a random selection of individuals is used. Otherwise, all are copied and the remainder of the surviving population consists of individuals created by mutating those that are already present.

Finally, λ_{tol} is a tolerance that is placed on the value of the negative eigenvalues, below which they must lie to be counted as negative. This parameter is required due to finite machine precision which can cause values that are essentially zero to be very small, and maybe negative.

5. Mathematical results

As a means of testing the algorithm implemented, the following mathematical function presented in Chong and Zak [14] for function minimization via genetic algo-

Table 1
Input parameters required in the current implementation of a genetic algorithm.

Variable	Description
x_0, y_0	initial guess at the optimum
μ	number of individuals in the population
G_{max}	maximum number of generations to perform
p_{c}	probability of performing crossover
p_{m}	probability of performing mutation
n_{b}	number of bits used to encode each variable
M_{init}	maximum perturbation used to form initial population
M_{sub}	maximum amount by which any subsequent individuals can deviate from the initial guess
S	method used for selecting parents (roulette-wheel (‘r’) or tournament (‘t’))
t_{size}	number of individuals to take part in a tournament
t_{prob}	probability of selecting the best individual from the current tournament
E	method of encoding (multiplicative (‘m’) or interval (‘i’))
B	binary representation (standard (‘b’) or gray (‘g’))
R	method used to choose individuals for the next generation (all offspring (‘o’) or combination of offspring and parents (‘a’))
λ_{tol}	how negative an eigenvalue must be before it is considered negative

rithms was used:

$$f(x, y) = 3(1-x)^2 \mathrm{e}^{-x^2-(y+1)^2} - 10\left(\frac{x}{2} - x^3 - y^4\right)\mathrm{e}^{-x^2-y^2} - \frac{\mathrm{e}^{-(x+1)^2-y^2}}{3}, \qquad (3)$$

where the encoded variables are the independent variables x and y of the function. This function contains two minima, three maxima, and three saddle points, as shown in the contour plot in figure 1. The precise locations and characteristics of these stationary points are given in table 2.

By systematically testing different values of the algorithm parameters and averaging the results from 25 runs of the code, the following parameter values were determined to give the best results: $\mu = 100$, $G_{\max} = 100$, $p_c = 0.75$, $p_m = 0.03$, $S =$ 't', $t_{\text{size}} = 6$, $t_{\text{prob}} = 0.75$, $E =$ 'r', $B =$ 'g', $R =$ 'a', $\lambda_{\text{tol}} = -5.0$. Note that since range encoding was determined to be the best, the parameters M_{init} and M_{sub} were not required since variable values are restricted to the specified intervals by the design of interval encoding. Also, the use of interval encoding encourages choosing n_b to be large, since a larger value results in greater precision. For the current implementation the maximum allowed value of $n_b = 31$ was chosen. The value λ_{tol} was chosen to concentrate the search within the interesting region of the surface.

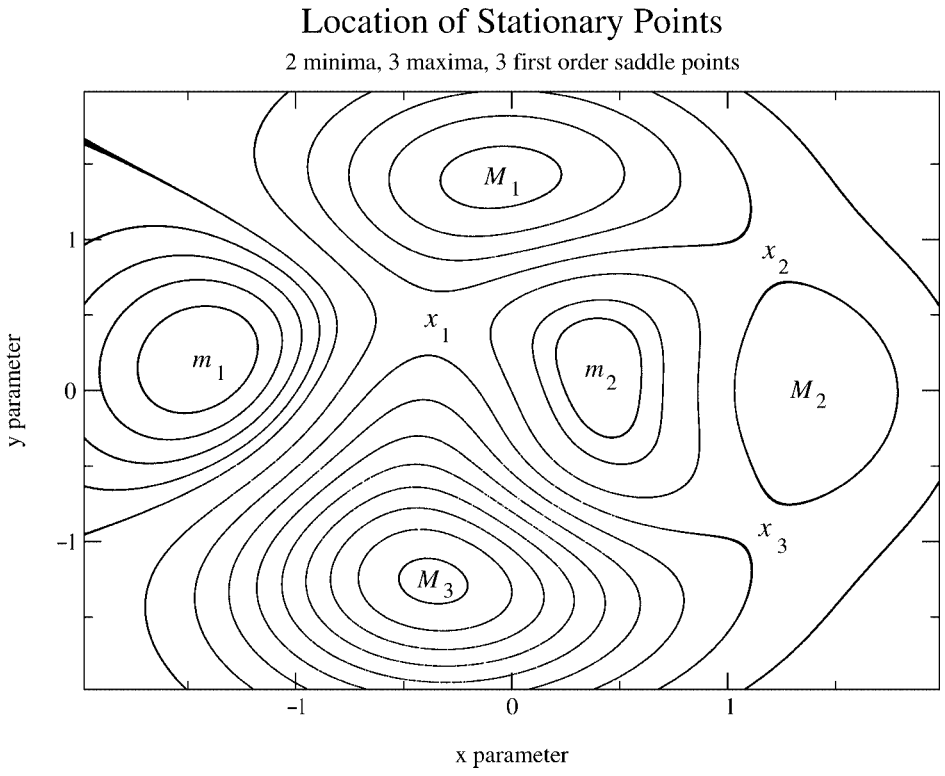


Figure 1. Contour plot of equation (3) showing minima (m), maxima (M), and saddle points (x) (see table 2).

Table 2
Location and characteristics of stationary points for the test function (3). (x, y) is the coordinate location, $f(x, y)$ is the function value, $\|\vec{g}\|$ is the gradient length, λ_1 and λ_2 are the eigenvalues of the Hessian matrix. Note that $\|\vec{g}\|$ would be exactly 0 at the stationary points if calculated analytically.

Label	x	y	$f(x, y)$	$\|\vec{g}\|$	λ_1	λ_2
m_1	−1.431359	0.206945	−2.467838	3.246533×10^{-6}	7.4497	14.3481
m_2	0.404936	0.166523	−0.930778	1.703883×10^{-6}	5.4577	23.8348
M_1	−0.059953	1.409113	5.425638	3.490810×10^{-6}	−21.6508	−11.1346
M_2	1.370701	−0.008093	2.909429	2.018528×10^{-6}	−14.7846	−5.7225
M_3	−0.365185	−1.263316	9.276397	4.078678×10^{-6}	−28.1092	−20.0956
x_1	−0.364835	0.455781	1.665537	6.698525×10^{-6}	−20.1040	15.5895
x_2	1.148302	0.868567	1.897576	3.462192×10^{-6}	−9.9574	6.7342
x_3	1.154115	−0.890394	1.915014	1.581266×10^{-6}	−9.2596	6.5939

While these parameter values proved best for the test problem (3), one must note that the values of many of the parameters are strongly dependent on the problem being solved. For example, in the case of chemical structures a mutation rate $p_{\mathrm{m}} = 0.05$, and $\lambda_{\mathrm{tol}} = 0.0$ was found to be better.

A scatter plot, demonstrating the evolution of the algorithm is shown in figure 2, with each point representing an individual.

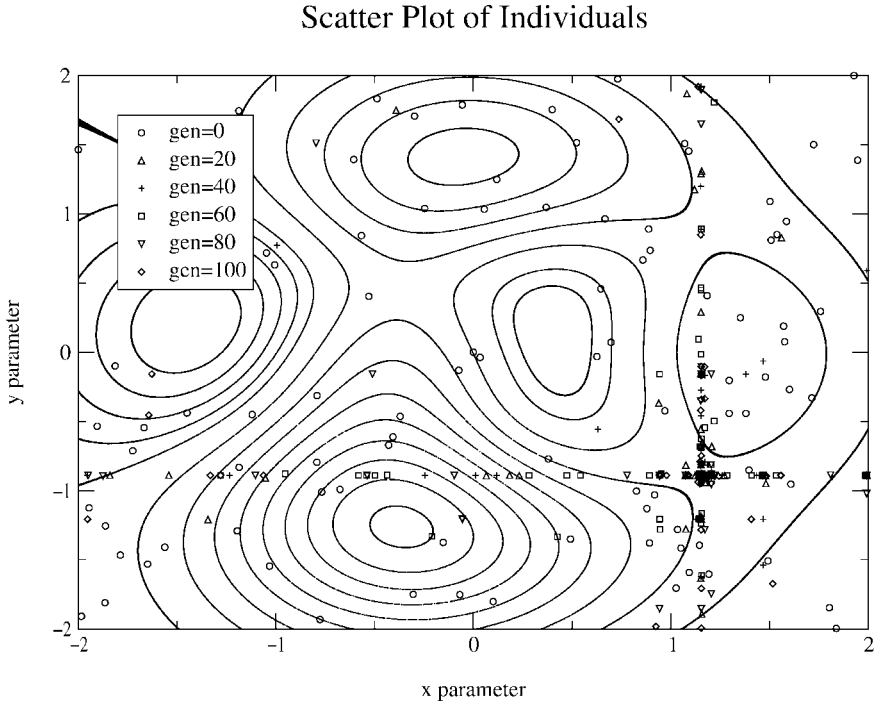


Figure 2. Scatter plot of individuals every 20 generations. Here, saddle point x_3 was reported as the optimum.

By modifying the fitness function to seek points with zero negative eigenvalues (replacing $(n-1)$ by n in equation (3)), the minima of the test problem were also found.

6. Chemical structure results

Four chemical structures were optimized with the genetic algorithm. These structures, along with their starting geometries were taken from Baker and Chan [15], and are given in table 3. The energies, first derivatives, and numerical second derivatives were

Table 3
Test cases used for transition state structure optimization (bond lengths given in angstroms and bond angles in degrees). The starting geometries used for the optimization are as shown in the form of a Z-matrix, and are the same as those given in [15].

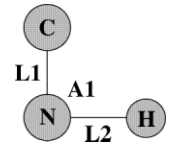
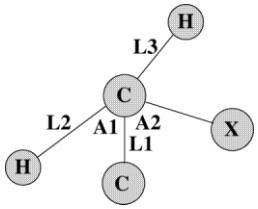
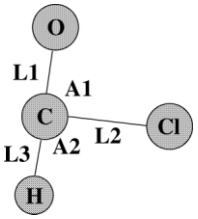
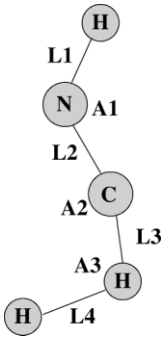
1.	HCN ↔ HNC			
	C1	L1	1.14838	
	N2 C1 L1	L2	1.58536	
	H3 C2 L2 C1 A1	A1	90.0	
2.	HCCH ↔ CCH_2			
	C1	L1	1.24054	
	C2 C1 L1	L2	1.65694	
	X3 C1 1.0 C2 90.0	L3	1.06318	
	H4 C1 L2 C2 A1 X3 180.0	A1	60.3568	
	H5 C1 L3 X3 A2 C2 180.0	A2	60.3568	
3.	HOCl ↔ HCl + CO			
	O1	L1	1.17	
	C2 O1 L1	L2	2.335	
	Cl3 C2 L2 O1 A1	L3	1.127	
	H4 C2 L3 Cl3 A2 O1 180.0	A1	90.0	
		A2	90.0	
4.	HNC + H_2 ↔ H_2CNH			
	H1	L1	1.0	
	N2 H1 L1	L2	1.2	
	C3 N2 L2 H1 A1	L3	1.0	
	H4 C3 L3 N2 A2 H1 D1	L4	1.2	
	H5 H4 L4 C3 A3 N2 D2	A1	120.0	
		A2	150.0	
		A3	90.0	
		D1	170.0	
		D2	10.0	

Table 4
Results obtained for the HCN ↔ HNC rearrangement showing the initial geometry and the optimized geometries from the GA and VA methods. The Hessian has one negative eigenvalue (n) for all three structures, but the genetic algorithm structure has a slightly lower gradient length ($\|\vec{g}\|$).

Variable	Initial	GA	VA
L1	1.14838	1.18265	1.18269
L2	1.58536	1.40780	1.40741
A1	90.0	55.0267	55.0541
E_t	−92.20273	−92.24604	−92.24604
$\|\vec{g}\|$	7.53×10^{-2}	5.72×10^{-5}	7.30×10^{-5}
n	1	1	1

Table 5
Results obtained for the HCCH ↔ CCH_2 rearrangement showing the initial geometry and the optimized geometries from the GA and VA methods. The Hessian has one negative eigenvalue (n) for all three structures, but the genetic algorithm structure has a slightly lower gradient length ($\|\vec{g}\|$).

Variable	Initial	GA	VA
L1	1.24054	1.24658	1.24645
L2	1.65694	1.42802	1.42920
L3	1.06318	1.05552	1.05565
A1	60.3568	54.1655	54.1117
A2	60.3568	86.6367	86.6471
E_t	−76.265417	−76.29343	−76.29343
$\|\vec{g}\|$	2.68×10^{-1}	1.50×10^{-4}	2.64×10^{-4}
n	1	1	1

calculated using an *ab initio* approach at the Hartree–Fock level, with the 3-21G basis set.

These structures were optimized with a minimization of sum of squares method (VA) [16,17], and the results are compared to those obtained from optimizing with the genetic algorithm. Reaction 1 (see table 3) is an HCN ↔ HNC rearrangement and the results obtained are shown in table 4. The bond lengths are reported in angstroms and the bond angles in degrees. The optimized structures obtained from both methods are very similar and both have a Hessian matrix with one negative eigenvalue and a gradient length on the order of 10^{-5}. The total energy E_t is reported in hartrees, and the initial and final values match those reported in [15].

Results obtained for reaction 2 are shown in table 5. The two optimized geometries are again very similar, both having gradient lengths on the order of 10^{-4}. Again, the energies match those reported by Baker and Chan.

Results obtained for reaction 3 are shown in table 6. There are only slight differences between the two optimized geometries and both converged with gradient lengths

Table 6
Results obtained for the HOCl ↔ HCl + CO reaction showing the initial geometry and the optimized geometries from GA and VA. The two optimized geometries are similar with comparable gradient lengths, with the VA gradient length slightly lower.

Variable	Initial	GA	VA
L1	1.17	1.11608	1.11607
L2	2.335	2.55176	2.55180
L3	1.127	1.10407	1.10418
A1	90.0	126.6372	126.6381
A2	90.0	47.6255	47.5872
E_t	−569.87865	−569.89752	−569.89752
$\|\vec{g}\|$	4.58×10^{-1}	4.73×10^{-5}	6.67×10^{-5}
n	1	1	1

Table 7
Results obtained for the HNC + H_2 ↔ H_2CNH reaction showing the initial geometry and the optimized geometries from GA and VA.

Variable	Initial	GA	VA
L1	1.0	1.01305	1.01184
L2	1.2	1.21603	1.21292
L3	1.0	1.11342	1.11201
L4	1.2	1.14220	1.15952
A1	120.0	117.6791	118.7790
A2	150.0	152.2407	152.7408
A3	90.0	93.1678	93.9809
D1	170.0	181.1704	180.0548
D2	10.0	−1.2450	−0.0829
E	−93.30097	−93.31110	−93.31114
$\|\vec{g}\|$	2.97×10^{-1}	5.21×10^{-4}	2.55×10^{-4}
n	2	1	1

on the order of 10^{-5}. All three geometries have a single negative eigenvalue, and the energies listed agree with those in Baker and Chan.

Results obtained for reaction 4 are shown in table 7. Again, both optimized geometries are similar. However, note that the initial geometry has two negative eigenvalues but both optimized geometries have just one. The energy of the optimized structures differ in the fifth decimal place, with the VA energy matching that reported by Baker and Chan.

7. Advantages and disadvantages of the algorithm

Unlike other transition state optimization techniques, the genetic algorithm approach is not sensitive to the structure of the initial Hessian. The algorithm promotes the

Table 8
CPU time (in hours, minutes and seconds) required to optimize the chemical structures presented on a 600 MHz Pentium III. These run times do not compare to the time required to optimize the same structures with the VA method, for which the run times were less than 5 min.

Reaction	Number of variables	Number of generations	Time (hh.mm.ss)
HCN ↔ HNC	3	100	19.04.05
HCCH ↔ CCH_2	5	100	30.57.33
HOCl ↔ HCl + CO	5	73	88.55.04
HNC + H_2 ↔ H_2CNH	9	100	75.09.45

production of individuals with one negative eigenvalue, favouring the correct Hessian eigenvalue structure.

Furthermore, genetic algorithms are known for their ability to efficiently sample a search space to locate a global optimum. Although this is not the intention in the current implementation it is worth noting that transition state structure optimization is less of a local search than the optimization of minima; for transition states, it is very unlikely that an initial guess can be made as close to the desired optimum as is possible for minima.

The requirement of calculating both first and second derivatives for each individual in the population every generation leads to very long run times, especially in comparison to traditional methods. Examples of the time required to run the algorithm for the chemical structures given are shown in table 8. Despite the long run times required, it has been shown that the genetic algorithm can be designed to find first order (or any other order) saddle points. Future work on the algorithm will help to decrease this run time. In particular, the elimination of a large number of derivatives is an important consideration. In addition, parallelization of the algorithm via distribution of the fitness evaluation of the individuals can be easily done. This would significantly decrease the run time since the fitness evaluation is the most computationally intensive section of the code when optimizing chemical structures.

Finally, the genetic algorithm was implemented with the idea of optimizing transition state structures that proved difficult (or impossible) to optimize with traditional methods, as well as to allow the flexibility of providing an initial guess far removed from the saddle point while still achieving convergence.

8. Conclusions

Some important aspects of implementing a genetic algorithm were discussed, these included the following points. Genetic algorithms manipulate a collection of potential solutions to a problem in parallel, rather than successively improving a single estimate of the optimum as is done in traditional methods. The algorithm works with the encoded form of these potential solutions rather than the solution values themselves, and operates on these encoded values with stochastic operators. Implementation of a genetic algorithm is problem dependent and each piece of software is sufficiently detailed to

restrict it to solving only the type of problems for which it was written. Such algorithms which have been highly adapted for a specific problem, such as that discussed here, are often more efficient at solving that problem, at the expense of generality.

The work presented here has laid the foundation for ongoing research in the area of chemical structure optimization using genetic algorithms. Optimization of the sample problem with varying parameters demonstrated the behaviour and viability of the method, providing a good testing medium, as well as a basis for optimizing chemical structures. The code written was able to find all three saddle points of the function, which illustrates the effectiveness of the fitness function used. Furthermore, the two minima of the sample problem were also found, which demonstrates the robustness of the genetic algorithm technique.

Applying the implementation to the optimization of chemical structures proved that it was able to efficiently sample the regions given and effectively find a transition state structure. Transition state structures for several chemical reactions were determined, in agreement with results from other optimization techniques.

Acknowledgements

Many thanks are expressed to the Natural Sciences and Engineering Research Council (NSERC) and Memorial University of Newfoundland for financial support. Also, computational facilities were provided by the Departments of Mathematics, and Computing and Communications at Memorial, for which we are thankful.

References

[1] C. Broyden, The convergence of a class of double rank minimization algorithms, Parts I and II, J. Inst. Math. Appl. 6 (1970) 76–90 and 222–231.
[2] R. Fletcher, A new approach to variable metric algorithms, Comp. J. 13 (1970) 317–322.
[3] D. Goldfarb, A family of variable metric methods derived by variational means, Math. Comp. 24 (1970) 23–26.
[4] D. Shanno, Conditioning of quasi-Newton methods for function minimization, Math. Comp. 24 (1970) 647–656.
[5] W. Davidon, Optimally conditioned optimization algorithms without line searches, Math. Prog. 9 (1975) 1–30.
[6] J.M. Anglada and J.M. Bofill, How good is a Broyden–Fletcher–Goldfarb–Shanno-like update Hessian formula to locate transition structures? Specific reformulation of Broyden–Fletcher–Goldfarb–Shanno for optimizing saddle points, J. Comput. Chem. 19 (1998) 349–362.
[7] J. Holland, *Adaptation in Natural and Artificial systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence* (University of Michigan Press, Ann Arbor, MI, 1975).
[8] S.D. Bungay, Optimization of transition state structures using genetic algorithms, Master's thesis, Memorial University of Newfoundland, St. John's, NF, Canada (2000).
[9] J. Mestres and G.E. Scuseria, Genetic algorithms: A robust scheme for geometry optimizations and global minimum structure problems, J. Comput. Chem. 16 (1995) 729–742.
[10] A.Y. Jin, F.Y. Leung and D.F. Weaver, Development of a novel genetic algorithm search method (GAP1.0) for exploring peptide conformational space, J. Comput. Chem. 18 (1997) 1971–1984.

[11] A.Y. Jin, F.Y. Leung and D.F. Weaver, Three variations of genetic algorithm for searching biomolecular conformation space: Comparison of GAP 1.0, 2.0, and 3.0, J. Comput. Chem. 20 (1999) 1329–1342.
[12] S.M. Le Grand and K.M. Merz Jr., The application of the genetic algorithm to the minimization of potential energy functions, J. Global Opt. 3 (1993) 49–66.
[13] R. Judson, Genetic algorithms and their use in chemistry, Reviews in Computational Chemistry 10 (1997) 1–73.
[14] E.K.P. Chong and S.H. Zak, *An Introduction to Optimization* (Wiley, New York, 1996).
[15] J. Baker and F. Chan, The location of transition states: A comparison of Cartesian, Z-matrix, and natural internal coordinates, J. Comput. Chem. 17 (1996) 888–904.
[16] M.J.D. Powel, A method for minimizing a sum of squares of non-linear functions without calculating derivatives, Comp. J. 7 (1965) 303–307.
[17] M.J.D. Powell, Subroutine VA05AD, AERE Subroutine Library, Harwell, Didcot, Berkshire, UK.