8-1-2005

A Novel Evolutionary Global Optimization Algorithm and its Application in Bioinformatics

Hualin Wang

Okan K. Ersoy

Follow this and additional works at: http://docs.lib.purdue.edu/ecetr

http://docs.lib.purdue.edu/ecetr/65

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
A NOVEL EVOLUTIONARY GLOBAL OPTIMIZATION ALGORITHM AND ITS APPLICATION IN BIOINFORMATICS

HUALIN WANG
OKAY K. ERSOY

TR-ECE – 05-11
AUGUST 2005

PURDUE UNIVERSITY
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, IN 47907-2035
A Novel Evolutionary Global Optimization Algorithm and its

Application in Bioinformatics

Hualin Wang and Okan K. Ersoy
School of Electrical and Computer Engineering
465 Northwestern Ave.
Purdue University
West Lafayette, IN 47907
{hualin, ersoy}@purdue.edu
Table of Contents

List of Figures .................................................................................................................. 3
List of Tables .................................................................................................................... 3
Abstract ............................................................................................................................ 4
1 Introduction .................................................................................................................... 5
2 EM Algorithm to Model How GCO Searches Function Space ..................................... 6
   2.1 Expectation Maximization (EM) Algorithm ............................................................ 7
   2.2 Modeling the Data sets with the EM algorithm ...................................................... 10
3 Mixture Gaussian Optimization (MGO) Algorithm ......................................................... 10
4 Experimental Results ................................................................................................... 13
5 Convergence of the MGO algorithm .............................................................................. 15
6 Application in Molecular Conformation Search ............................................................ 19
   6.2 Implementation Details ......................................................................................... 21
   6.3 Test problems ......................................................................................................... 22
   6.4 Experimental Results ........................................................................................... 26
7 Summary and Future Research ...................................................................................... 28
References: ....................................................................................................................... 30
Appendix ............................................................................................................................ 32
List of Figures

FIGURE 1: HISTOGRAM GENERATED FROM INITIAL POINT 0.6946 ...........................................7
FIGURE 2: HISTOGRAM GENERATED FROM INITIAL POINT 0.5226 ........................................8
FIGURE 3: HISTOGRAM GENERATED FROM INITIAL POINT 0.4449 ........................................8
FIGURE 4: THE HISTOGRAM AND THE MIXTURE GAUSSIANS (INITIAL POINT 0.6946) ..........11
FIGURE 5: THE HISTOGRAM AND THE MIXTURE GAUSSIANS (INITIAL POINT 0.5226) ..........11
FIGURE 6: THE HISTOGRAM AND THE MIXTURE GAUSSIANS (INITIAL POINT 0.4449) ..........12
FIGURE 7: EXAMPLE OF A TRANSITION MATRIX AMONG 3 STATES ....................................16
FIGURE 8: GLOBAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)2CH3 .......................22
FIGURE 9: LOCAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)2CH3 .......................23
FIGURE 10: GLOBAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)3CH3 ....................23
FIGURE 11: LOCAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)3CH3 .....................24
FIGURE 12: LOCAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)3CH3 .....................24
FIGURE 13: LOCAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)3CH3 ....................25
FIGURE 14: GLOBAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)4CH3 ....................25
FIGURE 15: GLOBAL MINIMAL ENERGY CONFIGURATION OF CH3(CH2)4CH3 ....................26

List of Tables

TABLE 1: THE TEST FUNCTIONS & RESULTS (n = 30) .................................................................14
TABLE 2: CONFORMATION SEARCH RESULTS .........................................................................28
Abstract

Gray Code Optimization (GCO) algorithm is a deterministic algorithm based on the Gray code, binary numbers representation. It sometimes suffers from slow convergence and sub-optimal solutions. Expectation Maximization (EM) algorithm is used to analyze how the GCO explores the search space. The investigation of how the GCO generates a population indicates that it is similar to generating samples with a mixture Gaussian distribution. The EM algorithm extracts a three components mixture Gaussian model. Based on these findings, a novel stochastic optimization algorithm based on the mixture Gaussian model is proposed. The new Mixture Gaussian Optimization (MGO) algorithm is not only a continuous stochastic algorithm, but also provides a rigorous mathematic model for answering some theoretical questions. A proof of the convergence of MGO based on the Markov Model is given.

The MGO algorithm is applied to the global optimization problems in bioinformatics. For example, the conformations available to a molecule can have a dramatic effect on its activity. Obtaining global minimum energy conformations of molecule is a very hard optimization problem. The difficulty arises from the following two factors: the conformational space of a reasonable size molecular is very large, and there are many local minima that are hard to sample efficiently. The energy landscape in the conformational space is very rugged, and there are many large barriers between local minima. In this report, the MGO algorithm is used to search the conformation space and locate the global minimal energy structure.
1 Introduction

Gray Code Optimization [1, 2] (GCO) algorithm is a deterministic algorithm based on the Gray code, binary numbers representation. It sometimes suffers from slow convergence and sub-optimal solutions. Expectation Maximization (EM) algorithm is used to analyze how the GCO explores the search space. The investigation of how the GCO generates a population indicates that it is similar to generating samples with a mixture Gaussian distribution. The EM algorithm extracts a three components mixture Gaussian model. Based on these findings, a novel stochastic optimization algorithm based on the mixture Gaussian model is proposed. The new Mixture Gaussian Optimization (MGO) algorithm is not only a continuous stochastic algorithm, but also provides a rigorous mathematic model for answering some theoretical questions. A proof of the convergence of MGO based on the Markov Model is given.

The MGO algorithm is applied to the global optimization problems in bioinformatics. For example, the conformations available to a molecule can have a dramatic effect on its activity. Obtaining global minimum energy conformations of molecule is a very hard optimization problem. The difficulty arises from the following two factors: the conformational space of a reasonable size molecule is very large, and there are many local minima that are hard to sample efficiently. The energy landscape in the conformational space is very rugged, and there are many large barriers between local minima. Among many optimization methods in conformation search, the traditional gradient based algorithm, the random search algorithm, and the Monte Carlo algorithm
are the most popular. In this report, the MGO algorithm is used to search the conformation space and to locate the global minimal energy structure. The algorithm is implemented in an embedded language SVL (Scientific Vector Language), which is only available in a commercial software package MOE [3] (Molecular Operating Environment, from Chemical Computing Group, Inc). The results are compared with two popular methods used by MOE.

2 EM Algorithm to Model How GCO Searches Function Space

The GCO algorithm uses the Gray code binary number representation and bit flipping to search through the space. By flipping large portions of the bits, it generates sample points which are far away from the initial point. By flipping small portions of the bits, it generates points which are closer to the initial point. However, it is very difficult to exactly describe how it covers the search space. To further understand the GCO algorithm, a single variable problem is studied here. From an initial point, a population is generated following the GCO algorithm. Then a histogram is generated to describe how the children points cover the search space. For simplicity, we assume the search range to be [0, 1]. Figure 1 through Figure 3 show 3 different data sets\(^1\) and their histograms.

\(^1\) Data are given in Appendix
2.1 Expectation Maximization (EM) Algorithm

The EM algorithm [4, 5] is an efficient iterative procedure to compute the Maximum Likelihood estimate in the presence of missing or hidden data. It is often used to approximate a probability density function (p.d.f).

Each iteration of the EM algorithm consists of two processes: The E-step, and the M-step. In the expectation, or E-step, the missing data are estimated given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation. In the M-step, the likelihood function is maximized under the assumption that the missing data are known.

Figure 1: Histogram generated from initial point 0.6946.
Figure 2: Histogram generated from initial point 0.5226.

Figure 3: Histogram generated from initial point 0.4449.
Let the observed variable vector to be known as $Y$ and the latent variable vector as $Z$. Together, $Y$ and $Z$ form the complete data set. Assume $p$ is the joint conditional distribution of the complete data set with parameters $\theta : p(y, z | \theta)$. An EM algorithm will then iteratively improve an initial estimate $\theta_0$ and construct new estimates $\theta_1, \theta_2, \ldots, \theta_n$.

We define the expectation $Q(\theta | \theta')$ as:

$$Q(\theta | \theta') = E[\log p(z, y | \theta) | \theta', y]$$

$$= \sum_z p(z, y | \theta') \log p(y, z | \theta)$$

Estimate $\theta_{n+1}$ from $\theta_n$ can be expressed as

$$\theta_{n+1} = \arg\max \sum_z p(z, y | \theta^n) \log p(y, z | \theta)$$

The sketch of the EM algorithm is as follows:

1. $i = 0$, randomly initialize $\theta_0$
2. Compute $Q(\theta | \theta_i)$
3. Choose $\theta_{i+1}$ to maximize $Q(\theta | \theta_i)$
4. if $\theta_{i+1}$ and $\theta_{i+1}$ are not close enough, $i = i + 1$, go to step 2, else stop.

It is shown that the EM iterations do not decrease the observed data likelihood function. In practice, this means that an EM algorithm will converge to a local maximum of the observed data likelihood function [4].
2.2 Modeling the Data sets with the EM algorithm

A mixture Gaussian model can approximate any continuous probability density function. In this research, a three-component Gaussian model is used to approximate the data sets generated from GCO. Figure 4 through Figure 6 are the results from the EM algorithm corresponding to the data set 1, 2, and 3 given in the Appendix. Even though it looks like there are two Gaussians, in reality, there are three Gaussians, two of them have very close mean values, but one with very small variance.

3 Mixture Gaussian Optimization (MGO) Algorithm

Inspired by the mixture Gaussian model extracted from the GCO algorithm, a new continuous stochastic optimization algorithm MGO is proposed. Although the two algorithms share some similar ties, they are essentially different. The MGO algorithm operates on continuous space, and the representation accuracy is determined by the computer’s machine resolution, not by the bit length anymore. Unlike GCO which is a deterministic algorithm, the MGO is a stochastic algorithm. Another difference is that for the mutation operation, the GCO uses bit flipping; the MGO uses mixture Gaussian noise.
Figure 4: The histogram and the mixture Gaussians (initial point 0.6946).

Figure 5: The histogram and the mixture Gaussians (initial point 0.5226).
The biggest difference between the MGO algorithm and other continuous evolutionary algorithms is that for MGO, it uses a mixture of Gaussians; all others use a single Gaussian. The advantage of using a mixture of Gaussians is that it covers a broad range of the search space, and results in better global exploration. In each generation, even towards the end of the search, it keeps global exploration intact. Intuitively, it is easier to escape the local minima and prevent the pre-mature convergence.

The outline of the MGO algorithm is as follows:

1. Initialize the parent randomly
2. Generate a population by choosing each variable from a predetermined mixture Gaussian distribution.
3. Evaluate the population, find the best child.
4. If stop criteria is false, go to step 2, else stop.
The parameters of the mixture Gaussian distribution are very important. They are chosen experimentally based on the results with the GCO algorithm in the previous section. The first standard deviation $\sigma$ for the major Gaussian is chosen to be very large, and the standard deviations for the other two Gaussians are chosen to be smaller. Intuitively, big standard deviation favors local search, small standard deviation favors global search. The three means ($\mu$) are chosen to cover the whole search range as completely as possible. The coefficient of each Gaussian component is chosen empirically based on the results of the previous section.

4 Experimental Results

The experiments were conducted with a series of high dimensional problems. Each function has many local minima. For comparison purpose, two very popular Evolutionary Algorithm tool boxes were used in addition. GAToolbox 1 is from Matlab [6], GAToolbox 2 is from University of Sheffield, UK [7]. For fairness, each algorithm is given roughly the same resources. Here, it is the number of function evaluations of the objective function. The results reported are the average of 20 independent runs. They are reported in Table 1.
Table 1: The test functions & results ($n = 30$).

<table>
<thead>
<tr>
<th>Test Function</th>
<th>Variable Range</th>
<th>Average function minimum found by each algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>MGO</td>
</tr>
<tr>
<td>$f_1 = - \sum_{i=1}^{n} x_i \sin(\sqrt{x_i})$</td>
<td>$(-500,500)^n$</td>
<td>$-1.26e+4$</td>
</tr>
<tr>
<td>$f_2 = \sum_{i=1}^{n} [x_i^2 - 10 \cos(2\pi x_i) + 10]$</td>
<td>$(-5.12,5.12)^n$</td>
<td>$3.2e-7$</td>
</tr>
<tr>
<td>$f_3 = \frac{1}{4000}$ $\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$</td>
<td>$(-600,600)^n$</td>
<td>$5.1e-8$</td>
</tr>
<tr>
<td>$f_4 = \frac{\pi}{n} \left{ 10 \sin^2(\pi y_i) + \sum_{i=1}^{n-1} (y_i - 1)^2[1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2 \right}$ $+ \sum_{i=1}^{n} u(x_i,10,100,4)$</td>
<td>$(-50,50)^n$</td>
<td>$1.3e-8$</td>
</tr>
<tr>
<td>$y_i = 1 + \frac{1}{4}(x_i + 1)$</td>
<td>$u(x,a,k,m) = \begin{cases} k(x-a)^m, &amp; x &gt; a \ 0, &amp; -a \leq x \leq a \ k(-x-a)^m, &amp; x &lt; a \end{cases}$</td>
<td></td>
</tr>
<tr>
<td>$f_5 = 0.1 { \sin^2(3\pi x_i) + \sum_{i=1}^{n-1} (x_i - 1)^2[1 + \sin^2(3\pi y_{i+1})] + (x_n - 1)^2[1 + \sin^2(2\pi y_n)] } + \sum_{i=1}^{n} u(x_i,5,100,4)$</td>
<td>$(-50,50)^n$</td>
<td>$4.2e-8$</td>
</tr>
</tbody>
</table>
Compared to the GCO algorithm, the MGO algorithm has no problem to optimize the function \( f_2 \). On the contrary, GCO algorithm does poorly on this function. Another property deserved to mention is the running time. The GCO algorithm is implemented in the C language, where MGO algorithm is implemented in Matlab. The MGO algorithm is on the average 10 times faster than the GCO on these high-dimension problems. The main reason is that in Matlab, the function calls can be vectorized. One function call can evaluate the whole population. On the contrary, in the C language, to evaluate the whole population, a lot of function calls have to be made. When the population size is huge, the advantage of less function call is apparent.

5 Convergence of the MGO algorithm

A process or algorithm that has a random element is called a stochastic process. It can be thought of as a sequence of random events occurring in time:

\[ X_0, X_1, X_2, \ldots \]

Each of the \( \{X_i\} \) is a random variable. Each set of possible values that these variables can assume is called a state of the system. The simplest kind of such a stochastic process is when the distribution of the state at time \( t \) depends only on what happened at time \( t-1 \). If this is the case, it is called a Markov Process. Then the sequence \( X_0, X_1, X_2, \ldots \) forms a Markov Chain.
The search of the MGO algorithm can be formulated as a finite state Markov Chain. First, the parent of the next generation only depends on the current parent. It has nothing to do with the previous parents. So, this can be viewed as a Markov Process. Secondly, since each real number is represented discretely in a computer, it consists of a string of 0 and 1. The search space can be viewed as a finite space, although it is usually very huge.

The Markov Chain is characterized by a state vector $\pi$, which is a row vector describing the probability of being in each state in the initial stage of the algorithm, and a transition matrix $P$, which is the transition probability matrix between states. The probability of being in each state after one transition can be represented by $\pi P$. The probability of being in each state after $n$ transitions is $\pi P^n$. Figure 7 gives an example of 3-state case after one and two transitions.

$$\pi = [0.1 \ 0.5 \ 0.4]$$

$$P = \begin{array}{c|ccc}
 & 1 & 2 & 3 \\
\hline
1 & 0.2 & 0.3 & 0.5 \\
2 & 0.5 & 0.1 & 0.4 \\
3 & 0.7 & 0.1 & 0.2 \\
\end{array}$$

After one iteration

$$\pi P = [0.55 \ 0.12 \ 0.33]$$

After two iterations

$$\pi P^2 = [0.401 \ 0.21 \ 0.389]$$

Figure 7: Example of a transition matrix among 3 states.
A state in this chain which has probability 1 to transfer back to the same state is called an absorbing state. Obviously, the global minimal point is an absorbing state. All the other states are called transient states. As time progresses, the behavior of the non-absorbing states can be described by either 1) transition to an absorbing state with nonzero probability in a single step, or 2) transition to some other transient state. Thus, the transition matrix of the MGO algorithm can be described as follows:

\[
P = \begin{bmatrix} I & 0 \\ R & Q \end{bmatrix}
\]

where \( I \) is a 1x1 identity matrix, which describes the transition of the absorbing state. \( R \) is a \( t \times 1 \) vector, which describes the transitions from non-absorbing states to the absorbing state. \( Q \) is a \( t \times t \) matrix, which describes the transitions between non-absorbing states.

According to Goodman [8], the behavior of such a chain after \( n \) iterations have the following format:

\[
P^n = \begin{bmatrix} I & 0 \\ N_nR & Q^n \end{bmatrix}
\]

where \( N_n = I_t + Q + Q^2 + \ldots + Q^{n-1} \), and \( I_t \) is a \( t \times t \) identity matrix. As \( n \) goes to infinity,

\[
\lim_{n \to \infty} P^n = \begin{bmatrix} I & 0 \\ (I_t - Q)^{-1}R & 0 \end{bmatrix}
\]
where the matrix \((I_t - Q)^{-1}\) is guaranteed to exist.

**Theorem.** Let \(\pi\) be the probability of being in each state in the initial state, \(P\) be the transition matrix between states. For simplicity, assume the object function has only one global minimum, which is the absorbing state \(A\). When the iterations of the MGO algorithm tend to infinity, the algorithm reaches the absorbing state \(A\) with probability one.

**Proof:** Assume the algorithm reaches a state \(s\) after \(n\) iteration.

\[
\text{Pr}obability(s \in A) = \pi^* P^n = \pi^* \begin{bmatrix} I & 0 \\ N_nR & Q^n \end{bmatrix}
\]

When \(n\) goes to infinity,

\[
\text{Pr}obability(s \in A) = \lim_{n \to \infty} \pi^* P^n = \pi^* \begin{bmatrix} I & 0 \\ (I_t - Q)^{-1} R & 0 \end{bmatrix}
\]

\[
= \pi^* \begin{bmatrix} I \\ (I_t - Q)^{-1} R \end{bmatrix}
\]

\[
= \pi^* \begin{bmatrix} I \\ \tilde{I}_{rel} \end{bmatrix}
\]

\[
= 1
\]
Note: Since the sum of each row of matrix \( [R \quad Q] \) is 1, so \( R = [I_{nx1} - Q]^* \). Then

\[
(I_r - Q)^{-1} R = I_{nx1}.
\]

6 Application in Molecular Conformation Search

The conformations available to a molecule can have a dramatic effect on its activity. Obtaining global minimum energy conformations of molecule is a very hard optimization problem. The difficulty arises from the following two factors: the conformational space of a reasonable size molecular is very large, and there are many local minima that are hard to sample efficiently. The energy landscape in the conformational space is very rugged, and there are many large barriers between local minima.

6.1 Molecular Energy Model [3]

A potential energy model, equivalently, a forcefield, assigns a potential energy value to a molecular configuration. Virtually all calculations, from partial charge calculations to mechanic dynamics, or docking simulations, require evaluation of the
potential energy and/or the gradient of the potential energy function. The gradient or potential gives the forces in the system.

The potential energy model is a function comprising a number of terms each of which models a particular interaction, e.g. bond stretch or electrostatics. Model parameters are obtained by fitting to empirical data. Different models result from tuning the terms and parameters of a model to special classes of data, for example, proteins or carbohydrates. Some empirical models are Kollman's All-atom [9] model, the MMFF94 [10] medicinal chemistry forcefield, the Engh-Huber [11] united-atom protein forcefield, and the PEF95SAC Carbohydrate forcefield [12].

The potential energy is a sum of interaction energies:

\[
E = E_{str} + E_{ang} + E_{stb} + E_{tor} + E_{oop} + E_{ele} + E_{vdw} + E_{sol} + E_{con}
\]

Where the subscripts have the following interpretation:

- \textit{str} - bond stretch energies
- \textit{ang} - angle bend energies
- \textit{stb} - stretch-bend cross term energies
- \textit{tor} - dihedral rotation energies
- \textit{oop} - out-of-plane energies
- \textit{ele} - electrostatic interactions
• *vdw* - van der Waals interactions
• *sol* - implicit solvent electrostatic correction
• *con* - constraint and restraint pseudo-energies

### 6.2 Implementation Details

The MGO algorithm is implemented in SVL, an embedded language of the MOE software package. MOE has a local energy minimization method MM which is gradient based steepest decent and Newtown methods. MM operates on the Cartesian coordination of each atom in the molecule. Although the speed of MM is very fast, it has two shortcomings. First, it can only locate the local minima. Second, the number of variables grows very quickly with the number of atoms in the molecule. For example, for a 30 atoms molecule, the number of variables is 60 (3*20).

In conformation search, we do not consider the Cartesian coordinates. We are interested in the flexible bonds, which can be rotated freely. The bond length is kept fixed. In practice, the local optimization function MM is used to optimize the bond length. Concentrating on the flexible bonds greatly decrease the number of variables need to be optimized. For example, for the molecule $CH_3(CH_2)_2CH_3$, there are 14 atoms, but only one rotatable bond. If the Cartesian coordinates were used, there would be 42 variables. In conformation search, there is only 1 variable.
The initial position of the molecule is set randomly by rotating each bond a random angle from the original position. The range of each rotatable bond is from 0 to \(2\pi\).

### 6.3 Test problems

To test the algorithm, a series of organic molecules were generated. The size of the molecules ranges from 14 atoms to 41 atoms. In the experiments, the class of molecules \(CH_3(CH_2)_nCH_3\) was used. There are two reasons for this choice. First of all, it has a lot of local minima. Secondly, the global minimum is known to the scientists.

\(CH_3(CH_2)_2CH_3\) has 2 conformations, Figures 8 and 9 shows the two conformations. Figure 8 is the global minimal configuration; Figure 9 is a local minimal configuration. The energies are -5.031kcal/mol and -4.268 kcal/mol respectively.

![Figure 8: Global minimal energy configuration of \(CH_3(CH_2)_2CH_3\).](image)
Figure 9: **Local minimal energy configuration of** $\text{CH}_3(\text{CH}_2)_2\text{CH}_3$.

$\text{CH}_3(\text{CH}_2)_3\text{CH}_3$ has 4 conformations, Figures 10, 11, 12 and 13 show the four conformations. Figure 10 is the global minimal configuration; Figures 11, 12 and 13 are the local minimal configurations. The energies are $-5.201\text{kcal/mol}$, $-1.545 \text{kcal/mol}$, $-3.832 \text{kcal/mol}$ and $-4.393 \text{kcal/mol}$ respectively.

Figure 10: **Global minimal energy configuration of** $\text{CH}_3(\text{CH}_2)_3\text{CH}_3$. 


Figure 11: Local minimal energy configuration of $CH_3(CH_2)_3CH_3$.

Figure 12: Local minimal energy configuration of $CH_3(CH_2)_3CH_3$. 
Figure 13: Local minimal energy configuration of $CH_3(CH_2)_3CH_3$.

$CH_3(CH_2)_4CH_3$ has more than 10 configurations. Figure 14 is the global minimal energy configuration with energy -5.377 kcal/mol.

Figure 14: Global minimal energy configuration of $CH_3(CH_2)_4CH_3$.

$CH_3(CH_2)_11CH_3$ has more than 500 configurations. Figure 15 is the global minimal energy configuration with energy -6.633 kcal/mol.
Figure 15: Global minimal energy configuration of $CH_3(CH_2)_11CH_3$.

6.4 Experimental Results

In MOE, there are two built in conformation search algorithms. The first one is a systematic search algorithm. Systematic Conformational Search generates molecular conformations by systematically rotating bonds in a molecule. In any given molecule, all bonds, except bonds to terminal atoms, are candidates for rotation. Such bonds are called rotation bonds. For each rotation bond, a possible relative dihedral increment or step are pre-defined by the user. Once the step is determined for each rotation bond, the algorithm generates all combinations of conformations according to the step list. For example, if there are two rotation bonds and the step is 60 degree, then there are $(360/60)*(360/60) = 36$ combinations.

The second algorithm is a Stochastic Conformational Search algorithm. It generates conformations by randomly sampling local minima of the potential energy surface. This method is similar to the RIPS method [13] which generates new molecular
conformations by randomly perturbing the position of each coordinate of each atom in the molecule by some small amount, typically less than 2 angstroms, followed by energy minimization. In conformational search, the algorithm is similar in essence except that it is based on random rotations of bonds instead of the Cartesian coordination.

The MGO algorithm was implemented in SVL and the results were compared with the two algorithms discussed above. Table 2 shows the results. It contains the minimal energy reached and computation time of each algorithm.

The results clearly show that for small molecules, all of the three algorithms can result in satisfactory solutions. With the increasing molecule size, the systematic search algorithm quickly becomes non-applicable. For a moderate size molecule, the stochastic search algorithm still manages to find the good solution but takes considerable amount of time. When the molecule size increases further, the stochastic search algorithm can only find local minima. On the other hand, the MGO algorithm still can locate the global minima with much less computation time.
Table 2: Conformation Search Results.

<table>
<thead>
<tr>
<th>Molecular Formula</th>
<th>Global Minimal Energy kcal/mol</th>
<th>Minimal Energy Found by Each Algorithm (kcal/mol)</th>
<th>Computation Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Systematic Search</td>
<td>Stochastic Search</td>
</tr>
<tr>
<td>( CH_3(CH_2)_2CH_3 )</td>
<td>-5.031</td>
<td>-5.029</td>
<td>-5.031</td>
</tr>
<tr>
<td>( CH_3(CH_2)_3CH_3 )</td>
<td>-5.201</td>
<td>-5.201</td>
<td>-4.393</td>
</tr>
<tr>
<td>( CH_3(CH_2)_4CH_3 )</td>
<td>-5.377</td>
<td>-5.377</td>
<td>-5.377</td>
</tr>
<tr>
<td>( CH_3(CH_2)_5CH_3 )</td>
<td>-5.557</td>
<td>-5.557</td>
<td>-5.557</td>
</tr>
<tr>
<td>( CH_3(CH_2)_6CH_3 )</td>
<td>-5.736</td>
<td>-5.736</td>
<td>-5.736</td>
</tr>
<tr>
<td>( CH_3(CH_2)_7CH_3 )</td>
<td>-5.915</td>
<td>NoA²</td>
<td>-5.915</td>
</tr>
<tr>
<td>( CH_3(CH_2)_8CH_3 )</td>
<td>-6.095</td>
<td>NoA</td>
<td>-5.292</td>
</tr>
<tr>
<td>( CH_3(CH_2)_9CH_3 )</td>
<td>-6.274</td>
<td>NoA</td>
<td>-5.472</td>
</tr>
<tr>
<td>( CH_3(CH_2)_{10}CH_3 )</td>
<td>-6.453</td>
<td>NoA</td>
<td>-5.613</td>
</tr>
<tr>
<td>( CH_3(CH_2)_{11}CH_3 )</td>
<td>-6.633</td>
<td>NoA</td>
<td>NoA</td>
</tr>
</tbody>
</table>

7 Summary and Future Research

In this report, a new algorithm MGO inspired by the GCO algorithm was proposed. A mixture of Gaussians is used to generate the population. With this mathematic model, the convergence of the MGO algorithm is proved by modeling the search process as a Markov chain model.

² NoA means that the time needed to compute the result is not acceptable
The MGO algorithm is applied to an important computational chemistry problem called conformation search. By restricting the optimization to take place on those rotatable bonds, the difficult problem is greatly simplified. The MGO algorithm is implemented with the built-in language SVL in the MOE software package. The experiments on a series of molecules show that the MGO algorithm outperforms the two commercial algorithms built in with the MOE package.

In real life problems, especially in the bioinformatics area, most of the optimization problems are highly non-separable, which means that there are strong correlations between variables. Traditional evolutionary algorithms are not very successful on such problems. The MGO algorithm performs reasonably on some toy non-separable problems. But on some real life problems, the performance is still not satisfactory. An algorithm which is able to explore the correlations between variables is highly desirable. In future research, the goal is to extend the MGO algorithm, so that it can handle some badly scaled or highly non-separable real problems.
References:


Appendix

Data Set 1, initial point is 0.6946

\[ X_1 = \begin{bmatrix} 
0.69457075714473 & 0.10702188036102 & 0.69457681213845 & 0.10701582536730 \\
0.69457073246468 & 0.69457683681850 & 0.69457111477263 & 0.10863325700831 \\
0.69457075481642 & 0.69457347929999 & 0.69457111477263 & 0.69457647919063 \\
0.69457075667907 & 0.69457683681850 & 0.69457111477263 & 0.10863325700831 \\
0.69457075481642 & 0.69457347929999 & 0.69457111477263 & 0.69457647919063 \\
0.69457075481642 & 0.69457347929999 & 0.69457111477263 & 0.69457647919063 \\
0.69457075481642 & 0.69457347929999 & 0.69457111477263 & 0.69457647919063 \\
0.69457075481642 & 0.69457347929999 & 0.69457111477263 & 0.69457647919063 \\
\end{bmatrix} \]

Data Set 2, initial point is 0.5226

\[ X_2 = \begin{bmatrix} 
0.52259895636761 & 0.18511397954661 & 0.52260787610957 & 0.18510505980465 \\
0.52259893541285 & 0.52260787610957 & 0.18510505980465 & 0.18666145628939 \\
0.52259897645949 & 0.52259893401586 & 0.52260787610957 & 0.18510505980465 \\
0.52259874033231 & 0.52259893401586 & 0.52260787610957 & 0.18510505980465 \\
0.52259895590194 & 0.52259893401586 & 0.52260787610957 & 0.18510505980465 \\
0.52259887371583 & 0.52259893401586 & 0.52260787610957 & 0.18510505980465 \\
0.52269473945749 & 0.52259893401586 & 0.52260787610957 & 0.18510505980465 \\
0.53041145636942 & 0.53041145636942 & 0.53041145636942 & 0.53041145636942 \\
0.52259895660044 & 0.52259895660044 & 0.52259895660044 & 0.52259895660044 \\
0.52259895520345 & 0.52259895520345 & 0.52259895520345 & 0.52259895520345 \\
0.52259888148233 & 0.52259888148233 & 0.52259888148233 & 0.52259888148233 \\
0.52259948419468 & 0.52259948419468 & 0.52259948419468 & 0.52259948419468 \\
0.52259756846046 & 0.52259756846046 & 0.52259756846046 & 0.52259756846046 \\
0.52281199892975 & 0.52281199892975 & 0.52281199892975 & 0.52281199892975 \\
0.51646354364149 & 0.51646354364149 & 0.51646354364149 & 0.51646354364149 \\
0.60240104366150 & 0.60240104366150 & 0.60240104366150 & 0.60240104366150 \\
\end{bmatrix} \]
Data Set 3, initial point is 0.4449

\( X_3 = [0.44486831255370, 0.85657175999521, 0.44485911341497, 0.85658095913394, \]
\( 0.44486829253027, 0.44485913343841, 0.44251845903753, 0.85893081265011, \]
\( 0.44486831488201, 0.44486829020196, 0.44486875999521, 0.4485877581054, \]
\( 0.44495986528810, 0.44242690630314, 0.48393081256280, 0.81986831264102, \]
\( 0.44486831301937, 0.44486831441635, 0.44486832000429, 0.44486828275138, \]
\( 0.44486819334442, 0.44486878939086, 0.44486640520507, 0.44486068315917, \]
\( 0.44483779497557, 0.44499038286623, 0.44438003130359, 0.44291518755325, \]
\( 0.45268081255552, 0.47611831256098, 0.31986831255552, 0.94486831267012, \]
\( 0.44486831232087, 0.44486831278654, 0.44486831185521, 0.44486831371786, \]
\( 0.44486831744315, 0.44486832489373, 0.44486830999257, 0.44486828019025, \]
\( 0.44486822058560, 0.44486810137631, 0.44486833979489, 0.44486881663205, \]
\( 0.44486977030637, 0.44486786295773, 0.44486404826047, 0.44485641886593, \]
\( 0.44487167765500, 0.44484116007687, 0.44490219523313, 0.44502426554566, \]
\( 0.44526840617072, 0.44478012492060, 0.44380356242037, 0.44185043741992, \]
\( 0.43794418741901, 0.44575668742083, 0.46138168742447, 0.49263168743174, \]
\( 0.43013168741719, 0.30513168738809, 0.05513168732988, 0.55513168744630] \)