communications in mathematical
match
and in computer chemistry

no. 38, October 1998

ISSN 0340-6253

MATCDY (38) 137-159 (1998)

An Evolutionary Algorithm with Local Search and Classification for Conformational Searching*

Clemens Frey**
Chair II of Mathematics, University of Bayreuth,
95440 Bayreuth

Abstract

In this paper a software package is described that allows using the MM2 force field to compute the energy of three-dimensional conformations of molecules; this energy is minimized, the resulting structures are automatically classified. Thus we get an impression about the set of all low-energy three-dimensional conformations of a given molecule defined in terms of two dimensional connectivity information. The accuracy of the resulting information can be tuned by changing input parameters for the method presented. This is a hybrid which is built from a method for classification of three-dimensional molecules, from a conjugate gradient method for local minimization and from operators stemming from evolutionary algorithms. The latter have proven successful in the solution of difficult optimization tasks (not only) in mathematical chemistry. They are of special importance for the approximate solution of problems where global optima of multimodal functions in high dimensional spaces are sought.

*This work was supported by the DFG under grant Ke 201/16-1.
**e-mail: `clemens.frey@uni-bayreuth.de`

The paper is organized in three sections. The first one gives a formalization of the problem and shows in which way this problem was solved up to now. In the second section the evolutionary algorithm is introduced after a short presentation of a general formalization for this kind of algorithms. Each operator of our method is shown in a separate subsection. A definition of the function which determines the transition from one generation to the next generation concludes this section.
The last section shows trials and corresponding results obtained with the presented method. The summary of trials is followed by listing average energy values of resulting three-dimensional molecule conformations. The results of the classification incorporated are presented, too, as well as time consumption and complexity of the algorithm. A section of conclusions summarizes results and observations.

1 Introduction

The algorithm presented in this article makes fundamental use of the MM2 energy model to get as many conformations of a single (defined topologically in terms of a labelled multigraph, that means two dimensional) molecule as possible. These molecular structures should be stable, i.e. carry low energy in respect to the model. By choosing this approach we are able to get corresponding conformations without incorporating any heuristic information concerning substructures being more likely found than others. The method of Sadowski and Gasteiger is an example for the latter approach, cf. [GCJ90] and [SG93].
Opposite to quantum mechanical methods the MM2 model is easier to manage from the computational point of view. The original energy model and the various refined versions tailored for special types of molecules work by attaching an energy level to each of the possible spatial conformations.
Provided that a molecule is the more stable the lower its energetic level is, we can distinguish structures likely to be found from others hardly found in nature by numerically minimizing the energy function. Because we do not want to collect multiple isomorphic structures which are close together on the energy landscape, an additional classification routine will be employed. This task of minimization and classification is formalized in the following manner.

1.1 Formalization

In order to model the three-dimensional structure of a molecule $\mathcal{M}$ containing m atoms (including H-atoms), a $3m$-vector $x = (x_i)_{i=1\ldots 3m}$ is used which holds the atoms' coordinates. These coordinates are known as *object parameters* of the optimization problem. A function $\mu_{\mathcal{M}}(x)$ combines these parameters with topological information, information about the types of atoms etc. The image of this function is the three-dimensional molecule:

$$\mu_{\mathcal{M}}(x) = \mathcal{M}.$$

The amount of energy held by a conformation defined in terms of a vector x is computed with the help of the MM2 model, which comprises bond lengths, bond angles, torsion angles and van-der-Waals forces within the molecule. This energy of $\mathcal{M}$ resulting from the MM2 model is indicated by

$$E(\mu_{\mathcal{M}}(x)) \in \mathbb{R}^{+}.$$

1.2 Posing the Problem

The task that should be fulfilled by our program is like this: For a given function $\mu_{\mathcal{M}}$ three-dimensional structures x are sought whose energy is as close as possible to the stable minimum of energies

$$\arg\min\{\, E(\mu_{\mathcal{M}}(x)) \mid x \in \mathbb{R}^{3m} \,\}$$

So the energy surface consisting of all possible three-dimensional representations $\mu_{\mathcal{M}}(x)$ is scanned for global minima without using heuristic information being available a-priori. We call this an unconditional search for a *minimum point* x^{*} with respect to a $E \circ \mu_{\mathcal{M}}$. In the context of evolutionary algorithms this function is also called the *fitness function* of the problem.
The following approach for approximately solving this problem has already been tested. It consists of starting a reliable iterative algorithm for local minimization, e.g. the conjugate gradient algorithm, very often at randomly generated points in the domain $\mathbb{R}^{3m}$.

1.3 Previous Approach

This method known as 'random-starts'-method works on a bounded set $G \subset \mathbb{R}^{3m}$, e.g. $G = [-10, 10]^{3m}$; $\mu(G)$ describes the Lebesgue-measure of G:

- Compute $x \in \mathbb{R}^{3m}$ randomly according to a uniform distribution $1/\mu(G)$;
- Start the local minimizer at the point x and get an approximation for a local minimum x' of the objective function, where the following holds:

  $$\exists\, \varepsilon > 0 \;\forall x \in U_{\varepsilon}(x') : E(\mu_{\mathcal{M}}(x')) \leqslant E(\mu_{\mathcal{M}}(x)).$$

  Here $U_{\varepsilon}(x')$ stands for a neighbourhood of x' with radius ε.
- Repeat this process and save the minimum solution that was found during the iterations.

It is supposed that at some time a global minimum is reached within these iterations, if they are only run sufficiently often. Besides the fact that this cannot be known for sure at any time of the algorithm, the absolutely independent restarting in each iteration is an essential drawback of this approach. This means that information the algorithm discovers during one iteration is not used in the following cycles. This information is just forgotten. The algorithms might make the same mistakes all the time, e.g. restarting at a point where a minimum is not supposed to be. This leads to the highly time consuming behaviour of random starts methods.
Although the algorithm that will be presented in the sequel also starts from some absolutely randomly chosen vectors of atom coordinates, it maintains vectors of

successively improving approximate solutions for the problem during the steps of evolution. Those are used to get even better ones in each step of this iterative process.

2 Algorithm and Structure of Solutions

This section is dedicated to the presentation of the method used to evolutionary optimize the energy of molecules in respect to the MM2 force field. The optimization algorithm changes a population of individuals generation by generation; those individuals undergo a selection operator, which implements selection by individual fitness values. The fitness values are essentially determined by the objective function of the optimization problem, so here the fitness values are mainly determined by the MM2 force field.

For sake of presentation we use a formalism that was introduced by Bäck ([Bäc96], S. 63ff.). It puts Evolutionary Strategies, Evolutionary Programming and Genetic Algorithms into one formal frame, and it subsumes all these approaches under the term of 'evolutionary algorithms'. According to Bäck an evolutionary algorithm is defined as an 8-tuple:

$$\mathrm{EA} = (\mathcal{G}, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda).$$

Object parameters mentioned in the previous section specify the coordinates of the atoms belonging to a molecule. Each of these $3m$ object parameters is accompanied with a so-called strategy parameter. Later the values of these strategy parameters will be used as standard deviations for a normal distribution $N(x_i, \sigma_i)$.

Analogously to evolutionary programming the pairs of vectors x of object parameters and σ of strategy parameters form the *individuals* $g = (x, \sigma)$ to be processed during the course of the evolutionary algorithm; every individual is a member of the set $\mathcal{G}$ defined by:

$$\mathcal{G} := \{ g = (x, \sigma) \mid \in \mathbb{R}^{3m}, \sigma \in \mathbb{R}^{3m}_{>0} \}.$$

Note that the parameters are used as continuous variables directly; unlike genetic algorithms this algorithm neither needs nor performs any kind of discretization step. Every iteration of our algorithm ends up with a new vector containing new candidates for an approximate solution of the problem; this vector of length μ is said to be the primary population $P(t)$ at a particular time t ($t \in \mathbb{N} \cup \{0\}$):

$$P(t) = (g^{(t,1)}, \ldots, g^{(t,\mu)}) \in \mathcal{G}^{\mu}.$$

All these populations are considered to be ordered; having $g^{(t,\mu)} = (x^{(t,\mu)}, \sigma^{(t,\mu)})$:

$$\Phi(x^{(t,j)}) \leqslant \Phi(x^{(t,j+1)}) \quad \forall j = 1 \ldots \mu - 1, t \in \mathbb{N} \cup \{0\}, \tag{1}$$

where $\Phi := E \circ \mu_{\mathcal{M}}$ defines the *fitness function* of the evolutionary algorithm.

The *termination criterion* ι steps the evolutionary algorithm if a certain predefined number of generations $P(t)$ has been produced.

Other components of the 8-tuple displayed above, i.e. the selection operator s, the set Ω of genetic operators, the generation transition function Ψ and the size λ of temporary populations, will be explained in the following subsections.

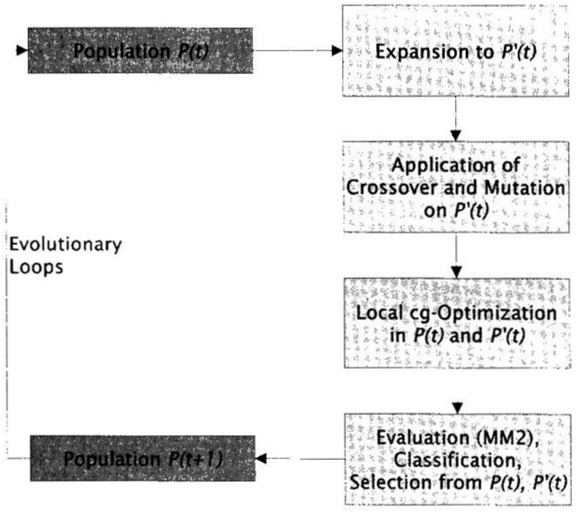
Population P(t)
Expansion to P'(t)
Application of Crossover and Mutation on P'(t)
Evolutionary Loops
Local cg-Optimization in P(t) and P'(t)
Evaluation (MM2), Classification, Selection from P(t), P'(t)
Population P(t+1)

Figure 1: Flow Chart of the Evolutionary Algorithm with Local Search and Classification.

2.1 Initialization

The evolutionary process that was used to iteratively generate populations $P(t)$ is displayed schematically in figure 1. At the beginning of this process a starting population $P(0)$ is needed. It is created by randomly setting every component $x^{(0,j)}$ and $\sigma^{(0,j)}$ of every individual $g^{(0,j)}$ for $j = 0, \dots, \mu$. This uniformly distributed random selection was done with the values chosen according to

$$x_i^{(0,j)} \in [-2, 2]$$
$$\sigma_i^{(0,j)} \in [0.2, 1.2].$$

This has proven useful with respect to the duration of evolution necessary to get good results. In case of setting these parameters disadvantageously, we could observe that more generations were necessary for good results. The built in self-adaptation mechanism of strategy parameters prevents from getting bad results, if only the evolution is allowed to run long enough. So the values given as bounds above are used only for the initialization of the individuals. These values are irrelevant to all operators that will follow.

2.2 Ranking of Individuals in $P(t)$

We have mentioned above that populations are always considered to be ordered. From this arrangement we derive a constant distribution on populations; this distribution will be applied within the expansion operator ω_0 and within the selection of the second operands for crossover. According to Grefenstette and Baker ([Bäc96]) let

$$p_j := \frac{1}{\mu}\left(\eta^+ - (\eta^+ - \eta^-)\frac{j-1}{\mu - 1}\right) \quad \forall j = 1 \dots \mu \tag{2}$$

supposed that $1 \leqslant \eta^+ \leqslant 2$ holds with $\eta^- := 2 - \eta^+$. Because of this restriction a discrete possibility distribution ensues:

$$\sum_{j=1}^{\mu} p_j = \eta^+ - \frac{2(\eta^+ - 1)}{\mu(\mu - 1)} \sum_{j=0}^{\mu-1} j = 1.$$

The value of η^+ controls selection pressure between individuals showing different values of fitness. Typically it is about $\eta^+ = 1.1$; the resulting selection strategy would be close to 'random walk', i.e. an uniformly distributed selection from the individuals of a population.
During the trials we made, setting $\eta^+ = 1.4$ has proven worthwhile; since we prevented similar individuals from showing in the same population more than once by the use of a classification step, we could even choose population sizes quite small. The classification protected against 'premature convergence' of the evolutionary process, this is convergence of the population as a whole against sub-optimal, local minima. Generally, having $\mu' := \lfloor \mu/2 \rfloor$:

$$p_1 = \eta^+ \frac{p_{\mu'} + p_{\mu'+1}}{2}$$

holds, so the possibility for selecting the best individual of a generation is just η^+ as much as the median of the discrete distribution defined above.

2.3 Expansion Operator ω_0

The expansion operator $\omega_0 : \mathcal{G}^\mu \to \mathcal{G}^\lambda$ serves to expand a primary population $P(t)$ containing μ individuals to a temporary population $P'(t)$ with $\lambda \geqslant \mu$ members. For that a sequence $(X_j)_{j=1\ldots\lambda}$ of λ random variables $X_j \in \{1, \ldots, \mu\}$ is evaluated, being distributed according to (2); then $P'(t)$ is generated:

$$P'(t) := (g^{(t,X_j)})_{j=1\ldots\lambda}. \tag{3}$$

Notice that, unlike $P(t)$, $P'(t)$ in (3) is not supposed to be ordered. Let $g'^{(t,j)}$ be defined $g'^{(t,j)} := g^{(t,X_j)}$ in canonical manner.

2.4 Crossover Operator ω_1

Uniform crossover ω_1 is the first really genetic operator that is discussed in this paper. It implements uniform crossover between each of the individuals of $P'(t)$ and a member of $P(t)$ chosen randomly at a time. ω_1 is defined using a given crossover possibility $\vartheta_1 \in [0, 1]$ and an operator $\omega_1 : \mathcal{G}^\mu \times \mathcal{G}^\lambda \to \mathcal{G}^\lambda$ defined below. From now on $(X_j)_{j=1\ldots\lambda}$ are random variables distributed according to (2); let Y and $(Z_i)_{i=1,\ldots,3m}$ be random variables uniformly distributed in $[0, 1]$, which are calculated each time ω_1' is evaluated. We propose $i = 1, \ldots, 3m$ and define:

$$\omega_1'((x', \sigma'), (x'', \sigma'')) := (x, \sigma) \text{ with}$$

$$(x_i, \sigma_i) := \begin{cases} (x_i', \sigma_i') & \text{if } Y < 1/2 \text{ and } Z_i < \vartheta_1 \\ (x_i'', \sigma_i'') & \text{otherwise} \end{cases}$$

This determines in which way ω_1 operates on the above-mentioned population:

$$\omega_1(P(t), P'(t)) := (\omega_1'(g^{(t,X_j)}, g'^{(t,j)}))_{j=1,\ldots,\lambda}.$$

Obviously this crossover operator always works on object parameters and strategic parameters simultaneously. By this a self-adaptation of the strategic parameters shall be achieved. This is based on the following consideration: the better an individual performs with respect to the objective function the better the strategic parameters are suited to the particular environment of the individual; remember that good performance causes a high possibility that the individual will have offspring in later generations, this resulting from the ranking distribution defined above and being ensued from the selection operator discussed in another subsection below. So not only the individuals themselves are improved during the evolution, but also the strategy for creating new individuals.

During the trials that were made within the framework of this project, we used $\vartheta_1 = 0.8$; it controls the frequency for the crossover operator coming into action. Like the ranking parameter η^+ its level was chosen quite high. This produces high rates for the mixture of best individuals from $P(t)$ with best individuals from $P'(t)$. Since the latter quite frequently are copies of the best ones from $P(t)$ because of the expansion operator, this method prefers the production and testing of combinations of coordinate values from the best individuals in $P(t)$.

2.5 Mutation Operator ω_2

Now a mutation operator $\omega_2 : \mathcal{G}^\lambda \to \mathcal{G}^\lambda$ is presented. It is very similar to operators used in the context of evolutionary strategies. Since it is an asexual operator, we can define it componentwise using $\omega_2' : \mathcal{G} \to \mathcal{G}$. Like in subsection 2.4, suppose Y and $(Z_i)_{i=1,\dots,3m}$ are uniformly distributed in $[0,1]$. If $Y \geqslant \vartheta_2$, we set $\omega_2'((x',\sigma')) := (x,\sigma)$, where $(x,\sigma) = (x',\sigma')$; here $\vartheta_2 \in [0,1]$ is a given mutation possibility. Otherwise the object parameters x_i are chosen according to the normal distribution

$$x_i \sim N(x_i', \sigma_i') \quad \forall i = 1 \dots 3m;$$

in this case strategic parameters are determined by

$$\sigma_i := \begin{cases} \alpha\sigma_i' & \text{if } Z_i < 1/2 \\ \beta\sigma_i' & \text{otherwise} \end{cases}$$

if $i = 1, \dots, 3m$.
Mutation of object parameters and strategic parameters at the same time results in self-adaptation of parameters to the optimization problem (cf. [Bäc96]). In our trials the constants were set to $\alpha = 1.3$ and $\beta = 1/\alpha$. As a whole the mutation operator we used can be written as:

$$\omega_2(P'(t)) := (\omega_2'(g^{(t,j)}))_{j=1,\dots,\lambda}.$$

2.6 Local Conjugate Gradient Optimization ω_3

While testing the evolutionary program we realized that we could not get reliable classifications of conformational isomers (cf. subsection 2.7) , if the energy values of individuals to be classified were too far away from the global optimum of the MM2-energy function. So, to get there, a lot of time had to be spent on the evolutionary process which only allowed a low convergence rate. Because of this we have combined the evolutionary approach with a deterministic local minimization method, namely a shortened version of the conjugate gradient search. It works just in between the application of the presented genetic operators on one side and the application of the selection and classification operator on the other side. The line search included in the conjugate gradient method uses the Goldstein-Armijo criterion to guarantee sufficient descent (cf. [Spe93]).
Normally, when using a conjugate gradient method for optimization without our additional modules, the number of iterations equals the problem dimension (here: $3m$); but here this method and the genetic or selection operators are used in turn, so only a fraction of that number of loops is executed. The conjugate gradient iteration is started all over again, after the genetic operators have been in charge.
Observe that the presented method could be interpreted as a conjugate gradient algorithm working in parallel, which tests after some steps if the individuals involved are equivalent. If there are two or more equivalent solutions, only one of them (i.e. the best one) is kept for further enhancement. All the others are thrown away. Then empty locations in the population, which are result of such deletions, are filled with individuals derived from existing individuals by the genetic operators, i.e. mutation

and crossover. After that, this 'parallel evolutionary conjugate gradient method' is started again.
The operator $\omega_3 : \mathcal{G}^\mu \times \mathcal{G}^\lambda \to \mathcal{G}^\mu \times \mathcal{G}^\lambda$ for local search consists of $\vartheta_3 \in \mathbb{N} \cup \{0\}$ steps of the ordinary conjugate gradient method for every individual in $P(t) \cup P'(t)$. The results discussed in section 3 arose from trials with $\vartheta_3 \in \{\frac{3m}{4}, \frac{3m}{2}, 3m\}$. Generally we observed that together with growing values of ϑ_3 the results of classification became sharper, too.

2.7 Selection Operator s

Now an elitist, deterministic $(\mu + \lambda)$-selection $s : \mathcal{G}^\mu \times \mathcal{G}^\lambda \to \mathcal{G}^\mu$ is described which includes information about the individuals' objective function values as well as information about their mutual three-dimensional equivalence. Benecke's [Ben98] classification routines, that serve for the discovery of equivalence, are encapsulated in the function $\kappa : \mathcal{G} \times \mathcal{G} \to \{0, 1\}$; here $\kappa((x', \sigma'), (x'', \sigma'')) = 0$ holds if and only if $\mu_{\mathcal{M}}(x')$ and $\mu_{\mathcal{M}}(x'')$ are equivalent in the sense of Benecke's program detecting full structure isomorphism.
For the sake of formulation of the selection operator, the following recursive definitions are needed; remember that we will write $\Phi(g)$ instead of $\Phi(x)$ for any individual $g = (x, \sigma)$:

$$\begin{aligned} g''^{(t,1)} &:= \arg\min\{\Phi(g) \mid g \in P(t) \cup P'(t)\} \\ \kappa_1(t) &:= \{g \in P(t) \cup P'(t) \mid \kappa(g''^{(t,1)}, g) = 0\} \end{aligned} \tag{4}$$

if $\kappa_j(t) \subsetneq P(t) \cup P'(t)$ $(j \in \mathbb{N})$ we go on with setting recursively:

$$\begin{aligned} g''^{(t,j+1)} &:= \arg\min\{\Phi(g) \mid g \in P(t) \cup P'(t) \setminus \kappa_j(t)\} \\ \kappa_{j+1}(t) &:= \kappa_j(t) \cup \{g \in P(t) \cup P'(t) \mid \kappa(g''^{(t,j+1)}, g) = 0\} \end{aligned} \tag{5}$$

The number of minima, and hence classes we have defined now shall be j_t. If $j_t \geqslant \mu$, the image $s(P(t), P'(t))$ of the selection operator is defined as:

$$s(P(t), P'(t)) := (g''^{(t,1)}, \dots, g''^{(t,\mu)}).$$

Otherwise new individuals are created in accordance with subsection 2.1; the resulting vector of individuals is sorted (2) to get the image $s(P(t), P'(t))$ of the selection operator in this case.
The selection operator that has just been described is elitist (see citeBaeck:1996) since

$$\Phi(g^{(t+1,1)}) \leqslant \Phi(g^{(t,1)})$$

holds for each $t \in \mathbb{N} \cup \{0\}$. It is called deterministic because it is strictly based on classification and comparison of fitness values without any randomness affecting the selection process.

2.8 Population Transition Function Ψ

The population transition function Ψ summarizes all the functions that determine transition from population $P(t)$ to an offspring generation $P'(t)$. Using previous definitions, we are able to write now for $\Psi : \mathcal{G}^{\mu} \to \mathcal{G}^{\mu}$:

$$\Psi(P(t)) := s \circ \omega_3(P(t), \omega_2 \circ \omega_1(P(t), \omega_0(P(t))).$$

Compare this with figure 1 showing a flow chart of the discussed algorithm.
We can define now what 'evolution' or 'sequence of populations' means in the context of the upper definitions. Let $\vartheta_4 \in \mathbb{N}$; then we call a vector $\mathcal{P} = (P(t))_{t=0,\dots,\vartheta_4}$ an *evolution*, if:

$$P(t+1) = \Psi(P(t)), \quad \forall t = 0, \dots, \vartheta_4.$$

So an evolution is just a *sequence of populations* produced by iteratively applying Ψ to a randomly created starting population.
At the beginning of this section a termination criterion ι was discussed. In all of our trials it made the evolutionary process stop after 30 generations had been created by Ψ.

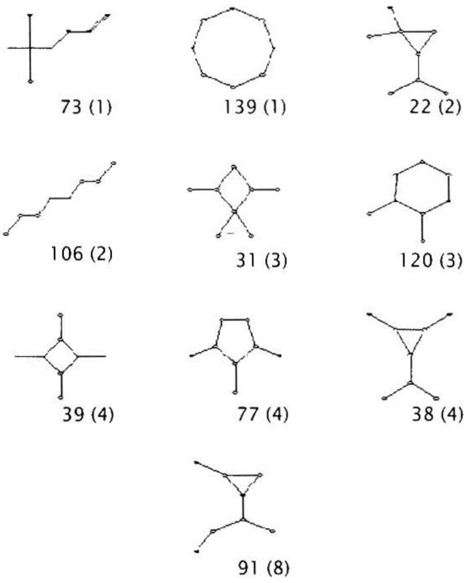

Figure 2: Constitutional isomers of the molecule C_8H_{16} having served as test cases.

3 Trials and Running Times

In the last section a hybrid algorithm employing evolutionary, gradient and classifying methods has been discussed. On 900 trials this algorithm was thoroughly tested and its performance precisely examined. Ten constitutional isomers of the molecule C_8H_{16} built a fundament for these tests. They were chosen so that as many types of substructures (rings, multiple bonds etc.) were represented as possible; in figure 2 those ten isomers are shown. Every molecule in this figure is labelled with two numbers. The first number gives the position of the corresponding isomer on a list produced by the program MOLGEN 3.5. The second number that was put into braces represents the number of stereo isomers corresponding to the constitutional isomer. The isomers are named by $\mathcal{M}_1$ to $\mathcal{M}_{10}$ from upper left to lower right side of the figure.

3.1 Test Problems and Parameters

It has been our purpose to get three-dimensional representations of the molecular isomers of C_8H_{16} show in figure 2. For this we run the algorithm discussed in the second section of this article with various values for parameters ϑ_3 and for population

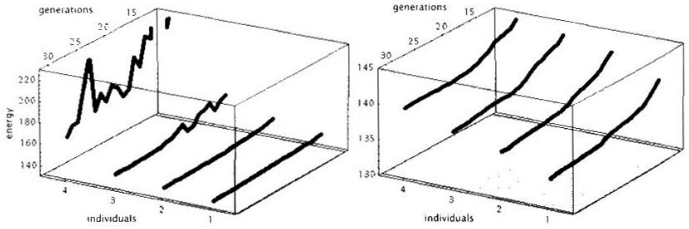

Figure 3: Development of average values taken over 10 population sequences of individuals in $\mathcal{P}(\mathcal{M}_6, 1, 1, .)$ (left hand side) and in $\mathcal{P}(\mathcal{M}_6, 3, 1, .)$ (right hand side).

sizes μ and λ respectively.
The problem size amounts to $3m = 72$ since the molecule under consideration contains 24 atoms (including all the H-atoms). For the number of conjugate gradient steps a value from the triple $(\vartheta_{3,j})_{j=1,\ldots,3} = (18, 36, 72)$ was chosen; population sizes were set to one of the pairs of values in $((\mu_i, \lambda_i))_{i=1,\ldots,3} = ((4,6), (8,10), (16,20))$ during each trial. The population sequences that were produced during the trials are labelled in a canonical manner with $\mathcal{P}(\mathcal{M}_l, i, j, k)$ with indices i and j referring to different population sizes and to different lengths of conjugate gradient iterations respectively.
To catch statistical fluctuations there were ten evolutions consisting of 31 generations each (since $\vartheta_4 = 30$) for every constitutional isomer and each of the 9 possible combinations of values for ϑ_3 and (μ, λ); this is shown by the existence of an additional index $k \in \{1, \ldots, 10\}$.
The last population of the fifth evolution, for example, which was produced by evolutive MM2-energy minimization of an isomer having MOLGEN 3.5-number 22 using parameter values $(\mu_1, \lambda_1) = (4, 6)$ and $\vartheta_{3,2} = 36$ is denoted by $\mathcal{P}(\mathcal{M}_3, 1, 2, 5)_{31}$.

3.2 Single Examinations

In order to get a first impression of the objective function values' evolution please take a look at figure 3. It shows the development of fitness values on positions 1 through 4 in generations 11 through 31. The values are averages taken over ten population sequences that were evolving for $\mathcal{P}(\mathcal{M}_6, 1, 1, .)$ (left hand side) and for $\mathcal{P}(\mathcal{M}_6, 3, 1, .)$ (right hand side).
These population sequences differ in the values used for population sizes only. The figure on the left hand side is derived from populations of 4 individuals, the figure on the right hand side is a derivation from populations containing of 16 members. A comparison of these two sequences of averages shows that variations in objective function values seem to be damped when population sizes are growing.
Worst individuals of small populations vary very much. This could be regarded as a hint for population sizes, especially the size λ of temporary populations having been chosen too small. One comment on this high variability could be that since too few new individuals are generated for $P'(t)$ there are too few being comparable

to the best ones in the primary population $P(t)$. This, together with subsequent selection and classification, might cause the observed high variability in the values of the worst individuals of $P(t+1)$.
It has been our purpose however to get results being satisfactory as to both objective function values and sharp classifications by computations consuming as little time as possible. Besides, we wanted classes close to minima once found in a temporary population to get members of consecutive primary populations and not to get wasted just because population size μ may have been chosen too small compared to population size λ.
That is why temporary populations $P'(t)$ were chosen to be only 50% and 25% bigger than primary populations $P(t)$; so some kind of statistical fluctuation had to be accepted in the sequence of objective function values of high order individuals in small populations. It is possible however to enlarge primary as well as temporary populations to reduce these fluctuations, at the expense of an increasing time consumption, of course.
On the other hand premature convergence and stagnation of populations could be prevented by using the conjugate gradient method in conjunction with classification between genetic operators. Small populations often entail those phenomenona when ordinary algorithms of evolutionary programming are in use. So these algorithms mostly use parameter values of (μ, λ) such that λ is about five to seven times as large as μ. Premature convergence is an undesirable effect which occurs when high selective pressures and small populations are in use; it emerges from a very good individual exponentially taking over all the positions in subsequent generations, assumed this individual stays better than average in these generations. And the latter is the case especially if there are only few newly generated individuals in each generation, hence small λ. Here on the opposite the parallel optimization of all the fellow individuals and most of all the deletion of isomorphic molecules prevent this dangerous effect which yields stagnation instead of further evolution.

3.3 Values of the Energy Function

All of the evolutionary processes that had been run were analysed according to values of the objective function produced in the last, i.e. the 31st populations. These are the values that would be observed in real world applications where the evolutionary process that lead to these values will be discarded. Averages

$$\bar{x} := \frac{1}{10} \sum_{\substack{j=1,\ldots,\mu \\ k=1,\ldots,10}} x_k^{(j)}$$

were calculated for every ten-tuple $\mathcal{P}(\mathcal{M}_l, i, j, k)_{31} = (g_k^{(1)}, \ldots, g_k^{(\mu)})$ of populations $(k = 1, \ldots, 10)$. Corresponding statistical deviations $\bar{s}$ were determined, too. Table 1 also contains the best values $\bar{m}$ of the MM2 force field that could be achieved during evolutionary iterations.

$$\bar{m} := \min\{x_k^{(j)} \mid j = 1, \ldots, \mu, k = 1, \ldots, 10\}.$$

		18			36			72		
		$\bar{x}$	$\bar{s}$	$\bar{m}$	$\bar{x}$	$\bar{s}$	$\bar{m}$	$\bar{x}$	$\bar{s}$	$\bar{m}$
$\mathcal{M}_1$	(4,6)	142.677	43.4924	125.399	134.721	42.8669	124.747	126.773	1.87127	124.673
	(8,10)	140.956	43.7527	124.591	137.311	29.4504	124.678	132.942	25.6125	124.699
	(16,20)	177.207	76.781	124.696	157.174	57.5804	124.675	149.434	45.8427	124.679
$\mathcal{M}_2$	(4,6)	157.927	20.4697	145.305	150.229	5.16504	145.276	155.078	27.8875	145.394
	(8,10)	184.479	67.7222	145.277	181.7	80.462	145.269	168.015	43.4763	145.292
	(16,20)	207.785	68.236	145.287	186.862	52.4538	145.263	180.992	43.8119	145.298
$\mathcal{M}_3$	(4,6)	221.734	28.0142	206.615	213.307	11.2948	206.401	214.598	21.9358	206.148
	(8,10)	243.589	80.3413	206.199	226.012	32.6836	206.536	221.868	29.1775	206.07
	(16,20)	252.341	50.5876	206.744	246.79	41.8442	206.271	245.225	50.307	206.208
$\mathcal{M}_4$	(4,6)	133.511	12.7065	121.946	128.431	11.29	121.765	132.786	35.3137	121.998
	(8,10)	130.393	8.51669	121.802	128.924	12.316	121.756	125.579	4.47907	121.734
	(16,20)	134.213	20.749	121.895	129.604	9.88545	121.43	125.159	3.45673	121.515
$\mathcal{M}_5$	(4,6)	203.655	60.7655	160.495	199.429	58.4659	160.583	247.094	189.064	160.556
	(8,10)	239.913	76.502	160.61	234.944	74.5437	160.48	234.526	66.7302	160.529
	(16,20)	270.494	82.7946	160.515	260.13	69.5188	160.558	264.831	70.0881	160.565
$\mathcal{M}_6$	(4,6)	147.725	22.2858	133.561	144.861	23.9266	133.554	138.69	3.16951	133.549
	(8,10)	160.906	61.962	133.877	142.222	8.15394	133.559	140.505	4.80386	133.527
	(16,20)	178.686	68.9278	133.532	161.489	41.9203	133.533	160.657	57.1597	133.566
$\mathcal{M}_7$	(4,6)	186.247	54.9959	156.924	174.432	35.7513	156.947	165.488	27.5154	156.756
	(8,10)	210.624	68.2593	156.784	203.447	61.2632	156.795	208.74	68.6633	156.794
	(16,20)	248.607	79.6498	156.726	229.136	59.5561	156.788	237.049	78.4861	156.756
$\mathcal{M}_8$	(4,6)	153.28	37.3461	140.305	146.505	24.6583	140.311	142.957	2.13568	140.213
	(8,10)	162.406	51.3329	140.19	147.841	33.0719	140.139	143.547	8.52481	140.215
	(16,20)	188.584	65.1431	140.244	183.362	62.4722	140.135	178.626	61.1839	140.134
$\mathcal{M}_9$	(4,6)	228.779	29.1868	206.442	215.774	13.0911	205.52	209.777	3.69471	205.264
	(8,10)	224.514	23.3703	205.028	215.862	13.0416	205.651	213.091	10.7428	205.425
	(16,20)	233.267	32.6374	205.454	222.908	19.6878	205.444	219.104	17.9315	205.003
$\mathcal{M}_{10}$	(4,6)	216.737	11.4492	204.599	210.447	3.8052	203.339	209.985	3.76558	204.244
	(8,10)	215.768	11.5566	203.994	211.471	6.18994	204.522	209.006	4.53537	202.61
	(16,20)	214.698	12.6357	204.018	210.848	5.16541	203.011	208.346	3.25108	203.009

Table 1: Averages, variances and minima collected in last generations.

Now some of the effects shown in table 1 have to be interpreted. First of all it indicates that an enlargement of the values for parameter ϑ_3, i.e. the number of loops performed during conjugate gradient iterations, resulted in partly considerable improvement of averages $\bar{x}$. The latter decrease in 29 out of 30 trials if 72 instead of 18 conjugate gradient steps are done; the decrease amounts to an average of 6.4% compared to particular values got for 18 conjugate gradient steps. If there are 36 instead of 18 steps performed this decrease amounts to average 4.8%; again it has occurred in 29 out of 30 trials.
It is remarkable that changing the value of ϑ_3 does not noticeably influence the best values $\bar{m}$ ever achieved. So it is very likely that even those trials which spent least time on conjugate gradient computations revealed very good approximations to the absolute minimum intrinsic to the MM2 force field of the according molecule. This means that 18 conjugate gradient steps per generation can be enough if only a rough approximation to the absolute minimum is sought.
The same is valid for changing the values of μ and λ, i.e. for enlarging the population sizes. As far as possible this leaves absolute minima $\bar{m}$ found during the iterations untouched, too.
Averages of energy values however grow quite much if populations are enlarged; here they have grown in 27 out of 30 test cases. On an average this growth totalled 18% when population sizes have been raised from $(\mu_1, \lambda_1) = (4, 6)$ to values of $(\mu_3, \lambda_3) = (16, 20)$.
Compared to the results obtained by ordinary evolutionary or genetic algorithms this conduct seems astonishing at first glance. Since these methods do not allow stable subpopulations developing on distinct minima of the objective function it is observed that the individuals of generations they produce increasingly concentrate on one minimum. So the result of ordinary evolutionary algorithms is one best individual, and all of the other individuals are crowded around this one. Our algorithm however incorporates a classification step (cf. definition of selection operator s in subsection 2.7); there is only one individual allowed on each peak of the energy function. All the other individuals which are equivalent to this best one are discarded. They are substituted by other individuals that may perform worse since they are generated by genetic operators. Equivalence of individuals is defined in terms of Benecke's method that detects full structure isomorphism between three-dimensional molecular structures. So the individuals in all of the generations $P(t)$ are kept apart; each of them belongs to another equivalence class.
Assuming that the number of stereo isomers of a molecule in a certain way refers to the number of minima whose objective function values are near the absolute minimum we can explain the differences of population averages $\bar{x}$ in trials $\mathcal{P}(\mathcal{M}_7, i, 3, .)$ and $\mathcal{P}(\mathcal{M}_{10}, i, 3, .)$,for example (with $i = 1, \ldots, 3$). When discussing figure 10 it will be obvious that all of the tests we made yielded only four clearly distinguishable classes of three-dimensional conformations for the constitutional isomer $\mathcal{M}_7$ of C_8H_{16}. Their energy values were in about 10% around $\bar{m}$; all of the other conformations' values were beyond 172. The former four minima represent stereo isomers of $\mathcal{M}_7$; they have been found in each of the ten trials we made. All the other conformations showed essentially worse energy values; but because of the classification these local minima are also included in populations when their sizes ϑ_3 are greater than $\vartheta > 4$. This consideration explains the sequence of growing val-

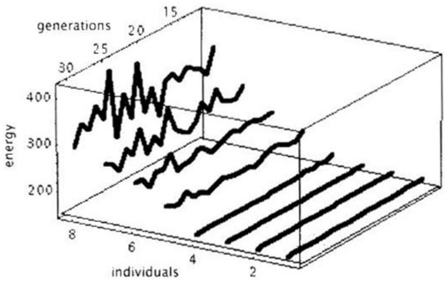

Figure 4: Development of average values taken over 10 population sequences of individuals in $\mathcal{P}(\mathcal{M}_7, 2, 3, .)$.

ues $(165.448, 208.74, 237.049)$ for growing parameters (μ_i, λ_i) for the molecule $\mathcal{M}_7$. Please take a look at figure 4 showing the corresponding courses of values.
The second above-mentioned trial is somewhat different. The best values for averages $\bar{x}$ and deviations $\bar{s}$ were obtained when the biggest populations had been in use. Those values were 208.346 and 3.25108 respectively. All of the averages that have been obtained for $\vartheta_{3,3}$ are very close to the particular absolute minimum. These observations reveal that this molecule has lots of low energy isomers which are found the more easily and the more precisely the higher the values (μ_i, λ_i) are. And indeed MOLGEN 3.5 gives 8 stereo isomers for molecule $\mathcal{M}_{10}$. Comparing pictures 4 and 5 shows that all of the sequences of average values for $\mathcal{M}_{10}$ behave much more uniformly than those for $\mathcal{M}_7$.

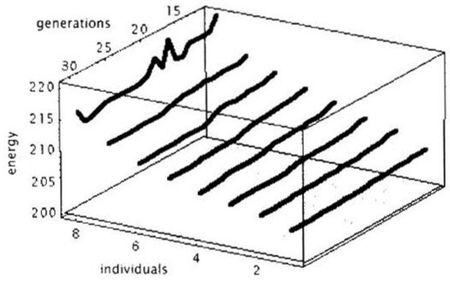

Figure 5: Development of average values taken over 10 population sequences of individuals in $\mathcal{P}(\mathcal{M}_{10}, 2, 3, .)$.

We have observed that the numbers listed in table 1 are too much dependent on the particular problems as to allow any statement about the algorithm itself when the population sizes are changed. That is why we put together table 2. It contains

		18		36		72	
		$\bar{\bar{x}}$	$\bar{\bar{s}}$	$\bar{\bar{x}}$	$\bar{\bar{s}}$	$\bar{\bar{x}}$	$\bar{\bar{s}}$
$\mathcal{M}_1$	(4,6)	142.677	43.4924	134.721	42.8669	126.773	1.87127
	(8,10)	126.389	1.11027	125.91	1.01631	125.84	0.817922
	(16,20)	125.996	0.900844	125.434	0.573709	125.418	0.642924
$\mathcal{M}_2$	(4,6)	157.927	20.4697	150.229	5.16504	155.078	27.8875
	(8,10)	148.518	3.3805	148.196	6.10003	147.071	1.90898
	(16,20)	147.436	2.12088	147.042	1.85606	146.94	1.77573
$\mathcal{M}_3$	(4,6)	221.734	28.0142	213.307	11.2948	214.598	21.9358
	(8,10)	212.326	2.85202	210.768	2.44315	209.906	2.62462
	(16,20)	210.858	2.5769	210.574	2.59856	209.763	2.48124
$\mathcal{M}_4$	(4,6)	133.511	12.7065	128.431	11.29	132.786	35.3137
	(8,10)	124.906	2.03223	124.263	2.18545	123.486	1.32591
	(16,20)	123.637	0.918487	122.741	0.9683	122.279	0.385971
$\mathcal{M}_5$	(4,6)	203.655	60.7655	199.429	58.4659	247.094	189.064
	(8,10)	181.707	28.7031	181.52	27.9834	184.355	30.083
	(16,20)	174.683	18.2179	176.096	18.1107	178.674	22.4954
$\mathcal{M}_6$	(4,6)	147.725	22.2858	144.861	23.9266	138.69	3.16951
	(8,10)	138.833	2.33664	137.621	2.21671	137.25	2.12473
	(16,20)	137.1	2.45957	136.474	1.99771	136.38	1.88386
$\mathcal{M}_7$	(4,6)	186.247	54.9959	174.432	35.7513	165.488	27.5154
	(8,10)	162.002	7.46624	160.183	2.43969	159.777	2.18594
	(16,20)	160.416	2.65655	159.968	2.41481	159.767	2.24274
$\mathcal{M}_8$	(4,6)	153.28	37.3461	146.505	24.6583	142.957	2.13568
	(8,10)	141.623	0.891722	141.36	0.701238	141.363	0.745844
	(16,20)	141.423	0.744183	141.147	0.724842	141.151	0.722835
$\mathcal{M}_9$	(4,6)	228.779	29.1868	215.774	13.0911	209.777	3.69471
	(8,10)	211.792	3.48048	209.225	2.38418	208.599	2.26878
	(16,20)	209.457	2.06792	208.092	1.38289	207.206	1.3283
$\mathcal{M}_{10}$	(4,6)	216.737	11.4492	210.447	3.8052	209.985	3.76558
	(8,10)	208.921	3.03477	207.845	1.94724	206.069	1.75985
	(16,20)	207.161	1.85451	206.003	1.36616	204.786	1.06135

Table 2: Averages, variances and minima over the particular best individuals of last generations.

average numbers

$$\bar{\bar{x}} := \frac{1}{40} \sum_{\substack{j=1,\dots,4\\k=1,\dots,10}} x_k^{(j)}$$

for every generation of the sequences of populations created. Statistical deviations are also listed again. The table indicates in which way enlarging the gene-pool to 8 or 16 individuals per population affects the best four approximate solutions found. So any impact of classification is ignored. The fact that possibly worse individuals are forced to be kept in populations just because of the enlargement of populations is left aside since $\bar{\bar{x}}$ is affected only by the best four individuals.
It is obvious that enlarging populations has advantages even if only the best classes are subject to observation. A growth of populations from 4 to 16 members results in an average 6.4 % improvement of the particular value $\bar{\bar{x}}$; it was present in all of the 30 trials. Doubling population size to 8 individuals still has resulted in an average 5.6 % improvement again in all of our trials. Deviations $\bar{\bar{s}}$ also improve drastically, often this improvement is as big as whole orders of magnitude.

3.4 Results of Classification

In this section insight is given into the results of classifications that were done for all of the trials. Each individual in each generation of the evolution calculated by our al-

gorithm represents an equivalence class of three-dimensional molecular structures. They are indicated by the definition of Benecke's method for detecting structural isomorphism. An important question about our evolutionary approach is whether it is able to find a representative for certain classes in nearly every run. Only if this is true our algorithm can be useful for practical applications since then it should give reliable results after at most a few (possibly one) evolutions. Statistical evaluations like the ones performed in this work should not be necessary then, either.
In order to get an answer to this question the final results $\mathcal{P}(\mathcal{M}_l, i, j, k)_{31}$ with $k = 1, \ldots, 10$ were successively classified in the following way. For brevity indices i,j and l shall be kept fixed; a set $\mathcal{P}$ shall contain all of the $10\mu_i$ final results of a particular trial; like above $\Phi(x)$ will be written instead of $\Phi(g)$ having an individual $g = (x, \sigma)$. Like when defining the selection operator s in equations 4 and 5, the definition

$$\begin{aligned} g_1 &:= \arg\min\{\Phi(g) \mid g \in \mathcal{P}\}, \\ \kappa_1 &:= \{g \in \mathcal{P} \mid \kappa(g_1, g) = 0\}; \end{aligned} \tag{6}$$

is followed by

$$\begin{aligned} g_{j+1} &:= \arg\min\{\Phi(g) \mid g \in \mathcal{P} \setminus \bigcup_{j'=1}^{j} \kappa_{j'}\}, \\ \kappa_{j+1} &:= \{g \in \mathcal{P} \mid \kappa(g_{j+1}, g) = 0\} \setminus \bigcup_{j'=1}^{j} \kappa_{j'}. \end{aligned} \tag{7}$$

with having $\bigcup_{j'=1}^{j} \kappa_{j'} \subsetneq \mathcal{P}$ for $j \in \mathbb{N}$. We define $\bar{j}$ to hold the number of g_j of this kind, i.e. $\bigcup_{j'=1}^{\bar{j}} \kappa_{j'} = \mathcal{P}$.

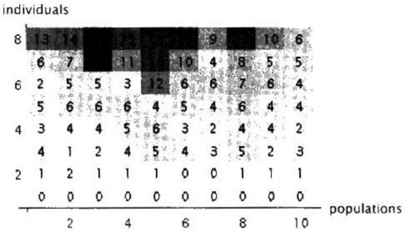

Figure 6: Class affiliations of individuals in $\mathcal{P}(\mathcal{M}_3, 2, 3, .)_{31}$.

Figure 6 has ensued from classification of all the individuals of ten populations $\mathcal{P}(\mathcal{M}_3, 2, 3, .)_{31}$ in this way. These populations are grouped as columns in the upper figure; individuals are arranged in the columns ordered in accordance with their MM2 force field values, e.g. the bottom row of small boxes comprises the best individual of every population. A number in each single box represents the class this particular individual stands for according to the formulae 6 and 7.

Obviously the best individuals all belong to one class, namely class 0 in figure 6; 70% of all second-best individuals are members of class 1. For individuals with worse objective function values there is no exact statement that can be made, as they belong to various classes. Observe that this perfectly fits the fact that $\mathcal{M}_3$ has two stereo isomers.
In figure 7 a link is established between different classes, which individuals (x, σ) are affiliated to, and their particular objective function values $\Phi(x)$. Data underlying figure 7 have been taken from the same trials as the data for figure 6. In the figure these data points having ordinates greater than 0 belong to individuals whose energy is enumerated as abscissa values. The figure contains the 50% best individuals of the last generation of the trial. Every point mentioned above corresponds to one point with ordinate value less or equal to 0; with the the ordinate values such points designate the class the corresponding individual is member of. Because there are ordinate values from -6 to 0 every individual shown in figure 7 belongs to one out of 7 different classes.
There are exactly 12 individuals lying on the first branch of objective function values between 206 and 208; they are members of class 0. The second branch between 208 and about 210 almost exclusively contains individuals of class 1. On the right hand side of these two branches there are individuals that belong to various classes; their objective function values begin ascending steeply. Observations like these agree with $\mathcal{M}_3$ having two stereo isomers, too.

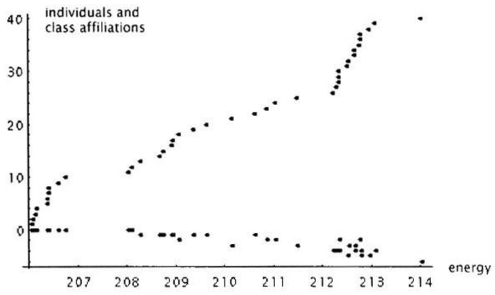

Figure 7: Class affiliations and values of the objective function for the best 50% of the populations of $\mathcal{P}(\mathcal{M}_3, 2, 3, .)_{31}$.

On the following pages there are figures for molecules having one, three and four stereo isomers, showing similar results like figure 7 shown above. However the molecule $\mathcal{M}_2$ having only one stereo isomer shows four sharply separate classes in a quite narrow range of energy values; this is opposite to naive expectation. It means that our evolutionary method is able to compute results being more subtle than just a count of stereoisomers. It is exactly here where our approach of optimizing energies without incorporation of any a-priori heuristic information proves valuable. At the end of the presentation of our results a rather small table shall be discussed. It contains the number $\hat{j}$ of classes which were 'hit' at least 9 times by individuals

i	1			2			3		
j	1	2	3	1	2	3	1	2	3
$\mathcal{M}_1$	1	1	3	3	3	3	4	4	6
$\mathcal{M}_2$	1	1	3	2	3	4	4	5	6
$\mathcal{M}_3$	1	2	2	2	4	4	5	5	5
$\mathcal{M}_4$	0	0	0	0	0	2	0	3	4
$\mathcal{M}_5$	2	2	2	2	2	2	3	4	4
$\mathcal{M}_6$	0	0	0	1	3	3	7	7	8
$\mathcal{M}_7$	1	2	3	4	4	4	4	4	4
$\mathcal{M}_8$	0	0	0	3	5	5	6	8	7
$\mathcal{M}_9$	0	0	1	1	1	2	3	7	9
$\mathcal{M}_{10}$	0	0	0	0	0	0	0	0	0

Table 3: Number of classes containing at least 9 individuals.

of 31st generations of every trial; so we have got the numbers listed in table 3 by counting the members of the following sets:

$$\hat{j} := \left| \{ 1 \leqslant j \leqslant \bar{j} \mid |\kappa_j| \geqslant 9 \} \right| .$$

In fact these $\hat{j}$ different classes occurred in at least 90% of the trials with the particular values of parameters. Note from that table that in case of $\mathcal{M}_{10}$ none of the classes was big enough to be counted in any trial. A population size of 16 individuals seems to be too small to find stable classes in enough, i.e. in at least 9, of the population sequences.

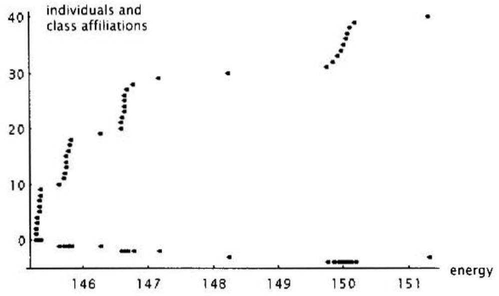

Figure 8: Class affiliations and values of the objective function for the best 50% of the populations of $\mathcal{P}(\mathcal{M}_2, 2, 3, .)_{31}$.

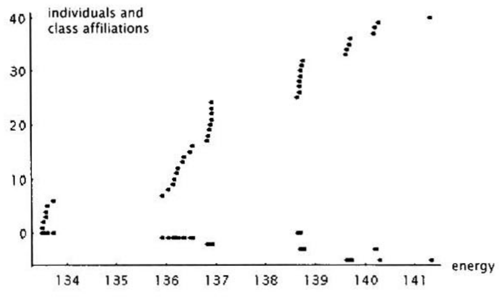

Figure 9: Class affiliations and values of the objective function for the best 50% of the populations of $\mathcal{P}(\mathcal{M}_6, 2, 3, .)_{31}$.

3.5 Running Times and Complexity of the Algorithm

All the tests that have been run have shown that the time required for all the computations clearly was dominated by the running time of the conjugate gradient method. Computations made by evolutionary operators can be neglected, as they only perform simple tasks like computation of pseudo random numbers or sorting individuals according to their objective function values. Therefore the complexity of our hybrid evolutionary algorithm is summarized by $O((\lambda+\mu)\vartheta_4 C(m))$, supposed $C(m)$ determines the complexity of the conjugate gradient method, which depends on the individuals' length m [Spe93].
The tests we presented in the previous subsections have been run on a quite outdated Pentium-150 machine. The computation of trials with indices $i = j = 1$ has taken about 40 seconds, for $i = 2$, $j = 1$ about 80 seconds as expected, for $i = 1$, $j = 2$ about 85 seconds etc. These running times are very much shorter than those Benecke had to spend on his detection routine for conformational isomers.

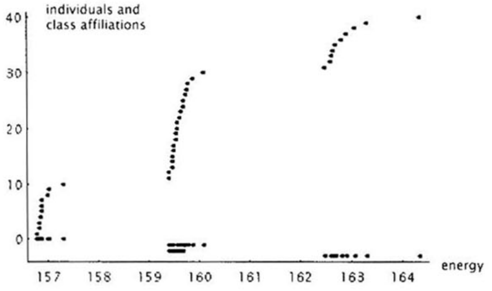

Figure 10: Class affiliations and values of the objective function for the best 50% of the populations of $\mathcal{P}(\mathcal{M}_7, 2, 3, .)_{31}$.

4 Conclusions

In this paper we have presented a hybrid algorithm combining three modern methods for optimization and classification of energy levels of molecules. The usage of Bäck's formalism for the description of evolutionary algorithms made it possible to unite those methods on a formal level, too. There have been many tests which yielded very good energy levels and stable classifications simultaneously. Running times were fast; they could be accelerated even more if our algorithm was made to work in parallel. Such an attempt is possible particularly since the hybrid method works on vectors of approximate solutions.
As far as the three-dimensional classification is concerned, our results indicate that it does have a counterpart in an diversification on the scale of energy values, but to observe this a quite precise analysis is necessary.

References

[Bäc96] Thomas Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford University Press, New York, 1996.

[Ben98] Christof Benecke, *Objektorientierte Darstellung und Algorithmen zur Klassifizierung endlicher bewerteter Strukturen*, Match **37** (1998), 7-156.

[GCJ90] J. Gasteiger, C.Rudolph, and J.Sadowski, *Automatic generation of 3d-atomic coordinates for organic molecules*, Tetrahedron Computer Methodology **3** (1990), no. 6C, 537-547.

[SG93] Jens Sadowski and Johann Gasteiger, *From atoms and bonds to three-dimensional atomic coordinates: Automatic model builders*, Chemical Reviews **93** (1993), 2567-2581.

[Spe93] Peter Spellucci, *Numerische Verfahren der nichtlinearen Optimierung*, ISNM Lehrbuch, Birkhäuser Verlag, Basel, 1993.