



## A genetic algorithm for the automated generation of small organic molecules: Drug design using an evolutionary algorithm

Dominique Douguet<sup>a</sup>, Etienne Thoreau<sup>a</sup> & Gérard Grassy<sup>b</sup>

<sup>a</sup>GALDERMA R&D, B.P. 87, F-06902 Sophia Antipolis Cedex, Valbonne, France (E-mail: douguet1@caramail.com, etienne.thoreau@galderma.com); <sup>b</sup>Centre de Biochimie Structurale, UMR CNRS 9955, INSERM U414, Université Montpellier I, Faculté de Pharmacie, avenue Charles Flahault, F-34060 Montpellier Cedex, France (E-mail: drtt.lr@wanadoo.fr)

Received 5 August 1999; Accepted 20 December 1999

**Key words:** automated structure generation, drug design, genetic algorithm, molecular modeling, QSAR, SMILES, variable mapping

### Summary

Rational drug design involves finding solutions to large combinatorial problems for which an exhaustive search is impractical. Genetic algorithms provide a novel tool for the investigation of such problems. These are a class of algorithms that mimic some of the major characteristics of Darwinian evolution. LEA has been designed in order to conceive novel small organic molecules which satisfy quantitative structure-activity relationship based rules (fitness). The fitness consists of a sum of constraints that are range properties. The algorithm takes an initial set of fragments and iteratively improves them by means of crossover and mutation operators that are related to those involved in Darwinian evolution. The basis of the algorithm, its implementation and parameterization, are described together with an application in de novo molecular design of new retinoids. The results may be promising for chemical synthesis and show that this tool may find extensive applications in de novo drug design projects.

### Introduction

Traditional Quantitative Structure-Activity Relationship Analyses (QSAR) establish linear or non-linear models of biological activity dependence on molecular properties. The models are useful either to explain the activity of a specific class of compounds or to predict the activity of new related analogues. Nevertheless, de novo design based on such QSAR models is difficult to achieve when one wishes to find not only new analogues but also novel chemical structures (not yet patented for example). In order to assist our imagination, we have designed an expert system which creates new structures with the desired shape, lipophilic and electronic properties. The 'virtual chemistry space' is extremely large, evaluated to  $10^{100}$  [1] possible molecules, and thus must be explored by an optimization method in order to find suitable structures. Results from quantitative and qualitative characterization of known drug databases [2, 3] can also be included to

generate 'drug-like' structures and to streamline the search in this huge space.

A simulation of the evolutionary pressure that is effected by natural selection can be incorporated into artificial intelligence algorithms to rapidly find good, if not optimal, solutions. The basic idea is to maintain a population of candidate solutions which evolves under a selective pressure that favors better solutions. The more fit a member of the population, the more likely it is to produce offspring. This selective pressure and the evolutionary operators *crossover* and *mutation*, which are based on ideas first described by Charles Darwin [4], can be used to optimize solutions to a wide variety of problems. Evolutionary methods [5–8] have been developed in the areas of elucidation, design [9] and modeling of chemical and biochemical structures [10]. Genetic algorithms (GA) were developed by Holland in the 1970s [11] and only recently their potential for solving combinatorial optimization problems has been explored [12, 13].

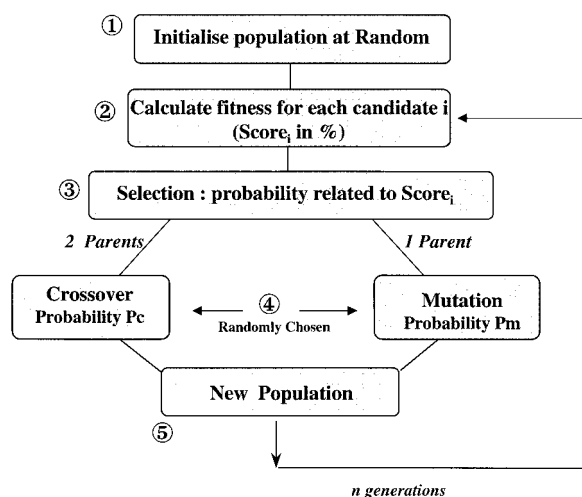


Figure 1. General flowchart for a genetic algorithm. ① An initial population of candidate solutions is generated by a random process. ② The fitness of each candidate is evaluated via a 'fitness function', which takes as input a candidate solution and returns a numeric score. ③ Selection criteria are applied to chosen candidates based on their fitness score for breeding. ④ Breeding functions are applied to produce new solutions, which replace the parent solutions. ⑤ The cycle (or generation) continues: go to step ② until convergence criterion is met (solutions are no more improved).

GA has been adapted to optimize chemical structures and evolve them to more useful compounds in the field of rational drug design. Glen and Payne [14] have applied a GA to the design of structures based on a wide variety of user-defined constraints. Their method can utilize scalar constraints such as molecular weight, surface constraints such as electrostatic potential, and grid constraints such as the hydrophobic/hydrophilic character of nearby atoms. This program operates, as does the LeapFrog program [15], directly upon the 3D molecular structure. PRO\_LIGAND [16] generates new leads by assembling fragments from substructure libraries. Then high- and low-quality molecules are used as the initial population for an optimization process based on GA. Results indicate that the GA optimization step not only increases the average fitness of the solutions, but also finds better solutions than the design step. The Blaney et al. [17] program was based upon two different types of encoding. First, molecular graph encoding was used for manipulating molecular structures and secondly, the SMILES line notation, one-dimensional representation of a molecule [18], was used for communication and storage of molecules. The fitness of each of the population members was evaluated by flexibly docking the molecule into the relevant active site.

The present program LEA (Ligand by Evolutionary Algorithm) is able to operate directly on the SMILES line notation and resembles the Glen and Payne system in terms of constraints. SMILES line notation is useful for communication and storage but its manipulation by crossover and mutation operators is more fastidious. Our program uses the concept of fragment combination [19]. Fragment libraries are of four types, fragment, function, ramification (small fragment) and amino acid database. Our results show that this GA is able to optimize molecules with appropriate 3D properties obtained from the SMILES manipulation. The algorithm, its implementation and parameterization are first described. The first example of optimization is the de novo design of retinoids which has been used to explain the influence of the GA parameters. A second example of optimization is presented to show how structures evolve during optimization (find salicylic acid structure). Finally, some promising retinoid analogs are presented.

## Methods

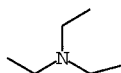
### Genetic algorithms

By analogy to the natural model, each candidate is called a chromosome. Each chromosome consists of genes, each gene representing a parameter or feature of the solution. Chromosomes are commonly a collection of integers or binary numbers, but can also be virtually any other type of information such as characters. The basic steps used in a genetic algorithm follow the patterns shown in Figure 1. First, an initial population of candidate solutions is generated by a random process. Then, the fitness of each candidate is evaluated via a 'fitness function', which takes as input a candidate solution and returns a numeric score (②). The breeding steps ③ and ④ select one or two candidates (parents) based on their fitness score and a breeding function, crossover or mutation, to produce new solutions, which replace the parent solutions (⑤). Steps ② to ⑤ are iterated until convergence is met (solutions are no more improved).

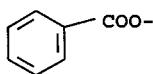
Genetic methods are stochastic in nature and have a finite probability of missing the optimal solution while finding near-optimal solutions, but they have proved to be very effective as search/optimization algorithms [20].

**A) SMILES line notation**

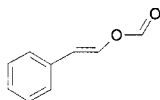
Butane  
CCCC



Triethylamine  
CCN(CC)CC



Benzoic Acid  
C1=CC=CC(=C1C(=O)O)

**B) Distinct Parts**

C1=C(C=COC(=O))C=C(C=C1)

**Skeleton**

this example cannot be 'linearly' segmented in parts

C1=C(C=COC(=O))C=C(C=C1)

**Branched group of the first order**

this example is a viable group which can be replaced, lengthened or used as a fragment

C=COC(=O)

**Branched group of the second order**

this group is not viable on their own (not a fragment). It can only be replaced

=O

Figure 2. Examples of molecules with their SMILES representation. Distinction is made between skeleton and branched groups.

**SMILES line notation**

SMILES (Simplified Molecular Input Line Entry System) [18] is a linear notation for molecules consisting of a series of characters not including spaces. Atoms are represented by atomic symbols. Hydrogens are generally omitted. Charges or unusual hydrogen attachments must be represented inside brackets. Symbols beginning with a lower case letter represent aromatic atoms but typing cyclohexatriene C1=CC=CC=C1 is exactly the same thing and is the notation employed in LEA. Adjacent atoms are assumed to be connected by a single bond. Double and triple bonds are represented by '=' and '#' respectively. Branches are indicated by enclosing the branched group in parentheses. Ring closures are indicated by pairs of matching digits representing extra bonds (Figure 2A). The basic SMILES rules are adequate to describe the vast majority of organic molecules and are sufficient for the purpose of this work. In our case, the difficulties come from the creation of specific and efficient operators (mutation and crossover) which act on this notation to meet chemical and SMILES notation rules. The SMILES string is expressed as a skeleton with branched groups of first and second order (Figure 2B).

**Breeding functions**

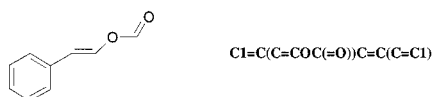
Genetic algorithms use the crossover operator to combine fragments from two parents and the mutation operator to modify the structure from one parent. In each case, these operators must incorporate chemical knowledge like valency rules. Mutation and crossover are randomly selected using a roulette wheel selection. The probability of mutation (pm) and the probability of crossover (pc) are either initially fixed (with  $pm + pc = 1$ ) [21] or modulated by an adaptation strategy. In an adaptation strategy, the probability of a mutation increases (and the probability of a crossover decreases) if the average fitness has been improved in the last breeding step by the mutation operator, otherwise the probability decreases (and the probability of a crossover increases).

In LEA, there are two crossover operators (Figure 4) and 13 mutation operators (Figures 5, 6). When crossover action is required, an operator is randomly selected using a roulette wheel selection in the crossover class and conversely, when mutation is required, one of the 13 is randomly selected using a roulette wheel selection. Crossover and mutation operators search, extract and recombine fragments. In SMILES notation, this means SMILES chain scanning, detection of ring and parentheses to be sure they are never cut during substructure extraction and checking the chemical viability of each fragment. To do this we have included several functions (Figure 3) which rewrite the SMILES string in order to have a skeleton longer than branched group (in parentheses), rewrite substructures in a 'palindromic', or inverse, form when association occurs on the left of the chain, check valency rules, detect undesirable groups [2, 22], segment the skeleton, extract parenthesis content and renumber rings when a mutation branches a new ring on to an existing ring.

Table 1 summarises the modification that will be made by operators (in vertical ground) on the SMILES string (composed by a skeleton and branched groups). Operators have been created in order to access the overall chemical structure space by conferring the ability to act on any part of the molecule (i.e., on the SMILES string) with a certain probability, given as a percentage. The user can define the breeding panel by selecting operators at the beginning of the optimization.

*Table 1.* Summary of the structural modifications made by breeding functions. SMILES strings are made up of a skeleton, branched group(s) of first order (in parentheses) and branched group(s) in a branched group (branched group(s) of second order). Operators may act on any part of the molecule (so of the SMILES string) in order to access the overall chemical structure space. Operator application is allowed on the SMILES part if the sign ✓ is present. Mutate element acts on any element on any part of the SMILES string. Make ring, bond Oxidation and Reduction act also on any part of the SMILES string. It is a pattern matching which allows these operations. The first occurrence in the SMILES string is replaced. Any fragment association must have a probability (indicated in percentage) of being obtained. Insert a Fragment is a replacement of a fragment of the skeleton (from linear segmentation) or a replacement of a branched group (first or second order). Insert an Amino Acid is a replacement of a fragment of the skeleton (from linear segmentation) or replacement of a branched groups (first order only because branched groups of second order are generally small molecular weight). Insert a Function means either a replacement of a fragment of the skeleton (from linear segmentation) or replacement of a branched group (first or second order) or a hydrogen substitution on a carbon (creates a new branched group). Insert a Ramification means either a small group addition in the middle of a fragment (linker) of the skeleton (from linear segmentation) or in a branched group (first or second order) or a hydrogen substitution on a carbon (creates a new branched group). Deletion of a fragment occurs either in the skeleton (from linear segmentation) or in a branched group (first or second order). Crossover 1 point combines either one fragment from the skeleton (from linear segmentation) with a fragment from the second parent or replaces a branched group (first or second order) of the first parent with a fragment from the second parent. Crossover 2 points combines two fragments from the skeleton (from linear segmentation) of the first parent with a fragment from the second parent.

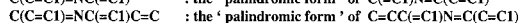
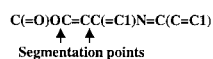
Applied Operator	On Skeleton	On Branched Group 1 <sup>st</sup> Order	On Branched Group 2 <sup>nd</sup> Order	On Carbon
Mutate Element (C, N, O, S, or Halides)	✓	✓	✓	✓
Bond Oxidation or Reduction	✓	✓	✓	
Make Ring	✓	✓	✓	
Substitute by Fragment	✓ 40%	✓ 50%	✓ 10%	0% Perturbation is Too High
Substitute by Amino Acid	✓ 50%	✓ 50%	0% Perturbation is Too High	0% Perturbation is Too High
Add or Substitute by Function	✓ 20%	✓ 30%	✓ 30%	✓ 20%
Insert a Ramification	✓ 20%	✓ 20%	✓ 20%	✓ 40%
Delete a fragment	✓ 25%	✓ 50%	✓ 25%	
Crossover 1 Point	✓ 60%	✓ 20%	✓ 20%	
Crossover 2 Points	✓ 100%			



**Rewrite SMILES string to reduce the length of the string in parenthesis :**



**Segmentation of the SMILES string (fragments must have at least 3 atoms)  
In the case below, fragments will be joined by their right part**

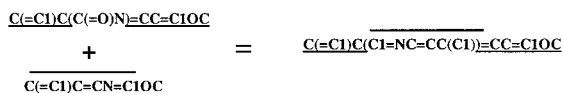
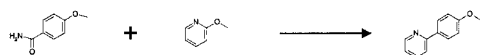


**Undesirable groups are searched and if matched then the molecule is eliminated**

For example : OO, NN, NO, SO, SS, NS, O(Halide)

Figure 3. SMILES strings are manipulated (rewritten, segmented) with respect for chemical and SMILES line notation rules.

(a) Crossover 1 point



(b) Crossover 2 points

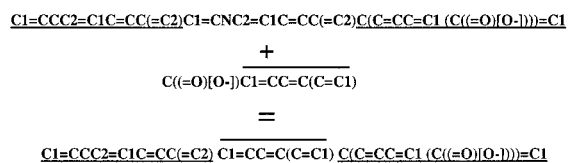
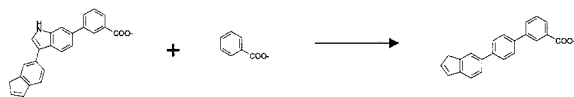


Figure 4. (a) Crossover 1 point – Parent 1 is linearly segmented in only one fragment (OC is constituted by only two atoms) and has a branched group which can be used as a fragment or be replaced by a fragment from Parent 2. Parent 2 is also linearly segmented in one fragment. This segment is rewritten in a 'palindromic' form before replacing the branched group of the first parent. (b) Crossover 2 points – Parent 1 is linearly segmented in three fragments and the middle is replaced by a fragment from Parent 2 which is linearly segmented in two fragments.

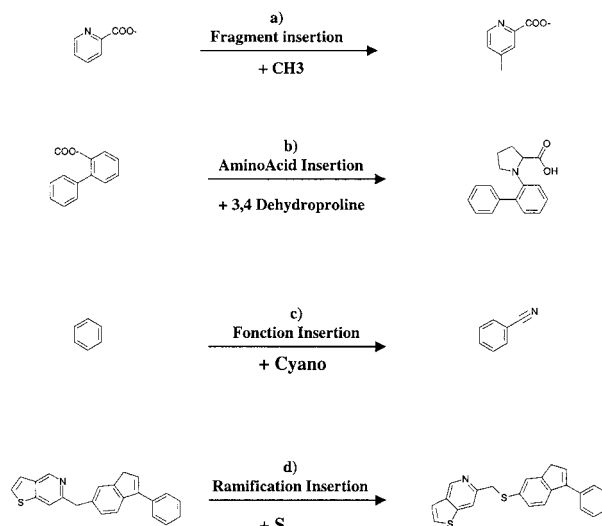


Figure 5. Examples of mutation operations. Fragment addition in Insert Function (c) or Insert Ramification (d) or fragment replacement in Insert Function (c), Insert Fragment (a) and Insert Amino Acid (b).

### Crossover

The crossover-1-point operator selects a random cut-point (splicing parents across a single bond) and combines the first portion of one parent with the second portion of the other to produce a child (Figure 4a). The crossover-2-points operator selects 2 random cut-points and combines the head and the tail of the first parent with a fragment of the second parent to produce a child (Figure 4b). In the segmentation step, parents (their SMILES strings) are linearly segmented and branched groups are extracted (Figure 2b). The fragments always have much more than 3 atoms.

### Mutation

The mutation operator emulates the mutation of DNA. In binary implementations this is done by changing a bit in the chromosome from on to off, or vice versa. In integer or floating point implementations this is done by changing a value to a different one within the allowed range. In our problem, this operator encompasses 13 different mutation types:

– *Insert a fragment* from a fragment database which can contain groups from a user specified database (Figure 5a and Table 1). Fragments are sometimes represented by two or three, chemically equivalent, SMILES representations differing by the substitution point. For example, a benzene group can be substituted in ortho, meta and para position. Each has a SMILES

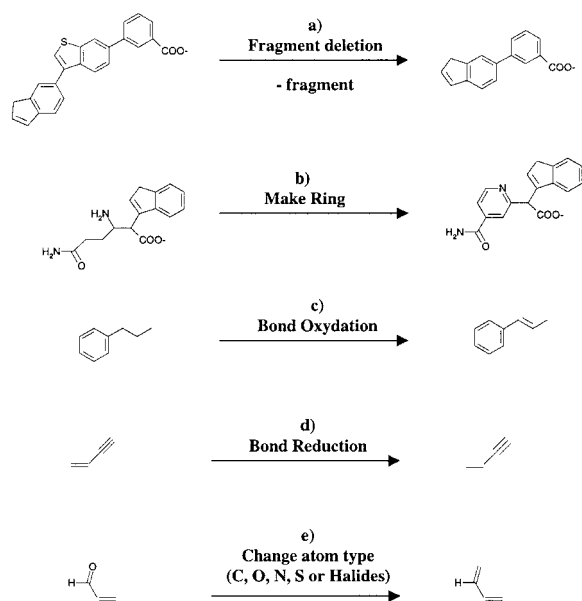


Figure 6. Examples of mutation operations. Fragment deletion (a) or changes inside a molecule by mutation of an element (e), bond oxidation (c), bond reduction (d) or make ring with 1 or 2 carbon addition (b).

notation:  $C1=CC=CC=C1$ ,  $C1=CC(=CC=C1)$ ,  $C1=CC=C(C=C1)$ , respectively.

- *Insert an amino acid* from a 45 amino acid (natural or synthetic) library (Figure 5b and Table 1).
- *Insert a function*. This mutation is useful to increase the probability of having a function in the molecule (Table 1). Functions are incorporated on the left of its SMILES string (Figure 5c).
- *Insert a ramification*. This can promote new substitution points or be used as a linker (Figure 5d and Table 1).

Ramification and function groups have low molecular weight and their related mutation operators can substitute any carbon of the molecule. Conversely, Insert Fragment, Amino Acid or crossovers maintain the original substitution (ortho, meta or para) (Table 1).

- *Delete a fragment* (Figure 6a), *bond Oxidation* (Figure 6c) or *Reduction* (Figure 6d) and *mutate an element* (C, N, O, S, Halides) (Figure 6e) occur only if the environment allows this procedure.

- *Make ring* when there is a match between a pattern allowing ring formation and the SMILES structure. The ring can be aromatic or not, with 5 or 6 atoms and SMILES representation related to the substitution place is randomly chosen (Figure 6b).

## Fitness

The fitness function or scoring function is the primary place in which the traditional genetic algorithm is tailored to a specific problem. The fitness does not have to be well characterized as an equation with calculable derivatives but needs only simple and relevant guidelines. The fitness function is usually the limiting step, in terms of computation time, for the optimization when compared to the breeding process. Score could be either an experimental biological measure [23, 24] or an empirical evaluation (binding energy [17], conformational energy [48], Carbo indices (similarity)).

In LEA, we use physico-chemical properties of molecules as indicative constraints (can be unsatisfied), employed in the scoring function, and imperative constraints (cannot be unsatisfied) like a correct valency. Available constraints in LEA are summarized and defined in Appendix I. These constraints are molecular descriptors such as volume, lipophilicity or electronic properties. They are used in QSAR analyses studies which have been performed using the Variable Mapping method [25–28].

This qualitative technique consists of an evaluation of the distribution of the active and inactive molecules as a function of the distribution of parameter values. The validity of the representation is also estimated by Cluster Significance Analysis [29]. The models based on bounded intervals of descriptors proposed by the Variable Mapping Technique are easily interpretable and can be readily used in the scoring of compounds provided by a GA. The composite fitness, in percentage terms, for a candidate  $i$ , is written as:

$$\text{Score}_i = \sum_p W_p * \text{Score}_{ip} \quad (1)$$

where,  $W_p$  is the weight (in percentage) applied to the property  $p$  (the sum is taken over all properties). For example, when a particular value  $X_t$  is targeted for a property  $p$  then the score of the candidate  $i$  for the property  $p$  ( $\text{Score}_{ip}$ ) is 100% when  $X_{ip} = X_t$  ( $X_{ip}$  is the property value of candidate  $i$ ) or takes a value between 1 and 99% when  $X_{ip}$  lies between  $[X_t - X_t/10; X_t + X_t/10]$  or is equal to 0% if  $X_{ip}$  is out of this range. Figure 7 shows the  $\text{Score}_{ip}$  variation when the constraint is expressed as a bounded interval. Constraint can also be expressed as greater than or less than a bound value. In this case match gives  $\text{Score}_{ip}$  equal to 100%, otherwise 0%.

Usually, fitness is set to be rapidly calculable in order to allow efficient exploration of the search space.

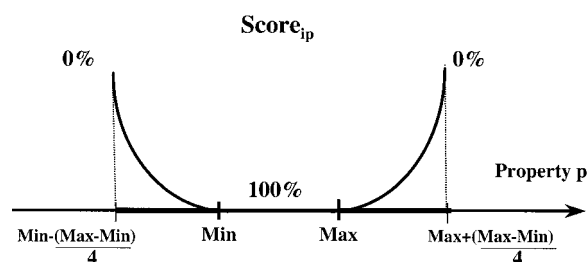


Figure 7. Fitness calculation.  $Score_i$  (%) is calculated over all the properties  $p$  (sum of  $Score_{ip}$ ).  $Score_{ip}$  represents the difference between the property of a new molecule  $i$  and the desired property value (a bounded interval [Min; Max]).

Table 2. The constraints list and the optimized parameter set for the design of retinoid agonist analogues. The definition of the properties is given in Appendix I and the parameter set is introduced in the Results part

Properties <sub>p(1...11)</sub>	Variable mapping	Wp	
Length	12.5; 15.0	20%	
Inertial axis along y	5.0; 9.0	10%	
Inertial axis along z	3.0; 6.0	10%	80%
Solvent accessible surface	500; 600	20%	
Radius of gyration	0.7; 0.9	20%	
Insaturation	<3.5		
Lipophilicity	4.0; 6.0	5%	
Dipole moment	50; 70	5%	
Carboxylate function	Present		
VAC 2 Å (log P)	-0.6; -0.4	5%	
Pharmacophore (AH B)	6.0; 8.0	5%	
Optimized parameter set:			
Generation max: 100			
Population size: 40 (random initialize)			
Elitism: 1			
All operators selected minus amino acid insertion mutation			
Range = [5-9]			

For example, our program evaluates thousands of molecules ( $\approx 3000$ ) during a 6 h run.

### Retinoid fitness

This first example of an optimization problem is the design of retinoid analogs from the evolution of fragments such as benzene. Our retinoid fitness function (Table 2) comes from a previous QSAR analyses study [30] on retinoids [31]. Eleven physico-chemical properties have been selected to characterize agonist retinoids. The constraints are expressed as either bounded intervals (Length must be between 12.5 Å and 15.0 Å) or upper bound (Insaturation < 3.5) or

$$SV_i = Score_{i(sdt)} * (Max-Min)/2 + (Max+Min)/2$$

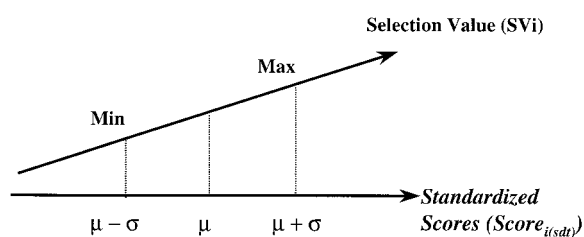


Figure 8. The Fitness scaling is made upon standardized scores. The mean value of scores will have a selection value equal to the medium of the range [Min - Max]. This scaling lowers the difference between scores and gives less fitted candidates a higher chance to be selected.

characteristic (presence of a carboxylate function). The property range differs between published QSAR studies [30] and this constraint list comes from the use of different software (conformational optimization carried out by molecular mechanics rather than semi-empirically). Some additional properties have been implemented: The unsaturation index and the match of a carboxylate fragment are imperative constraints (chiral center detection can also be used). The former serves to conceive aromatic compounds and the latter is a common feature in retinoid agonist structures. If unsaturation is not enough then a structure is eliminated but if the carboxylate pattern is not matched then it is added. Other properties useful to focus the search in a promising sub-space are inertia axes (y and z in our case), solvent accessible surface and lipophilicity. We also take account of an intra-molecular distance constraint between key functional groups which reflects the distance between the carboxylate part of the molecule and a hydrogen bond acceptor and/or donor element placed in the linker. This comes from observations and crystallographic studies [32, 33] which show the importance of H-bond interaction with serine 272 (specific amino acid of the cavity of the receptor RAR $\alpha$ ). In summary, 80% of the composite fitness is associated with structural features in order to rapidly focus on a retinoid structure type.

### Salicylic acid fitness

This second fitness is to find the salicylic acid molecule from the evolution of methane fragments. The fitness is based on very simple constraints such as molecular weight and a number of oxygen atoms (Table 3). In this study, rings are created (not inserted by insertion of a fragment). The objective was just to

Table 3. The constraint list and the optimized parameter set for the design of salicylic acid (target). The definition of the properties is given in Appendix I and the parameter set is introduced in the Results part

Properties <sub>p(1...7)</sub>	Variable mapping	Wp
Molecular weight	152	20%
Insaturation	2.0; 3.0	30%
Carbons	8	10%
Oxygens	3	10%
Element	11 atoms	10%
Carboxyl	Present	10%
Pharmacophore (AH DH)	2.2; 2.8;	10%

Parameter set:		
Generation max:	100	
Population size:	10 (Initial population: 10 methanes)	
Elitism:	1	
All operators minus amino acid and fragment insertion mutation	Range = [2-10]; increases during run until range = [2-18]	

show an example of the evolution of few molecules over several generations starting from methane. This can be done only if the population size is small and if the problem is easy to solve (Appendix II).

### Implementation

The general diagram of LEA is given in Figures 9 and 10. Analogy with the general flowchart for a genetic algorithm in Figure 1 is given with reference numbers. In LEA, the population is initialized with *n* fragments (benzene, methane or any other molecule). The fitness of each molecule is evaluated by Equation 1 which returns a numeric score in percentage (step ②). In step ③, roulette-wheel selection randomly chooses parents for breeding with a selection value for each molecule proportional to its fitness. The standardized score ( $Score_{i(std)}$ ) is scaling in a selection value ( $SV_i$ ), which is the sector size of the roulette wheel (Figure 8). Selection value is null when a structure is non viable (undesirable groups) or does not satisfy imperative constraints such as valency. The scaling process helps to maintain competition by shrinking the difference between scores. As the run goes on, it would be wise to increase the selection pressure in order to force the convergence near the end of the optimization. The

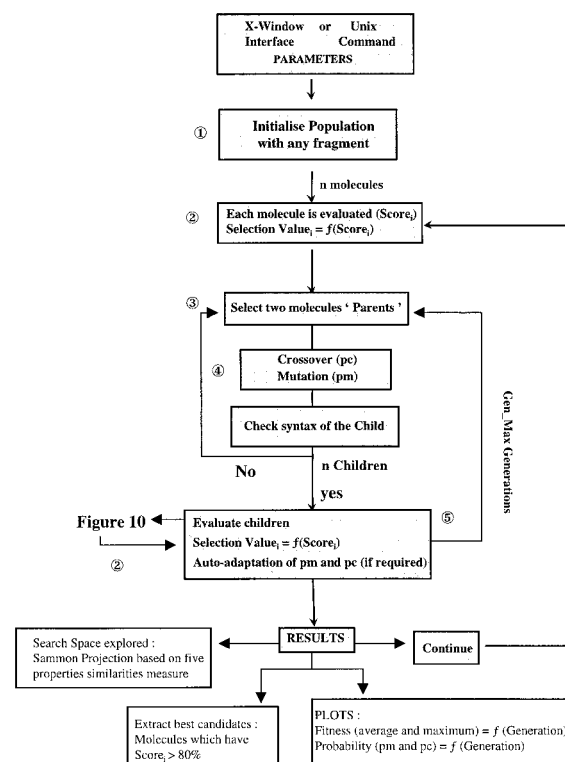


Figure 9. Genetic Design Flowchart. (1) General overview of the steps involved in the optimization of molecules.

range [Min-Max] is then increased along a run by the following equation:

$$\text{Range} = \text{Range} + [\text{Progress} * \text{Range}], \quad (2)$$

$$\text{Progress} = 1 - [\text{Gen\_Max} - \text{Gen}] / \text{Gen\_Max} \quad (3)$$

where Progress is set from 0.0 to 1.0 (0.0 at generation 0 and 1.0 at the generation Gen\_Max). Gen\_Max is the maximum number of generations in the run and Gen the current number of generation. In the present calculation, Range will be multiplied by 2 from generation 0 to generation Gen\_Max.

In Step ④, roulette-wheel selection randomly chooses an operator (crossover or mutation) with a selection value proportional to its probability (pc and pm) respectively). Then, another roulette-wheel selection chooses one of the 2 different crossovers (if crossover action is chosen) or one of the 13 different mutations (if mutation action is chosen). The operator is applied to combine the 2 selected parents (crossover) or to mutate the first selected parent (mutation). Steps ③ to ④ are iterated until the number of



n children is reached. Finally, generation replacement (step ⑤) involves total or partial selection of breeding candidates, whose children replace them. Elitism [34] (hereafter called elite mode) ensures that the best solution in any generation is automatically carried into the next generation.

The convergence criterion, which stops iteration, can be a population convergence criterion or the best candidate convergence criterion. The purpose of our GA is not to have all molecules near the global minimum but to have a new starting structure. Thus, our convergence criterion is related to the fittest (the elite) candidate convergence. Results show that one hundred generations are enough to observe convergence. The application of a genetic algorithm offers fast and powerful optimization properties as well as the generation of a diverse set of possible structures. Each run, due to the non reproductive property, generates a new family of structures. In Results, explored space is displayed by a Sammon Projection [35] on a two-dimensional plot, where the similarity measure is based on 5 properties: length, lipophilicity, volume, radius of gyration, and dipole moment. The projection uses about 10% of the entire population (each 10th generation).

The algorithm LEA is encoded in Perl [36], C and Fortran and was implemented under UNIX with an X-Window interface. Runs were performed on an SGI Indigo2 R10000 175MHz 64MB RAM. Several available programs are interfaced with LEA (Figure 10): Rasmol [37] for the molecular visualization, Discover for the conformational optimization (the CFF91 force field is employed) [38], VAMP for semi-empirical calculation [39], Corina [40] for the SMILES conversion into a .mol2 file format [41], GEPOL [42] for the calculation of volume and surfaces and Chemicalc2 [43] for the lipophilicity and solubility calculation. Typically, the calculation takes about 6 h for 100 generations with a population size of 40 candidates.

## Results

GA requires the specification of several parameters such as the population size, the range [Min-Max] for the fitness scaling, the replacement mode and the mutation and crossover operator rates. A number of test runs were designed to optimize and to evaluate the influence of the value of these parameters. Due to the non reproductive property of GA, three runs were performed for each parameter variation. The fitness used

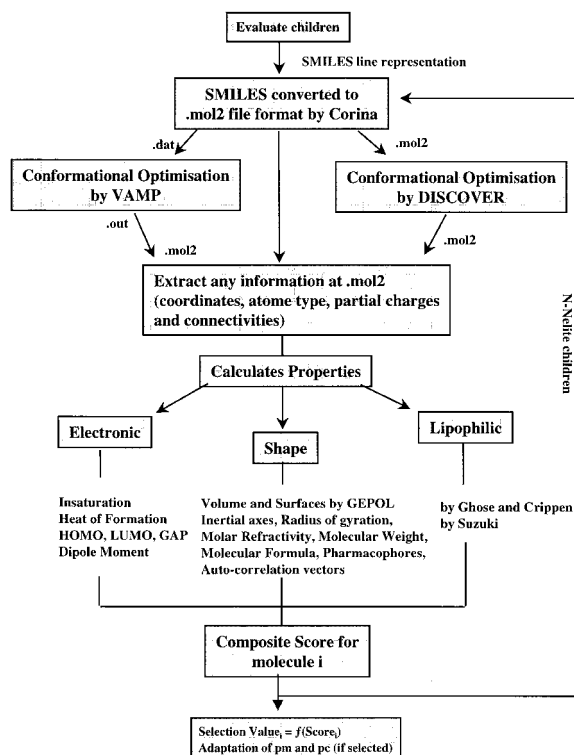


Figure 10. Genetic Design Flowchart. (2) Steps involved during fitness calculation.

for the parameter studies is the retinoid fitness given in Table 2.

### Population size

The influence of the population size is given in Figure 11. Analysis of the curves of the elite shows an improvement of the maximum fitness with 10, 20 and 40 candidates but not with 60 or 80. From the 0th to the 20th generation, the optimization searches and focuses in a promising sub-space (the exploration step). This phase is important since after 20 generations, the elite score plateaus and any following modifications on the elite structure are minor. Small (10) or large (60, 80) population sizes produce less well fitted elites than the 20 or 40 population size optimization. In contrast, the average fitness is superior with a population size of 10 compared to 60 or 80. The population size of 10 is too small and premature convergence leads to a sub-optimal solution. Such a population size allows a rapid diffusion of analogues into the population and thereby a rapid convergence of the average fitness. The effect is inverted for a large population size. Elite is drowned in the mass and cannot emerge from it. In this case

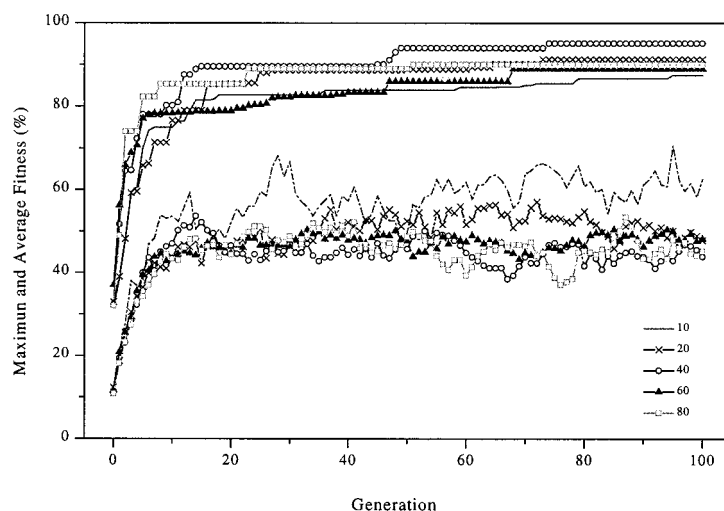


Figure 11. Average and maximum fitness variation with different population sizes (10, 20, 40, 60 and 80 candidates).

the average fitness convergence is slow and the elite fitness is sub-optimal. Finally, the occurrence of an unsuitable molecule (fitness equal to 1%) has more impact on the average fitness of the small population size than on the larger one. This is observed on the average fitness magnitude variation. The slowness of the average fitness convergence also comes from the fact that only one child is produced by crossover (the diffusion of the elite is slower).

#### *Fitness scale*

Influence of range size during fitness scaling is shown in Figure 12. The trends are related, but inversely, to the effects of the population size. Large range (high selection pressure) has an effect similar to the small population size in terms of elite and average fitness convergence (see evolution of Range 4 optimization). The more the elite is selected the more analogues are produced and the faster average fitness convergence is.

#### *Elitism strategy*

In Figure 13, it is obvious that optimization without elite mode is too erratic and that the best structures are forgotten during a run (square curves). In this case, the optimization failed. As already mentioned, we have chosen to produce only one child by crossover and we have taken a high mutation probability. This strategy favors molecular diversity but compels use of elite mode strategy.

Studies have shown that the speed of convergence can be modulated either by the population size or by

the range size during fitness scaling as has already been reported [14]. Fast average fitness convergence means that the selection pressure is too high (too large a range or too small a population size) and slow average fitness convergence means that the selection pressure is too weak (too small a range or too large a population size).

#### *Crossover and mutation probability*

It is established that the choice of crossover probability is dependent upon the problem to be solved and other factors such as population replacement [44]. In traditional problems, mutation rate is usually small [45] and is defined with regard to the candidate genes (Holland style genetic algorithms). Usually, mutation at a very low level (for example 1 bit, per 1000, per generation in binary implementations) is necessary to counter premature convergence. In chemical problems, this mutation rate is more variable since it takes a value from a few percent [24, 23] to 50% or 80% [14, 17, 46] application probability (in terms of candidates rather than genes (Davis style genetic algorithms [21])). Interestingly, the low and high mutation rate corresponds to two sorts of GA in these de novo design programs, the first encodes the molecule in a '1-dimensional' chromosome (association of amino acids to make a peptide for example) and the second encodes the molecule in a '2D' chromosome (any general organic structure which can be branched). The first problem consists of searching the right linear arrangement of predefined blocks (peptide). The second problem has many more available combinations

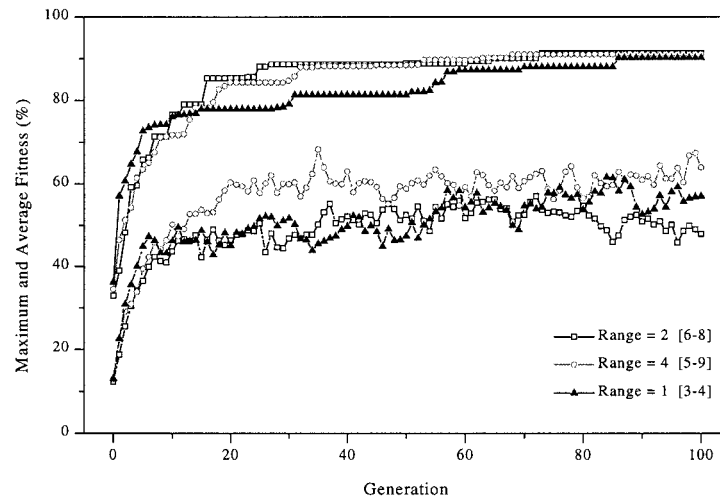


Figure 12. Influence of different range size (1, 2 and 4) on the average and maximum fitness.

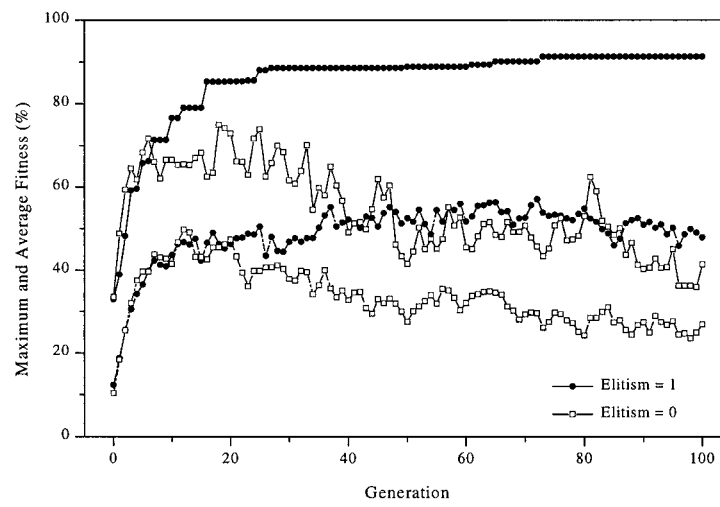


Figure 13. Average and maximum fitness variation with different elitism strategy.

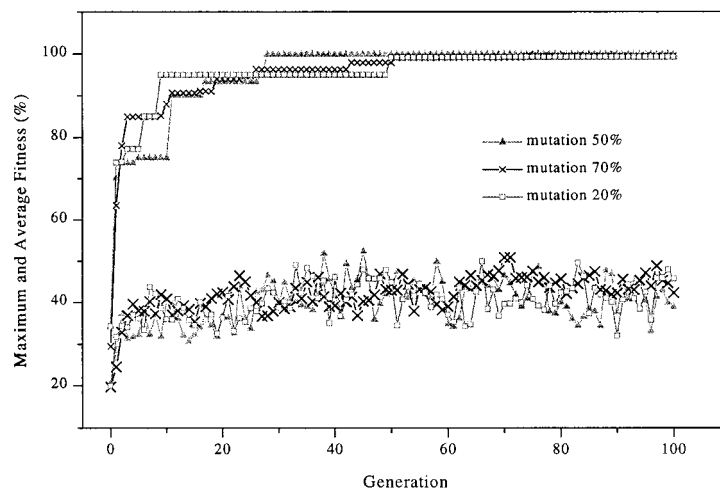


Figure 14. Average and maximum fitness variation with different mutation probabilities (20, 50 and 70%).

(the search space is larger since there are numerous cut-points (no predefined block) in the 2 dimensions). The former generally needs a low mutation rate and the second a high mutation rate. Sundaram et al. [46, 47] have shown that the crossover/mutation optimum rate is strongly dependent on the target sought: their results show that when a particular side-chain is required then high mutation rates give better results. Mutation is the only way to create a new substitution point on the skeleton of the candidate (replace a hydrogen by a fragment).

Crossovers act on a 2-dimensional chromosome since a branched group can be either replaced or be a fragment to combine. This suggests a more difficult combinatorial problem than in a '1D' chromosome and a sub-optimal efficiency of this operator is expected. These operators are too disruptive, on average, since they have a high probability of breaking the right combinations (randomly chosen cut-points can be any single bond either in the skeleton or within parentheses). In traditional GA, disruption is less probable since cut-points are generally set between well formatted genes representing a variable/parameter of the problem to solve (torsion angles in a molecular mechanics calculation [48], amino acids in peptide). Moreover, in our problem, functions which rewrite SMILES strings in 'palindromic' form or in order to limit high molecular weight fragment in parentheses hinder stable (or canonic) strings which would reduce disruptions by keeping features in their right initial order. Nevertheless, these functions are necessary to access the entire search space (fragments of the skeleton must be able to be linked by their right and/or left sides if valency is correct).

Mutation is useful to prevent the algorithm from becoming trapped in a local optimum. But it has, theoretically, a dramatic effect since it can delete desirable features. Figure 14 shows the fitness (average and maximum) evolution with different probability of mutation (pm). Variations are small and no conclusions can be drawn on these results (these mutation probabilities have no effect on the present optimization). Nevertheless, Sammon projections have been calculated in order to evaluate the search space explored during a run (Figure 15a, b and c). The diversity measure, based on 5 properties (length, lipophilicity, volume, radius of gyration, and dipole moment), is projected in a 2-dimensional plot. It shows differences between mutation probability in terms of areas. In each case, this surface is relatively small and thus shows the good exploitation property of the GA (su-

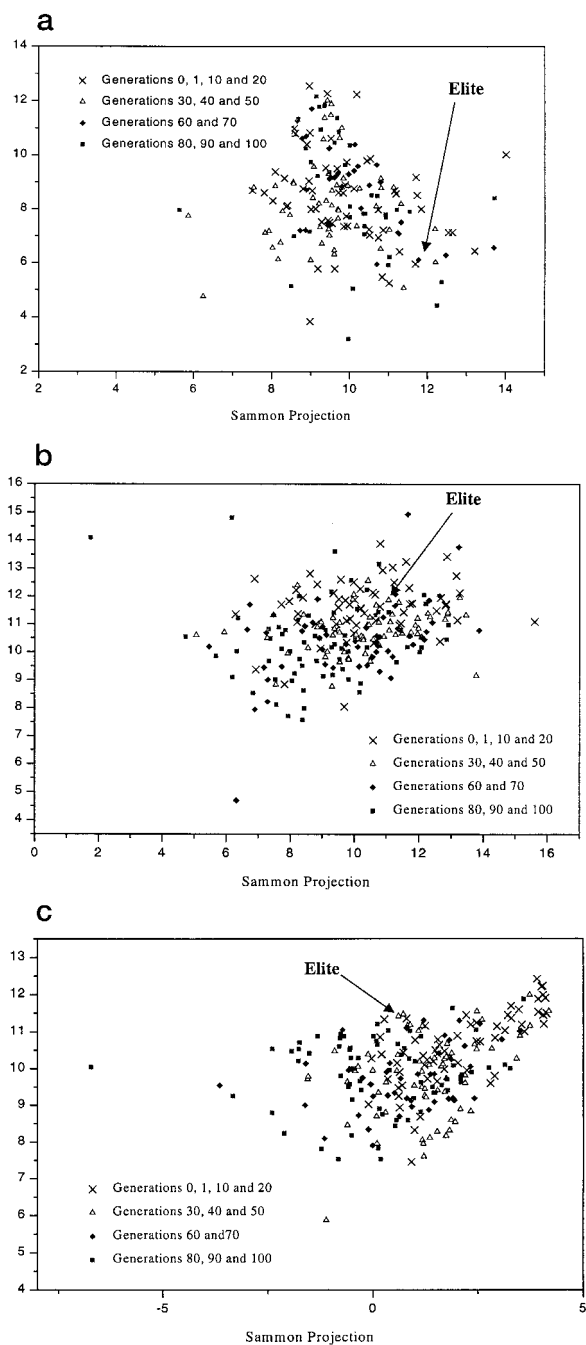


Figure 15. The diversity measure, or evaluation of the search explored during a run, is provided by a Sammon projection with 20% (a), 50% (b) and 80% (c) mutation rate. Points are molecules sampled every 10th generation and grouped in four periods. In (c), 2 molecules are omitted since they are far from the cluster. Figures show about 14% of the unique structure generated by GA. Unique structures are about 47% of the generated molecules in the retinoid fitness optimization since theoretically, 3901 molecules are produced (100 generations with a population size of 40 candidates and with one elite strategy). The elite is also indicated. Lastly, about 3–4% of the 3901 molecules are not 'viable' since they are neither converted by Corina nor optimized by Insight.

perposition of points). Differences come from the common area between the four groups of snapshots which decreases with mutation probability. With high mutation probability, area or explored space increases along a run. But, we also point out that the search space explored is relatively small and this ensures that any structure contains substructure(s) of the elite. Nevertheless, it does not involve more unique structures and no correlation can be made between the number of adapted structures (>80%) and the mutation probability. In our 'chemical type' implementation it is important to emphasize that mutation is the only source of functional novelty. Indeed, contrary to GA where genes represent numeric coding of parameters, in our GA, crossover is unable to cause the emergence of novel functionality: combining methane and benzene will never produce a hydroxyl group, while combining two numeric values will give novel parameter values (continuity of values).

Finally, we noted that distinction between one-point-crossover and mutation by inserting a fragment (or an Amino Acid) is tight since they frequently substitute fragments by an equivalent molecular weight fragment. In fact, we work with a pseudo-population larger than the initial fixed size. The effect of high mutation rate is to allow large structural jumps in each generation, while elite mode promotes generation-to-generation stability. The resultant evolution is highly accelerated, at the cost of losing some fine tuning of the population.

#### *Adaptation of crossover and mutation probability*

The adaptation strategy, which improves or penalizes operators according to the gain they provide, can be used at operator type level (modulation of the probabilities  $p_m$  and  $p_c$ ) or at operator subtype level (modulation of the probability of each of the 15 breeding operators).

In the last adaptation level, studies have shown that the more an operator is selected the faster it is penalized and reaches 1% probability. This means that, on average, operators destroy good features since their benefits are less than 50% of child which is more fit than its parent(s). There are 13 mutation operators for only 2 crossover types, thus mutations are always quite frequent. Finally, operators are efficient enough to improve children with respect to parents in order to achieve optimization.

Optimization made under an adaptative strategy modulates the probability  $p_c$  and  $p_m$  rather than the

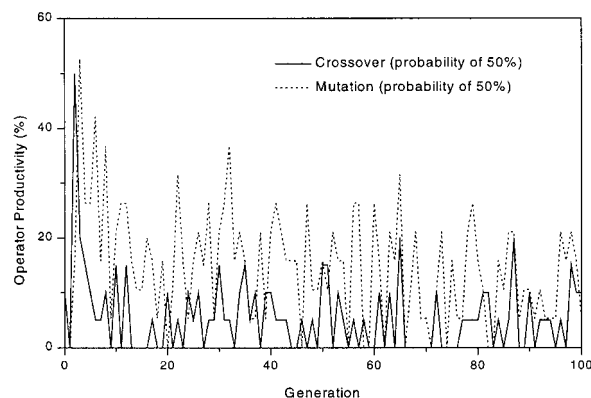


Figure 16. Operator productivity (%) along generation. Percentage of children which have been improved compared to parent(s). This has been obtained during optimization with a population size of 40, and one elite mode and with 50% mutation probability. The breeding process involves 20 crossovers and 19 mutations at each generation.

probability of each of the 15 breeding operators. In this case, the penalization or increase of probability is made according to the mutation effects (on average). Then, crossover probability rapidly reaches the maximum 90% and mutation probability reaches the minimum 10%. These optimizations generally lead to sub-optimal solutions. Such adaptation too frequently involves crossovers which have been shown to be less effective than mutation operators. Indeed, study of the operator benefits, or productivity in percentage of child improvement compared to parents, at each generation in a fixed strategy problem has shown that mutations are generally more productive (Figure 16 for the salicylic acid study with 50% crossover probability). Inverted trends are observed in traditional GA results [44].

It is obvious that the adaptation strategies do not improve optimization performance. The systematic probability evaluation at each generation provides flawed feedback, perhaps due to the noise in the operator benefits (due to inconsistent productivity even in the fixed strategy (Figure 16)). Adaptation happens too late in the operator modulation strategy (operator probability increases after its benefits). Moreover, we consider that the productivity threshold of 50% involved in the penalization or increasing of operator probability is too high. An index of 20% of improvement would be more appropriate to modulate probability (referred to productivity in Figure 16). Therefore, adaptation is not necessarily a good thing in itself [44] and in our case it is more powerful to use a fixed strategy (crossover probability is fixed af-

ter some trial runs) combined with dynamic mutations (mutations which occur in a foreseeable manner; this will be discussed in the Conclusions).

#### Salicylic acid similarity

Best results have been obtained with a population of 10, combined with an elite mode strategy and a fitness scaling by a range [2–10]. The initial population was composed of 10 methanes. Many salicylic acid isomers with scores of 100% have been obtained; these are a combination of a benzene ring, one carbon, two oxygens and a carboxyl function. Such constraints are not relevant enough to distinguish between isomers. There exist many available combinations with di-substituted benzene but not all of them match the pharmacophore constraint. The evolution of salicylic acid over 25 generations is given in Appendix II. Scores and associated weights are given. The optimization begins with 10 methanes with a score of 1% (viable but unsuitable structures). The first breeding step introduces branched groups since molecular weights must increase. This is a necessary but not sufficient step (ramifications are small fragments). The first benzene appears in the second generation by making a ring out of the previous ramification input. This optimization does not set fragment insertion mutation nor amino acid insertion mutation; thus a ring must be created. The first ring is unsaturated enough that it gives a score of 30%. Generation 3 is composed of 6 and 5 rings. The elite contains the exact number of elements and is sufficiently unsaturated. Generation 13 shows the first combination of a benzene and an ester function. Following generations promote the diffusion of the substructure of the elite. Generation 22 shows two structures with equal fitness of 69.6%. Unfortunately, the elite which will be kept is the first which appeared (meta substitution) instead of the second (the ortho and the expected substitution). This structure has not survived. Only one carbon mutation would give the salicylic acid (exact number of oxygens and matching of the pharmacophore key). Nevertheless, generation 25 has given the right structure. The salicylic acid has been created by an insertion of a hydroxyl function in a structure similar to the generation 13 elite.

These studies have also been performed in order to tackle dynamic mutations as a kind of cooling schedule. Thus, the first generation focuses search in space where molecular weight and unsaturation are, on average, satisfied. To do this, ramification and ring making operators are frequently applied. Small mu-

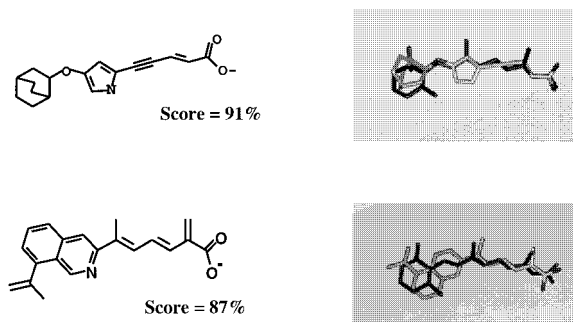


Figure 17. Some examples of retinoid analogues using retinoid fitness. Structures have been docked in the RAR $\alpha$  and then superimposed on a bound conformation of ATRA (all trans retinoic acid).

tation operators such as mutate an element are little used (probability equal to 1%). This prerequisite step is to find an adapted skeleton before fleshing it out by heteroatoms and functions.

#### Retinoid analogues

Best results have been obtained with a population of 40, combined with an elite mode strategy and a fitness scaling by a range [5–9]. The initial population was randomly initialized. Results show that LEA is able to find retinoid structures, either used to derive the model or not used to derive the model but similar to a compound already present in the Galderma database. Two novel structures are given in Figure 17. They have been docked in the receptor RAR $\alpha$  and superimposed on the bound conformation of all trans retinoic acid (ATRA). The receptor model has been obtained by homology with the RAR $\gamma$  crystallographic data. Proposed molecules are sometimes difficult to synthesize and require modifications. This last step is conducted by a medicinal chemist and new molecules are 're'-evaluated. In the retinoid project, some structures have been chosen and will be synthesized and tested.

#### Conclusions

This article describes an implementation of a genetic algorithm for tackling the de novo design problem. A parameterization study shows that GA is rather robust since small changes in parameters such as population size or range scale should have little effect on the performance. Major differences have been seen with mutation probability. Interestingly, best results have been obtained with high probability mutation

since high probability crossover hinders average and maximum fitness convergence or leads to premature convergence. In such cases, we have shown that the elite mode strategy is unavoidable. These properties are somewhat similar to those used by Evolution Strategy (ES) [6, 49]. Generally, ES solves real-valued function optimization problems. Typically, it operates on the real values themselves (phenotypic) rather than any coding of the real values as is often done in GAs. ES and GA both perform some form of recombination (conversely to Evolutionary Programming [5]). ES also uses deterministic selection. Usually, such methods use a self-adaptive method for determining the appropriate mutation to use. Mutations are individually adjusted (the so-called dynamic mutation) and consist of the addition of a random number taken from a Gaussian distribution. Such continuous real-valued mutations are impossible in structure design but we can devise a system in which each chromosome will be associated with an appropriate 'mutation panel' based on its fitness rather than a standard deviation of the Gaussian. This strategy will allow an internal control of the extent of the modification that will occur on the structure. For example, mutation by insertion of a benzene has more effect on a structure than an atom type mutation. The former is useful at the beginning of the search (when score<sub>i</sub> is poor) and the latter may be important at the end of the optimization (high score<sub>i</sub>). Such a method is currently being examined in order to devise another de novo design program.

Our results show that LEA can be used for finding novel analogues (each run generates a different family of solutions) but results strongly depend on the quality of the model. The most successful GA implementations use hybrid GA combinations with other techniques [50]. It can be obtained by a Classical Neighborhood Search (a local search descent procedure) that works on one solution at a time. Future development will concentrate on implementing this local search on the best structures at the end of run.

In a more general manner, LEA can be applied to evolve diverse kinds of molecules, small organic compounds, peptides or polymers (with changes on operators). The program is also independent of the genetic algorithm framework and thus can be used to optimize structures in QSAR or QSPR studies or binding energy if a docking program is used in a batch mode manner.

## Acknowledgements

The authors thank William Pilgrim for reading and commenting upon the manuscript.

## References

1. Walters, W.P., Stahl, M.T. and Murcko, M.A., *Drug Des. Today*, 3 (1998) 160.
2. Ghose, A.K., Viswanadhan, V.N. and Wendoloski, J.J., *J. Comb. Chem.*, 1 (1999) 55.
3. Lewis, R.A., Mason, J.S. and McLay, I.M., *J. Chem. Inf. Comput. Sci.*, 37 (1997) 599.
4. Darwin, C., *The Origin of Species*, Dent Gordon, London, 1973.
5. Fogel, L.J., Owens, A.J. and Walsh, M.J., *Artificial Intelligence through Simulated Evolution*, John Wiley & Sons, New York, NY, 1966.
6. Rechenberg, I., *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment Transl., No 1122, B.F. Toms, Transl. (Ministry of Aviation, Royal Aircraft Establishment), Farnborough, Hants., U.K., 1965.
7. Clark, D.E. and Westhead, D.R., *J. Comput.-Aided Mol. Design*, 10 (1996) 337.
8. Hibbert, D.B., *Chemometr. Intell. Lab. Syst.*, 31 (1996) R5.
9. Schneider, G. and Schrödl, W., *Proc. Natl. Acad. Sci. USA*, 95 (1998) 12179.
10. Parrill, A.L., *Drug Des. Today*, 1 (1996) 12.
11. Holland, J., *Adaptation in Natural and Artificial Systems* (second edition), M.I.T. Press, Cambridge, MA, 1992.
12. Forrest, S., *Science*, 261 (1993) 872.
13. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA, 1989.
14. Glen, R.C. and Payne, A.W.R., *J. Comput.-Aided Mol. Design*, 9 (1995) 181.
15. LeapFrog, Tripos Inc., St Louis, MO, U.S.A.
16. Westhead, D.R., Clark, D.E., Frenkel, D., Li, J., Murray, C.W., Robson, B. and Waszkowycz, B., *J. Comput.-Aided Mol. Design*, 9 (1995) 139.
17. Blaney, J.M., Dixon, J.S. and Weininger, D., *Molecular Graphics Society Meeting on Binding Sites: Characterising and Satisfying Steric and Chemical Restraints*, York, U.K., March 1993. Weininger, D., WO95/01606.
18. Weininger, D., *J. Chem. Inf. Comput. Sci.*, 30 (1990) 237.
19. Bemis, G.W. and Murcko, M.A., *J. Med. Chem.*, 39 (1996) 2887.
20. Lucasius, C.B. and Kateman, G., *Chemometr. Intell. Lab. Syst.*, 19 (1993) 1.
21. Davis, L., In J.D. Schaffer (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms and Their Applications*, San Mateo, CA, Morgan Kaufmann, 1989, pp. 61–69.
22. Rishton, G.M., *Drug Des. Today*, 2 (1997) 382.
23. Weber, L., Wallbaum, S., Broger, C. and Gubernator, K., *Angew. Chem. Int. Ed. Engl.*, 34 (1995) 2280.
24. Singh, J., Ator, M.A., Jaeger, E.P., Allen, M.P., Whipple, D.A., Solowej, J.E., Chowdhary, S. and Treasurywala, A.M., *J. Am. Chem. Soc.*, 118 (1996) 1669.
25. Grassy, G., Fagart, J., Calas, B., Adenot, M., Rafestin-Obelin, M.E. and Auzou, G., *Eur. J. Med. Chem.*, 32 (1997) 869.
26. Grassy, G., Trappe, P., Bompart, J., Calas, B. and Auzou, G., *J. Mol. Graphics*, 13 (1995) 356.

27. Grassy, G., Yasri, A., Buelow, R., Kaczorek, M. and Calas, B., *Actualités de Chimie Thérapeutique*, 24, 1998.
28. Grassy, G., Calas, B., Yasri, A., Lahana, R., Woo, J., Lyer, S., Kaczorek, M., Floc'h, R. and Buelow, R., *Nat. Biotech.*, 16 (1998) 748.
29. McFarland, J.W. and Gans, D.J., *J. Med. Chem.*, 29 (1986) 505.
30. Douguet, D., Thoreau, E. and Grassy, G., *Quant. Struct.-Act. Relat.*, 18 (1999) 107.
31. Chambon, P., *Faseb J.*, 10 (1996) 940.
32. Klaholz, B.P., Renaud, J.P., Mitschler, A., Zusi, C., Chambon, P., Gronemeyer, H. and Moras, D., *Nat. Struct. Biol.*, 5 (1998) 199.
33. Ostrowski, J., Roalsvig, T., Hammer, L., Marinier, A., Starrett, J.E. Jr, Yu, K.-L. and Reczek, P.R., *J. Biol. Chem.*, 273 (1998) 6, 3490.
34. DeJong, K., *Machine Learning*, 3 (1988) 121.
35. Sammon, J.W., *IEEE Trans. Comput.*, C-18 (1969) 401.
36. Wall, L., Christiansen, T. and Schwartz, R.L., *Programming Perl*, O'Reilly.
37. RasMol v2.5, Sayle, R., *Biomolecular Structure*, Glaxo Research and Development, Greenford, Middlesex, U.K., 1994.
38. InsightII, Molecular Simulations Inc., San Diego, CA.
39. VAMP, Oxford Molecular Ltd, Oxford, U.K.
40. CORINA, Oxford Molecular Ltd, Oxford, U.K.
41. Sybyl, Tripos Associates, St. Louis, MO.
42. Silla, E., Tunon, I. and Pascual-Ahuir, J.L., *J. Comput. Chem.*, 12 (1991) 1077. QCPE #554.
43. Suzuki, T., *J. Comput.-Aided Mol. Design*, 4 (1990) 155. QCPE #608. CHEMICALC-2.
44. Tuson, A. and Ross, P., *Evol. Comput.*, 6 (1998) 161.
45. Hibbert, D.B., *Chemometr. Intell. Lab. Syst.*, 19 (1993) 277.
46. Venkatasubramanian, V., Chan, K. and Caruthers, J.M., *J. Chem. Inf. Comput. Sci.*, 35 (1995) 188.
47. Sundaram, A. and Venkatasubramanian, V., *J. Chem. Inf. Comput. Sci.*, 38 (1998) 1177.
48. Brodmeier, T. and Pretsch, E., *J. Comput. Chem.*, 15 (1994) 588.
49. Schwefel, H., *Evolution and Optimum Seeking*, Wiley, New York, NY, 1995.
50. Osman, I.H., *Operational Research Tutorial Papers*, Operational Research Society Press, Birmingham, U.K., 1995.

## Appendix I

Auto-correlation vectors (Vac3D); hydrogen are also included.

Auto-correlation vectors weighted by positive partial charge (Vac3D\_chgp)

Auto-correlation vectors weighted by negative partial charge (Vac3D\_chgn)

Auto-correlation vectors weighted by lipophilicity logP (Vac3D\_logP)

Lipophilicity LogP by Ghose & Crippen (atomic contribution method) or by Suzuki (fragmental contribution method)

Molar Refractivity calculated by Ghose & Crippen (atomic contribution method)

Molecular Weight

Formula Weight

Insaturation Index (ratio of number of bond on number of double and triple bonds)

Surfaces (Solvent Accessible Surface, Excluded Surface or Van der Waals Surface) calculated by GEPOL.

Volume by GEPOL or by VAMP

Length (longest distance between atoms in molecule)

Inertial axis along  $x$ ,  $y$  and  $z$

Radius of gyration (globularity index)

HOF, Heat of Formation obtained by VAMP calculation

Lowest Unoccupied Molecular Orbital (LUMO) obtained by VAMP calculation

Highest Occupied Molecular Orbital (HOMO) obtained by VAMP calculation

GAP (LUMO-HOMO), Stability index obtained by VAMP calculation

Dipole Moment (in Debye units) obtained by VAMP or Discover optimization

Pattern research (generally function like keton, hydroxyl, carboxylate, ...) Pharmacophore research (AH: hydrogen bond acceptor, DH: hydrogen bond donor, B: hydrogen bond donor and acceptor).



## Appendix II

The evolution of salicylic acid analogues over 25 generations starting from methane

Generation 1

1	Score = 1% P : 6.04	2	Score = 1% P : 6.04
<chem>CH4</chem>		<chem>OH2</chem>	
3	Score = 1% P : 6.04	4	Score = 1% P : 6.04
<chem>OH2</chem>		<chem>OH2</chem>	
5	Score = 1% P : 6.04	6	Score = 1% P : 6.04
<chem>OH2</chem>		<chem>SH2</chem>	
7	Score = 1% P : 6.04	8	Score = 1% P : 6.04
<chem>NH3</chem>		<chem>CCCC</chem>	
9	Score = 1% P : 6.04	10	Score = 1% P : 6.04
<chem>CCCC</chem>		<chem>CCCC</chem>	

Generation 2

1	Score = 30% P : 14.24	2	Score = 30% P : 14.24
<chem>c1ccccc1</chem>		<chem>c1ccccc1</chem>	
3	Score = 1% P : 4.0	4	Score = 1% P : 4.0
<chem>CH4</chem>		<chem>C1=CC=CC=C1</chem>	
5	Score = 1% P : 4.0	6	Score = 1% P : 4.0
<chem>CCCC=CC</chem>		<chem>CC=CC</chem>	
7	Score = 1% P : 4.0	8	Score = 1% P : 4.0
<chem>CCCCC</chem>		<chem>CC=CC</chem>	
9	Score = 1% P : 4.0	10	Score = 1% P : 4.0
<chem>CCCC</chem>		<chem>C1=CC=CC=C1</chem>	

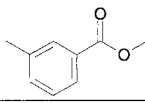
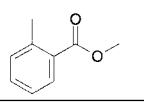
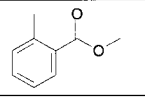
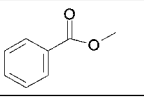
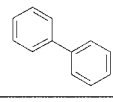
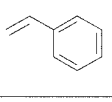
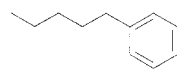
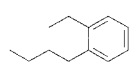
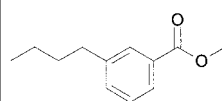
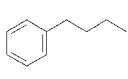
Generation 3

1	Score = 51.3% P : 13.7	2	Score = 40% P : 10.54
<chem>c1ccc(cc1)-c2ccccc2</chem>		<chem>c1ccc(cc1)CC</chem>	
3	Score = 30% P : 7.32	4	Score = 29.6% P : 7.64
<chem>c1ccc(cc1)C</chem>		<chem>CCCCC</chem>	
5	Score = 28.6% P : 7.35	6	Score = 28.6% P : 7.35
<chem>c1ccc(cc1)CCCCC</chem>		<chem>c1ccc(cc1)CCCC</chem>	
7	Score = 13% P : 2.98	8	Score = 10% P : 2.12
<chem>CCCCC</chem>		<chem>CCCC</chem>	
9	Score = 10% P : 2.12	10	Score = 1% P : 1
<chem>CCCCC</chem>		<chem>C1=CC=CC=C1</chem>	

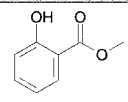
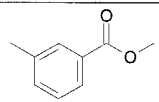
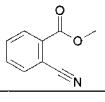
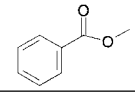
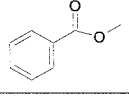
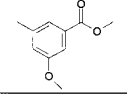
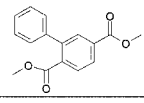
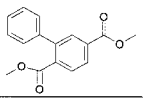
Generation 13

1	Score = 51.7% P : 12.85	2	Score = 51.3% P : 12.76
<chem>CC(=O)c1ccccc1</chem>		<chem>c1ccc(cc1)-c2ccccc2</chem>	
3	Score = 48% P : 12.0	4	Score = 30% P : 7.9
<chem>c1ccc(cc1)-c2ccccc2</chem>		<chem>c1ccc(cc1)-c2ccccc2</chem>	
5	Score = 30% P : 7.9	6	Score = 10% P : 3.34
<chem>c1ccc(cc1)C</chem>		<chem>CC(=O)c1ccc(cc1)C=C</chem>	
7	Score = 10% P : 3.34	8	Score = 6.2% P : 2.49
<chem>CCCCC(=O)OC</chem>		<chem>c1ccc(cc1)-c2ccccc2</chem>	
9	Score = 1% P : 1.2	10	Score = 1% P : 1.2
<chem>CCCCC</chem>		<chem>c1ccc(cc1)-c2ccccc2</chem>	

Generation 22

1	Score = 69.6% P :12.71	2	Score = 69.6% P :12.71
			
3	Score = 69.6% P :12.71	4	Score = 51.7% P :9.16
			
5	Score = 48% P :8.43	6	Score = 40% P :6.83
			
7	Score = 28.6% P :4.57	8	Score = 13% P :1.49
			
9	Score = 10% P :1	10	Score = 1.73% P :1
			

Generation 25

1	Score = 100% P :18.3	2	Score = 69.6% P :11.4
			
3	Score = 54.7% P :8.0	4	Score = 51.7% P :7.33
			
5	Score = 51.7% P :7.33	6	Score = 44.9% P :5.79
			
7	Score = 40% P :4.66	8	Score = 40% P :4.66
			
9	Score = 40% P :4.66	10	Score = 10% P :1
