An Empirical Study on the Influence of Genetic Operators for Molecular Docking Optimization

Jorge Tavares, Nouredine Melab, El-Ghazali Talbi

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

An Empirical Study on the Influence of Genetic Operators for Molecular Docking Optimization

Jorge Tavares — Nouredine Melab — El-Ghazali Talbi

N° 6660

September 2008

Thème NUM

ISSN 0249-6399 ISRN INRIA/RR--6660--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
LILLE - NORD EUROPE

An Empirical Study on the Influence of Genetic Operators for Molecular Docking Optimization

Jorge Tavares, Nouredine Melab, El-Ghazali Talbi

Thème NUM — Systèmes numériques
Équipe-Projet DOLPHIN

Rapport de recherche n° 6660 — September 2008 — 40 pages

Abstract: Evolutionary approaches to molecular docking typically use a real-value encoding with standard genetic operators. Mutation is usually based on Gaussian and Cauchy distributions whereas for crossover no special considerations are made. The choice of operators is important for an efficient algorithm for this problem. We investigate their effect by performing a locality, heritability and heuristic bias analysis.

Our investigation focus on encoding properties and how the different variation operators affect them. It is important to understand the behavior and influence of these components in order to design new and more efficient evolutionary algorithms for the molecular docking problem.

Results confirm that high locality is important and explain the behavior of different crossover and mutation operators. In addition, the heritability and heuristic bias study provides some insights in how the different crossover operators perform. Optimization runs in different instances of the problem support the analysis findings. The performance and behavior of the variation operators are consistent on several molecules.

Key-words: representation, genetic operators, locality, heritability, heuristic bias, mutation, crossover, molecular docking, optimization.

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne
40, avenue Halley, 59650 Villeneuve d'Ascq
Téléphone : +33 3 59 57 78 00 — Télécopie : +33 3 59 57 78 50

An Empirical Study on the Influence of Genetic Operators for Molecular Docking Optimization

Résumé : Evolutionary approaches to molecular docking typically use a real-value encoding with standard genetic operators. Mutation is usually based on Gaussian and Cauchy distributions whereas for crossover no special considerations are made. The choice of operators is important for an efficient algorithm for this problem. We investigate their effect by performing a locality, heritability and heuristic bias analysis.

Our investigation focus on encoding properties and how the different variation operators affect them. It is important to understand the behavior and influence of these components in order to design new and more efficient evolutionary algorithms for the molecular docking problem.

Results confirm that high locality is important and explain the behavior of different crossover and mutation operators. In addition, the heritability and heuristic bias study provides some insights in how the different crossover operators perform. Optimization runs in different instances of the problem support the analysis findings. The performance and behavior of the variation operators are consistent on several molecules.

Mots-clés : representation, genetic operators, locality, heritability, heuristic bias, mutation, crossover, molecular docking, optimization.

Contents

1 Introduction

The aim of molecular docking [1] is to design pharmaceuticals by identifying potential drug candidates targeted against proteins. In general, the effect of a drug is dependent on the formation of a complex between a small molecule, the ligand, and a macromolecule, the protein, whose malfunction is resulting in a disease. The biological function of a protein is connected to its three-dimensional structure. As such, altering the structure modifies the protein functionality. The design of new pharmaceuticals must correctly predict and optimize the specific interactions between both molecules of the formed complex. This process is very expensive and time-consuming. The potential drug candidates can be found by using a docking algorithm. The algorithm will try to identify the bound conformation of the ligand to the active site of a protein. The active site is the region where the natural substrate binds, i.e., the specific molecule an enzyme acts upon. The structure and the spatial arrangement of the atoms at the active site, complements the shape and properties of the substrate. In this way, it catalyzes a particular reaction. Therefore, drug discovery aims to find ligands which bind stronger to a given protein target than the natural substrate.

Molecular docking is an energy minimization search problem with the aim to find the best ligand conformation and orientation relative to the active site of a target protein [2]. The docking problem can be very difficult since the relative orientation and conformations of two molecules must be considered. Typically, the receptor (usually a protein) is fixed in a three-dimensional coordinate system. By contrast, the ligand can be repositioned and rotated. In case that both receptor and ligand are allowed to be flexible, the problem difficulty increases. As such, the problem is classified, by increasing complexity, into the ensuing categories: rigid-structure docking (both molecules are rigid); rigid protein and flexible ligand; flexible protein and rigid ligand; and, both molecules are flexible. With both molecules flexible, usually the active site of the protein and the ligand, the problem becomes harder. In fact, a higher degree of flexibility implies a considerable increase of the search space size.

For the past years, numerous molecular docking methods have been proposed using different techniques, e.g., incremental construction algorithms, stochastic algorithms and molecular dynamics. For more detailed descriptions, we refer the reader to several review studies [3, 2]. Evolutionary algorithms have recently become one of the dominant search techniques for docking methods and proved to be very successful [2]. Although several applications exist, no comprehensive set of studies could be found *to understand why* these algorithms and their components are successful. To the best of our knowledge, the only attempt was made in [4] where several parameters (e.g., population size) and some genetic operators are empirically investigated. When designing an evolutionary approach for this problem, to make it efficient is important to understand its components behavior and effects. Locality is an important requisite to ensure the efficiency of search and it has been widely studied by the evolutionary computation community [5, 6, 7]. In general terms, this property indicates that small variations in the genotype space, usually originated by mutation, imply small variations in the phenotype space [5]. A locally strong search algorithm is able to efficiently explore the neighborhood of the current solutions. When this condition is not satisfied, the exploration performed by the algorithm is inefficient and, in a worse case scenario, tends to resemble random search. Heritability refers to the

interaction between a representation and a crossover operator [7, 8]. A crossover operator should be able to combine meaningful substructures from the parents in order to produce descendants. This kind of behavior allows the exploitation of successful substructures from the parents. Moreover, it should mix the parents' phenotypic properties to generate new phenotypes in a creative manner. A useful crossover operator must have enough heritability to not cripple the evolutionary search process. Heuristic bias is the property related to the mapping between a genotype and a phenotype [9]. An evolutionary algorithm explores a search space defined by its representation and genetic operators. Usually this space is indirectly connected to the phenotype space and thus, the efficiency of the process is influenced by the mapping between the elements of both spaces. The use of heuristics or other means of problem knowledge can affect the distribution of phenotypes, introducing, or not, a bias towards fitter phenotypes.

The main objective of this paper is to perform an empirical analysis on the representation and variation operators using the evolutionary algorithm model [10, 2] that is usually adopted for molecular docking optimization. Locality, heritability and heuristic bias measures are adopted from the framework proposed by [7] and extended by [11] to deal with real-valued encodings. One distance measure suitable for the selected representation phenotype is applied. The present study concentrates on the questions: do Gaussian and Cauchy mutation operators have a different effect on phenotypes? Which type of mutation and crossover operator is more suitable for evolutionary approaches to molecular docking? Can we explain results from optimization according to the encoding properties? What is the impact of different genetic operators in a representation? Our main research focus is the study of representation properties and the effects of variation operators. We expect to answer these questions by investigating the influence of the operators on locality, heritability and heuristic bias. The presented work, to the best of our knowledge, is the first wider study that includes an analysis of representation and operators in the context of molecular docking. Some of our initial results concerning locality and mutation operators can be found here [12].

Results in this paper allow us to gain some insights about the degree of locality, heritability induced by different mutation operators and their heuristic bias. The search space is highly multimodal and its shape is influenced by the size, shape and topology of the ligand and the active site being docked [4]. As a consequence of this, even small modifications performed by genetic operators in the structure of an individual lead to large phenotypic changes. An evolutionary algorithm operating on its own is unable to deal with these difficulties. Thus, it is important to know how encoding properties relate to mutation and crossover operators commonly used in evolutionary algorithms for molecular docking. Furthermore, understanding the role played by each algorithm's component may provide useful insights for future applications of evolutionary algorithms to this problem.

The rest of the paper is structured as follows. Section 2 contains an overview of the evolutionary algorithms' components used in our experimentation. Next, in section 3 we present the framework used in our analysis. In section 4 we present the empirical analysis on the influence of genetic operators and respective discussion. Finally, in section 5 the main conclusions of our work are described.

2 Evolutionary Algorithms for Molecular Docking Optimization

Evolutionary algorithms applied to molecular docking can be found since 1993 [13]. This first initial approach used a simple genetic algorithm. A binary encoding was used to represent the torsion angles for rotatable bonds in the ligand. Later, similar attempts were made using simple genetic algorithms with binary representations [14, 15]. The first approach with a real-value encoding was proposed by [16]. The algorithm employed was based on Evolutionary Programming and used a self-adaptive scheme to evolve the variance used in the mutation operator. Several evolutionary algorithms using real-valued encodings were proposed. The approaches based on this type of encoding were shown to achieve better results, being more efficient and accurate. A comprehensive review of these efforts, including an outline state-of-the art applications, can be found in [17, 2].

One of the most important approaches to molecular docking is the software suite referred to as *AutoDock*. It started by using simulated annealing as its search method but later on it moved to evolutionary algorithms [10]. This approach is a conformational search method which uses an approximate physical model to evaluate possible protein-ligand conformations. It incorporates flexibility by allowing the ligand to change its conformation during the docking simulation. In addition, pairwise interactions between atoms are pre-calculated, considerably speeding up the docking simulation. To search the space of possible protein-ligand conformations, the approach uses an evolutionary algorithm with a local search method. When this method is applied, the genotype of the individuals is replaced with the new best solution found. This process is usually referred to as Lamarckian evolution.

2.1 Representation and Evaluation

During the docking process the protein remains rigid whilst the ligand is flexible. In this case, an individual represents only the ligand. A genotype of a candidate solution is encoded by a vector of real-valued numbers which represent the ligand's translation, orientation and torsion angles [10]. Cartesian coordinates represent the ligand translation, three variables in the vector, whereas four variables defining a quaternion represent the ligand orientation. A quaternion can be considered to be a vector (x, y, z) which specifies an axis of rotation with an angle θ of rotation for this axis. For each flexible torsion angle one variable is used. The phenotype of a candidate solution is composed of the atomic coordinates that represent the three-dimensional structure of the ligand. Therefore, the representation is indirect since the atomic structure of the ligand is built from the translation and orientation coordinates in the ligand crystal structure with the application of the torsion angles. In this work, besides the real-valued encoding we also use a binary one. The semantic of the representation is the same as before. The only difference is that the real-valued numbers are encoded in a binary string. This representation is used only for comparative purposes in some parts of our analysis.

To evaluate each individual an energy evaluation function is used. The fitness for each candidate solution is given by the sum of the intermolecular interaction

energy between the ligand and the protein, and the intramolecular energy that arises from the ligand itself [10]. An empirical free energy potential composed of five terms is used. The first three terms are pairwise interatomic potentials that account for weal long-range attractive forces and short-range electrostatic repulsive forces. The fourth term measures the unfavorable entropy of a ligand binding due to the restriction of conformational degrees of freedom. The fifth and last term uses a desolvation measure. Further details of the energy terms and how the potential is derived can be found in [10].

2.2 Genetic Operators

Common crossover and mutation operators are applied on the population. In *AutoDock* a standard two-point crossover is used. Cut points only occur between related genes, i.e., separating translational values, orientation values and rotation torsion angles into separate blocks. This is done to avoid disruption of useful parts of the solution [10]. However, for real-valued encodings it is recommend that operators designed to deal with this type of encoding are used. In this work, we analyze several common crossover operators, such as Simple Arithmetical crossover, Whole Arithmetical crossover, Discrete crossover, Simulated Binary Crossover and Blend-α crossover. Since these are typical genetic operators, we refer the reader to the following literature for a description of their operation [18, 19, 20]. In the case of binary encoding, we use the classic 1-Point and Uniform crossover [18].

Since the encoding is a real-valued vector, mutation is performed by using evolutionary strategies based operators. The genetic operator acts in the following way: when undergoing mutation, the new value for a gene x' is obtained from the old value x by adding a random real number sampled from a distribution $U(0,1)$:

$$x' = x + \sigma \times U(0,1) \tag{1}$$

The common distribution used for $U(0,1)$ is the standard Gaussian distribution, $N(0,1)$. In spite of that, the *AutoDock* approach replaces the Gaussian distribution with a Cauchy distribution:

$$C(x,\alpha,\beta) = \frac{\beta}{\pi\beta^2 + (x-\alpha)^2} \tag{2}$$

where $\alpha \leq 0, \beta > 0, -\infty < x < +\infty$ (α and β are parameters that control the mean and spread of the distribution). The Cauchy distribution has a bias toward small variations. However, unlike the Gaussian distribution, it has thick tails which allows larger variations more frequently. Some evolutionary approaches use both distributions for mutation operators [4].

One important aspect is the value for the parameter σ. If it is set too low, exploitation overcomes exploration and if set too high *vice versa*. The value can be fixed or self-adapted (e.g., if an evolutionary strategy approach is used). In [4] are proposed annealing schemes to control σ as a function of time, i.e., the number of generations, scaled with 0.1:

$$A1 : \sigma(t) = \frac{1}{1+t} \tag{3}$$

$$A2 : \sigma(t) = \frac{1}{\sqrt{1+t}} \tag{4}$$

Only a fixed σ and the annealing schemes are considered for analysis. The parameter σ is set to a value of 0.1 which previous studies in computational chemistry problems have shown to be a good value [11]. In addition, preliminary tests in this problem with other values also helped us to pick this value. We also include in the analysis the simple uniform mutation operator. It works in the following way: when applied to a gene, it assigns a new random value according to the gene bounds, sampled from a standard uniform distribution. This operator serves as a comparison baseline. The mutation used with the binary representation is the typical bit-flip operator [18].

For our analysis, we adopt an experimental model which uses the main components from [10], since *AutoDock* serves as a basis for the large majority of evolutionary-inspired approaches, e.g.,[2, 21].

3 Analysis Framework

3.1 Related Work

Several techniques have been proposed to estimate and study the behavior of evolutionary algorithms and their components. In the literature we can find some studies related to the properties of locality, heritability and heuristic bias, as well as the relationships between the different spaces that characterize evolutionary search. In this section, we highlight the most relevant ones.

Evolutionary search can be represented by three spaces: the *search space*, the *phenotype space* and the *fitness space*. The fitness space reflects the solution quality whilst the search space is made of the candidate solutions. The set of all possible genotypes is denominated *genotype space* and it is equivalent to the search space. This equivalence is possible because variation operators work in the genotype space and an evolutionary algorithm searches for genotypes that decode into phenotypes with high fitness. In [22], correlation coefficients for the fitness values of solutions were studied in between the application of variation operators. Locality in this investigation was characterized by the relation between the search space and the fitness space. It was concluded that a dependency between the correlation coefficients and performance existed and was strong.

The concept of *fitness landscape*, introduced by [23] to demonstrate the dynamics of biological evolutionary optimization, has been useful for the analysis and understanding of evolutionary algorithm's behavior. Several measures have been proposed for this task. In [24, 25], fitness distance correlation is presented as a way to determine the relation between fitness and distance to the optimum. If fitness values increase as the distance to the optimum decreases, then search is expected to be easy. An evolutionary algorithm can be seen as navigating a landscape in order to find the highest peak. Higher points in the search space correspond to solutions with higher fitness[1].

An alternative way to analyze the fitness landscape is to determine its ruggedness. In [26], it is proposed the adoption of autocorrelation functions to measure the correlation of all points in the search space at a given distance. The structure of a fitness landscape can be examined by measuring the degree of correlation between points on the landscape. The degree of correlation depends on the difference between the fitness values of the points. Smoother landscapes are highly correlated, making the search for an evolutionary algorithm easier. This is the result of similar fitness values. If the difference of fitness values is higher, the landscape is less correlated, which implies a rugged landscape, thus being harder to search in it. In [27, 28] is described how fitness landscapes can be useful in the design of memetic algorithms. In [29, 30] the role of representation and heuristics in combinatorial optimization problems is investigated by means of fitness landscape analysis.

Conditions for strong causality are studied in [5]. A search process is said to be locally strong causal if small variations in the genotype space imply small variations in the phenotype space. Fitness variation is used to access distances in the phenotype space. A probabilistic causality condition is proposed and studied in two different optimization situations. The research leads to the conclusion that strong causality is essential, as it allows the control of small steps in the

[1] In a maximization problem. It is also trivial to apply to minimization problems.

phenotype space by small steps performed in the genotype space. In the above-mentioned investigation only mutation was used in the study.

In [31], the impact of representation on the search complexity is studied and three elements are identified as important to build a general theory of representations: redundancy, scaling of building blocks and distance distortion. A framework is described that uses these three elements as the basis for representations analysis and performance prediction for evolutionary algorithms. The performance of an evolutionary algorithm is determined by the expected quality of the solutions and the necessary time to achieve them. The framework theoretically describes how different types of encodings can affect the behavior and performance of an evolutionary algorithm by building a model that allows the prediction of how a specific representation can perform on different problems.

Raidl and Gottlieb [7] proposed an empirical framework to study locality, heritability and heuristic bias. The analysis of these properties is based on static measures which are applied to randomly generated individuals. The measures help to quantify the distance between genotypes and how they are related in similarity to their corresponding phenotypes. This model is useful to study how a representation and associated variation operators are related and how their relationship affects the performance of the evolutionary algorithm. The framework can also be used to dynamically analyze these properties during the optimization runs of an algorithm. Moreover, the authors claim that the results provided by these measures can provide a reliable basis for accessing the efficiency of representations and genetic operators.

In this work, we will use the above mentioned framework [7] to conduct our analysis of representations and genetic operators to the problem of molecular docking.

3.2 Definitions

3.2.1 Distance Measures

Investigations with an evolutionary framework usually means considering only two spaces: the genotype space Φ_g and the phenotype space Φ_p. Genetic operators are applied on Φ_g while the fitness function, f, is applied to solutions from the phenotype space: $f : \Phi_p \rightarrow \Re$.

To establish the similarity between two individuals from Φ_p a phenotypic distance has to be defined. This measure captures the semantic difference between two solutions and is directly related to the problem being solved. The phenotypic distance can be determined with a structural distance measure. To evaluate a final ligand conformation we compare it with the experimental structures using the standard Cartesian root-mean-square deviation (RMSD):

$$RMSD_{lig} = \sqrt{\frac{\sum_{i=1}^{n} dx_i^2 + dy_i^2 + dz_i^2}{n}} \tag{5}$$

where n is the number of atoms in the comparison and dx_i^2, dy_i^2 and dz_i^2 are the deviations between the crystallographic structure and the corresponding coordinates from the predicted structure lig on Cartesian coordinate i. RMSD values below or near 2.0Å can be considered to be a success criterion. Thus, lower values mean that the observed and the predicted structures are similar.

Therefore, our structural distance measure determines the difference between RMSD values of two phenotypes:

$$d_{struct}(A, B) = |RMSD_A - RMSD_B| \tag{6}$$

Another possibility is to use a fitness-based distance where the fitness distance between two phenotypes A, B is determined. In this case, it calculates the difference between energies values of two candidate solutions

$$d_{fit}(A, B) = |f(A) - f(B)| \tag{7}$$

However, the use of this type of distance function can be considered less accurate when the main goal is to study the internal behavior of an evolutionary algorithm. The structural distance is able to reflect the structural differences found in the phenotypes, while the fitness differences can only reflect the scalar solution quality. For instance, a phenotypic distance of zero implies the same fitness values whereas the contrary is not true. In our analysis, several experiments were performed using both distance functions. The structural distance, for most cases, showed to be more accurate than the fitness-based distance. Before concluding, it should be noticed that fitness-based distances can be useful if the definition of a structural distance is difficult to define.

3.2.2 Locality Measures

We adopt the Mutation Innovation (MI) measure [7] to study the effect of mutation on locality. To predict the effect of applying this operator we use the distance between individuals in a mutation step. Let X be a solution and X^m the result of applying m mutation steps to X, then the mutation innovation is given by:

$$MI = dist(X, X^m) \tag{8}$$

MI illustrates how much innovation the mutation operator introduces, i.e., it aims to determine how much this operator modifies the semantic properties of an individual. Locality is directly related to this measure. The application of a locally strong operator implies a small modification in the phenotype of an individual. The distance between the two solutions is small. On the other hand, operators with weak locality allow large jumps on the search space.

To evaluate MI, 1000 random individuals are generated. Afterwards, a sequence of mutation steps is applied to each one of them and the distance between the original individual and the new solution is measured. In our experimentation, we start by applying a single mutation step. Later, we repeat the experiment with k successive mutation steps, with $k \in \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$.

To analyze the effect of crossover we use the Crossover Innovation (CI) measure [7]. This measure evaluates the ability of this type of operator to create offspring which are different from their parents. In other words, it quantifies the ability of the genetic operator mixing of the parent's phenotypic properties. Let C be a child resulting from the crossover application to parents P_1 and P_2, the innovation CI can be measured as follows:

$$CI = min\{dist(C, P_1), dist(C, P_2)\} \tag{9}$$

According to the previous equation, CI determines the phenotypic distance between a child and its phenotypically closer parent. Normally, the distance between parents involved in a crossover operation is expected to be directly related to CI. This means that similar parents have a tendency to create closer descendants while dissimilar parents have a propensity to generate distant offspring, i.e., with larger crossover innovations.

It is important to notice that when different operators are applied to the same individuals, i.e., under the same circumstances, the operators might induce distinct levels of innovation. This reflects on how the genetic material is combined. For a good exploration of the search space, a moderately high value of CI might be desired for crossover operators since usually they have the role of exploration. In addition, it helps to preserve some diversity in the population and ensures that a proper search space exploration is performed. However, a very high value of CI could be prejudicial since it will be difficult to preserve and combine the useful features inherited from the parents.

We study the value of CI for different crossover operators. To see how the parental distance affects crossover innovation, the following procedure is adopted: parent P_1 is randomly generated and kept unchanged while parent P_2 is derived from $P1$ via $k > 0$ consecutively applied mutations. In our experiments, CI is measured after k successive mutation steps, in a similar way to MI, with $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$.

3.2.3 Heritability Measure

Heritability is a measure of the interaction between an encoding and a crossover operator. A meaningful operator should be able to create descendants mostly from inherited substructures from either of the parents phenotypes. Crossover Loss (CL) evaluates how this condition is violated, i.e., the total size of newly introduced phenotypic substructures in a child. According to [7] and based on the definition of the structural distance, CL can be expressed as:

$$CL = \frac{1}{2}(dist(C, P_1) + dist(C, P_2) - dist(P_1, P_2)) \tag{10}$$

From the definition of CL, it should be noted that a value of zero for CI also implies a zero value for CL. However, the opposite is not true.

3.2.4 Heuristic Bias

The exploration of the search space by an evolutionary algorithm is performed by iteratively applying selection and variation operators. This process is unbiased if each element of the phenotype space has the same probability of being represented by either being generated from variation operators or, by random selection from the search space. In this situation, selection is the sole responsible to conduct the search towards fitter individuals. However, additional bias which favor phenotypes near optimal solutions, can be introduced into the process. This changes the probabilities for certain individuals of being sampled and could benefit the evolutionary search since the expected average fitness of the solutions population is higher. These type of bias can be induced by heuristics in the genotype-phenotype mapping function or by genetic operators, mainly

problem-specific variation operators. As a natural consequence of Heuristic Bias is a loss in diversity.

In our investigation, problem-specific operators are not examined since in the context of molecular docking, the typical variation operators used during the search process are standard operators. Even so, it is important to examine the heuristic bias of the operators. Since the operators, mainly mutation, use different distributions and methods to control them, it is important to establish if these differences (e.g., the adaptive variance schemes) have impact on the probabilities for phenotypes being represented.

To analyze the effect of heuristic bias in the representations and the operators, we do the following. Bias in the encodings is examined without reference to the effect of genetic operators. For each encoding, we randomly generated 1000 individuals and perform a statistical analysis. Bias in the operators is investigated by running a simple evolutionary algorithm with 100 individuals *without* selection, for each operator to be tested. The population structural distance average to the optimal solution is then analyzed. Biased operators should show an approximation to the optimal solution whilst unbiased ones must show a constant line.

4 Empirical Analysis and Discussion

We selected several instances from the *AutoDock* test suite to perform the analysis. Due to space limitations, we will only present results obtained with the HIV-1 protease/XK 263 protein-ligand complex. It has 10 rotatable bounds with 8 torsional degrees of freedom and is one of the largest complexes in the suite. Results obtained with other instances (e.g., β-Trypsin/benzamidine) follow the same trend.

4.1 Locality Analysis

4.1.1 Mutation

Table 1 shows the characteristic values for MI with a single mutation step ($k = 1$). $P(MI = 0)$ represents the percentage of cases for which $MI = 0$. $E(MI)$ and $\sigma(MI)$ show the empirically obtained mean values and standard deviations of MI when $MI > 0$. They act as estimations for the expected values. $Max(MI)$ gives the maximum value for MI.

	$P(MI = 0)$	$E(MI)$	$\sigma(MI)$	$Max(MI)$
Flip (BR)	0.6	1.17	1.54	7.50
Uniform	0.3	1.28	1.71	7.49
Gaussian	7.3	0.04	0.10	1.19
Gaussian A1	10.7	0.02	0.05	0.73
Gaussian A2	9.1	0.03	0.07	0.75
Cauchy	4.7	0.15	0.52	5.94
Cauchy A1	7.6	0.08	0.30	4.56
Cauchy A2	7.4	0.11	0.39	5.54

Table 1: Characteristic values for Mutation Innovation with $k = 1$.

We start by considering the case where mutation does not affect the phenotype, $MI = 0$ (occurring with probability $P(MI = 0)$). A large value of $P(MI = 0)$ indicates that mutation does not make often moves in the search space. In alternative, it may also be an evidence of redundancy or strong heuristic bias since many elements could map to the same phenotype. Table 1 shows that this is not the case. The probability of $MI = 0$ is low for every operator. The major observable difference is detected between the first two operators, Flip mutation (which is used with binary encoding) and Uniform mutation, and the remaining mutation operators (based on Gaussian and Cauchy distributions). The first two operators present values close to 0%. Uniform mutation displays the lowest value 0.30, and Flip mutation is close by with 0.60. Since these operators replace a complete gene in opposition to performing a small modification, this modification is enough to produce a new phenotype. This is an indication for a possible highly disruptive behavior for these two types of operators. For Gaussian and Cauchy operators the $P(MI = 0)$ values are considerable larger, ranging between 4.7% and 10.7%. However, in overall terms, these numbers are still small. The modifications operated by these distributions will produce different phenotypes although the probability of generating a number that is

small enough to induce the same individual is slightly larger. The small differences between Cauchy and Gaussian mutation, when comparing directly the operators, are explained by the thick tails of the Cauchy distribution. These allow larger variations more frequently than with Gaussian distribution and as such, lower its $P(MI = 0)$. Another important difference found is the presence of annealing schemes. Using an annealing scheme will increase the value of $P(MI = 0)$. Whilst for Gaussian-based operators the differences are not very large (7.3% for the simple operator and 10.7% and 9.1% for the annealing ones), for Cauchy-based operators it's large (4.7% for the normal operator and 7.6% and 7.4% with annealing schemes). This indicates that the presence of an annealing scheme may help lessen the behavior produced by a Cauchy-based operator whilst for the Gaussian case, that is not so evident.

Moving on to $E(MI|MI > 0)$, $\sigma(MI|MI > 0)$ and $Max(MI)$, in general, small values indicate high locality. A single mutation changes the phenotype only a little and thus, should be aspired [7]. Although lower values are good signs for a *good* locality, it should be noted that larger values for the standard deviation and for Max of MI may not necessarily be a bad indication. In our case, both distributions show low values for the locality measures. However, Cauchy mutation operators present larger values. For example, $E(MI|MI > 0)$ displays 0.15, 0.08 and 0.11 in comparison to 0.04, 0.02 and 0.03. The same pattern is observed for the remaining measures. For example, the maximum innovation introduced by a Gaussian operator is always much lesser than a Cauchy operator. The magnitude of innovation given by a Gaussian mutation ranges between 0.73 and 1.19 while Cauchy mutation is between 4.56 and 5.94. The degree of innovation induced by Cauchy-based operators is considerable higher than Gaussian ones. This means that the first application of a mutation operator, Gaussian mutation operators preserve more locality than their Cauchy counterparts. Comparing to Flip and Uniform mutation the differences are even larger. Both operators have mean and deviation values superior to 1.0 and the maximum values of MI are around 7.50. It is interesting to see that the maximum MI values are not very far from the ones obtained from Cauchy mutation, although in terms of $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$ the differences are considerable larger. This means that Cauchy mutation is able to introduce occasionally large steps, hence the high values for Max(MI), while keeping a good locality. The same effect does not happen with Flip and Uniform mutation.

	Gss A1	Gss A2	Chy	Chy A1	Chy A2
Gss	$\approx$ **0.0**	0.0963	$\approx$ **0.0**	0.6685	**0.0056**
Gss A1		**0.0033**	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**
Gss A2			$\approx$ **0.0**	0.0389	$\approx$ **0.0**
Chy				$\approx$ **0.0**	0.0113
Chy A1					0.0276

Table 2: P-values for the Wilcoxon rank sum test, with significance $\alpha = 0.01$, for Gaussian (Gss) and Cauchy (Chy) operators.

To establish if these differences are statistically significant, we performed the Wilcoxon rank sum test with significance value $\alpha = 0.01$. As expected, there are no significant differences between Flip mutation and Uniform mutation. In

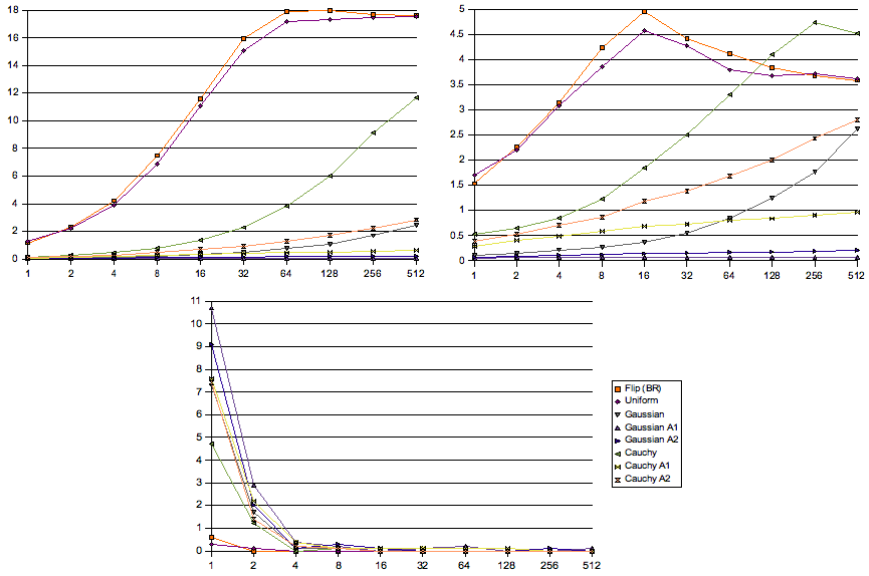

Figure 1: $E(MI|MI > 0)$, $\sigma(MI|MI > 0)$ and $P(MI = 0)$ over the $k \geq 1$ number of mutations.

addition, significant differences can be found between these operators and all the other mutation operators. In what concerns mutation operators with Gaussian and Cauchy distributions, we found significant differences between Gaussian and Cauchy mutation operators, although some of them are not strong. Table 2 shows the *p-value* obtained for each test. Results in bold indicated a statistically significant difference. In each column, the distributions are designated as Gss (Gaussian) and Chy (Cauchy).

From the table we can establish that between simple Cauchy mutation and any variant of Gaussian mutation we have significant differences for all cases. Moreover, Between Cauchy variants, there is also a significant difference between the fixed variance operator and the operator with the first annealing scheme. For the second annealing scheme there is no significant differences, as well as between both Cauchy variants with the non-fixed variance. These two variants also present differences with their Gaussian counterparts. The exception is between Gaussian with scheme A2 and Cauchy with scheme A1. In regards to the Gaussian operators, among themselves there are no differences between the fixed variance operator and with the second annealing scheme. Although we can find a statistically significant difference between annealing schemes, the p-value shows that the difference is not very large. From this table we can deduce that, in general, there are significant differences between the two groups of operators. However, the presence of an annealing scheme might change the behavior a little. For example, Cauchy with the first annealing scheme approximates to the Gaussian variants since this scheme provides a faster convergence to smaller values.

A Gaussian operator displays better locality properties but, how does the distribution of mutation innovation changes when considering $k > 1$ mutations?

We will now consider the case for $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$. Figure 1 plots the empirically obtained values for $E(MI|MI > 0)$, $\sigma(MI|MI > 0)$ and $P(MI = 0)$ over the number of mutations k.

A perusal of the $E(MI|MI > 0)$ plot reveals three distinct patterns. The first one is the behavior of Flip and Uniform mutation. Although with different encodings, these two operators operate in a identical way, inducing the same degree of locality. From $k = 1$ to $k = 64$ mutation steps, Flip and Uniform mutation have a considerable increase of innovation, from values below 2 to 18. After $k = 64$ stabilization occurs. This clearly shows that these operators induce a very low locality. The second pattern is the opposite of the first: operators with a very high locality. Belonging to this group we have both Gaussian operators with annealing schemes and Cauchy mutation with the first annealing scheme. The graph shows for all k mutation steps the same behavior, presenting almost a straight line with low innovation values. In fact, the induced locality is so strong that we might consider it as excessive. Nevertheless, for larger values of k, these two operators start to display a small $E(MI|MI > 0)$ increase. The third and final pattern is displayed by the remaining operators, although in this case, we can also distinguish two trends. Gaussian mutation and Cauchy with the second annealing scheme show an increase of innovation values starting around $k = \{16, 32\}$. The same is valid for the simple Cauchy operator. However, the magnitude of the innovation growth is much larger when compared to the other two operators. In fact, whilst Gaussian mutation and Cauchy with A2 give indications of strong locality, the simple Cauchy operator is closer to the behavior of Flip and Uniform mutation, thus showing weak locality. This is interesting to see the difference between distributions and the effect of adding annealing schemes. Without them, Gaussian mutation is able to induce a good locality but Cauchy not. Cauchy-based operators require the annealing schemes in order to be strong local operators and this difference in behavior is very strong. As for the Gaussian operators, the addition of an annealing scheme helps to induce an even higher locality. In this case, it could prove to be excessive.

Regarding $\sigma(MI|MI > 0)$ the pattern is similar but some remarks must be made. As expected, the most stable operator are the Gaussian operators with annealing schemes and Cauchy mutation with the first annealing scheme whereas the most unstable are Flip and Uniform mutation. Simple Cauchy mutation and with scheme A2 show a large values increase, followed by Gaussian mutation. The Cauchy operator starts with low standard deviation values but steadily increases its growth even passing Flip and Uniform mutation around $k = 128$. By this point, Flip and Uniform mutation show a descending trend after reaching a peak at $k = 16$. This is consistent with the mean values since by this time, both operators have stabilized, although the distance between the mutated individuals and the originals is very large. At this point there is no semantic relation between the individuals. While Cauchy mutation is still far from a stabilizing point and as such, the deviations are still increasing. Nevertheless, on $k = 512$, Cauchy shows some signs of a loss in innovation variance. This indicates that the operator is stabilizing although on a point where it induces low locality. Gaussian mutation and the Cauchy operator with the second annealing scheme present a parallel evolution of innovation. Nonetheless, the simple Gaussian mutation shows smaller values. In fact, until $k = 64$ the line is below the steadier behavior of Cauchy mutation with the first

annealing scheme only approaching the variant with the A2 scheme on $k = 512$. The loss of semantic relationship occurs later in the process in contrast with the Cauchy-based operator.

Although not shown, the evolution of the maximum innovation value, $Max(MI)$, supports the previous analysis. The same fundamental patterns are the same. The evolution of $P(MI = 0)$ over the k number of mutations is seen in the last graphic of figure 1. It shows a fast convergence to zero for all operators. As expected, it is faster for the Flip and Uniform mutation.

Grouping the distances between the original solution and the successive mutants allow us to observe the different types of changes operated by mutation for $E(MI|MI > 0)$. Given a structural distance d_{struct} between two phenotypes, the set G_i to which d_{struct} is assigned is determined the following way: $\{G0 : 0 \leq d_{struct} < 0.1; G1 : 0.1 \leq d_{struct} < 0.5; G2 : 0.5 \leq d_{struct} < 1; G3 : 1 \leq d_{struct} < 2; G4 : 2 \leq d_{struct} < 3; G5 : 3 \leq d_{struct} < 5; G6 : 5 \leq d_{struct} < 10; G7 : 10 \leq d_{struct} < 25; G8 : 25 \leq d_{struct} < 50; G9 : 50 \leq d_{struct}\}$. The specific values that were selected to determine intervals are arbitrary. The relevant information is the distribution of the structural distances through the sets. Low order sets (i.e., small variations) suggest that locality is strong.

The charts in figure 2 show the distribution of structural distances for 1000 individuals, for all operators variants, with each column representing a mutation step. We include on the top right corner a chart for a random operator. This gives a comparative idea of how locality is affected by the different operators. Furthermore, it is possible to see if over the k mutation steps, the mutated individuals reach the condition of random search. Important differences can be observed. Gaussian operators with annealing schemes exhibit high locality. With scheme A1 more than 80% of the individuals belong to the first group with the huge majority of the remaining ones belonging in the second group. With the second annealing scheme, $\geq 50\%$ of the distances belong to the first group until $k = 32$. From $k = 32$ to $k = 512$, the percentage of distances in the first groups stabilizes around 35%. Moreover, the majority of the remaining distances are within the lower three groups. This pattern is not observed elsewhere. This shows that this operator preserves the semantic properties of the individuals subject to mutation very well. It is also important to state that scheme A1 with this distribution completely removes the innovation capacity required on a mutation operator. For the second scheme, the operator is capable of producing some innovation yet it could not be enough. Comparing with the Cauchy variants, the observed patterns are similar to Gaussian with the second annealing scheme. The main difference is that Cauchy has more individuals distributed in higher groups. Thus, it is capable of producing more innovation. Whilst Cauchy with the first scheme presents a good balance between high locality and innovation capability, Cauchy with the second scheme does not. In fact, with $k \geq 128$ mutation steps, the number of individuals in the higher groups is significant.

Looking at the simple variant of Cauchy mutation, the loss of the semantic properties can clearly be seen from the last four columns (representing the mutation steps for large k). Here the amount of individuals belonging to the last groups is considerable. This supports our previous plots analysis of $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$. Gaussian mutation displays the same pattern as Cauchy mutation with scheme A2. Finally, the individuals distribution of Flip and Uniform mutation show the inefficiency of these operators. The

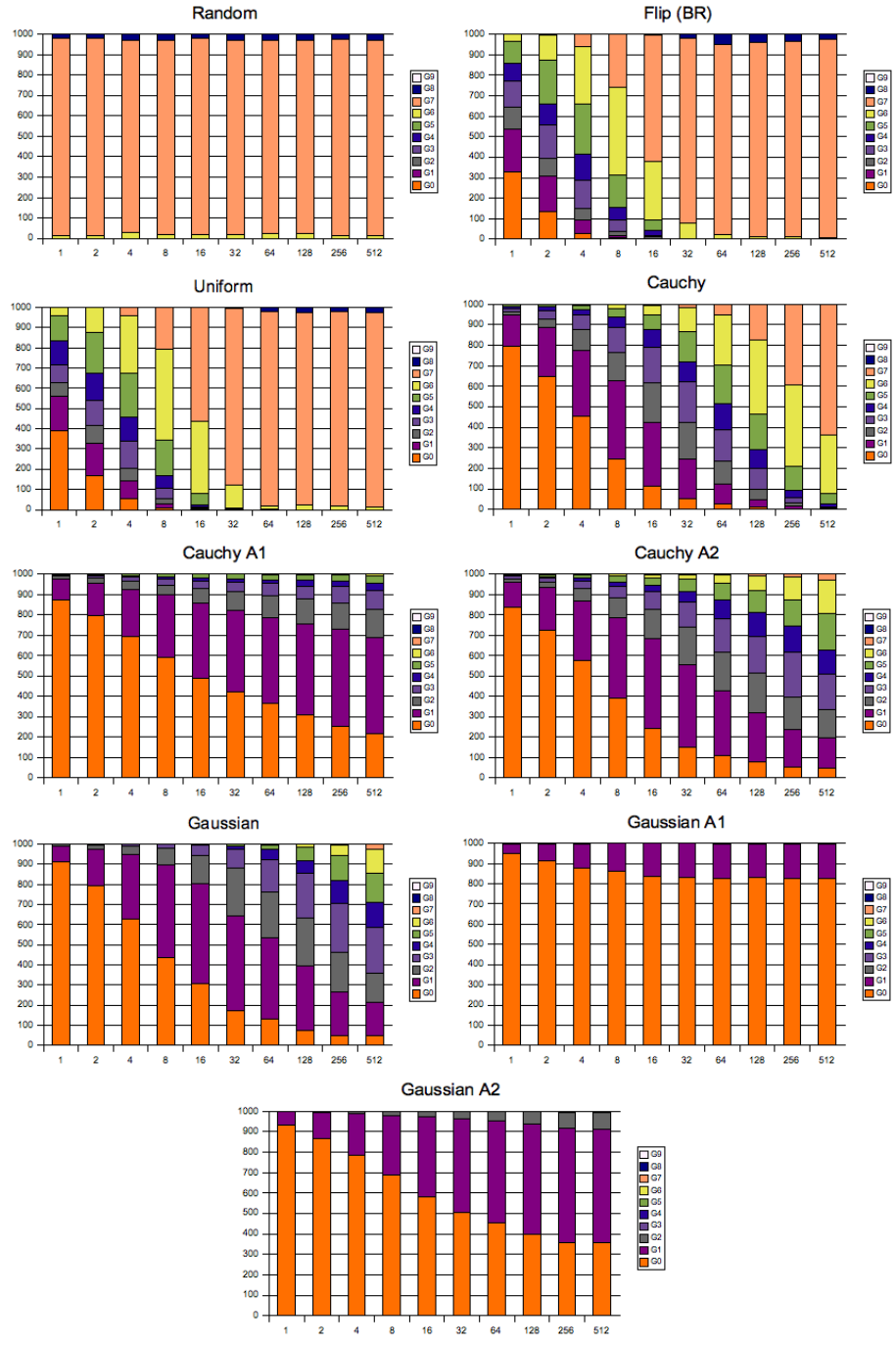

Figure 2: Distribution of structural distances for $k \geq 1$ mutations, with all mutation operators, for 1000 individuals.

induced locality is low and quickly lost. From a very early state, with $k \approx 16$, the behavior is very close to a random state, as it can be observed from the random chart.

This analysis leads us to conclude that for a mutation operator has a real influence on the degree of locality. Gaussian-based operators tend to be more stable and very locally strong. Cauchy-based operators can be more disruptive but with a good variance control scheme, it is also possible to achieve an operator which is locally strong. Therefore, on a distribution-based operator, the method that controls the variance is very important. Fixed variances can be on the long term less prone to help the operator in inducing a good locality. The simple Cauchy mutation is a good example. Flip and Uniform mutation act in the same manner. These two operators can be considered equivalent in spite of the different encoding. Their behavior is highly disruptive and very locally weak operators.

4.1.2 Crossover

To investigate locality using crossover it is essential to consider the issue of parental distance. As explained in 3.2.2, the first parent is randomly generated and kept unchanged while the second parent is derived from the first one by applying consecutively mutations steps. Therefore, the choice of the mutation operator to derive the second parent is important. Since it is not possible to present a full study considering all the mutation operators already discussed, we will focus the locality study for crossover (an later for heritability) on two operators: one Gaussian-based and another with a Cauchy distribution. We decided to use the variants with the second annealing scheme. This will allow us to see possible differences (or not) resulting from two different distributions without the interference of another variable (in this case, the annealing scheme). The simple versions of these operators are not used since with the annealing scheme we have the following: 1) an operator which induces a high locality, i.e., it will generate closer parents; 2) one operator which provides more innovation, i.e., parents with a larger parental distance but with semantically related. For binary representation only Flip mutation is used. Table 3 shows the characteristic values for CI with a single mutation step ($k = 1$). $P(CI = 0)$ represents the percentage of cases for which $CI = 0$. $E(CI)$ and $\sigma(CI)$ show the empirically obtained mean values and standard deviations of CI when $CI > 0$. They act as estimations for the expected values. $Max(CI)$ gives the maximum value for CI.

	Flip / Gaussian P_2 generation				Cauchy P_2 generation			
	$P(CI=0)$	$E(CI)$	$\sigma(CI)$	$Max(CI)$	$P(CI=0)$	$E(CI)$	$\sigma(CI)$	$Max(CI)$
1-Point (BR)	100.00	0.00	0.00	0.00	-	-	-	-
Uniform (BR)	100.00	0.00	0.00	0.00	-	-	-	-
Simple Arithmetical	56.40	0.02	0.03	0.44	54.70	0.03	0.06	0.76
Whole Arithmetical	12.70	0.02	0.05	0.68	12.60	0.04	0.11	1.55
Discrete	100.00	0.00	0.00	0.00	100.00	0.00	0.00	0.00
Simulated Binary	26.40	0.01	0.05	1.16	22.30	0.03	0.10	1.87
Blend-α	20.40	0.02	0.03	0.34	19.00	0.03	0.08	1.05

Table 3: Characteristic values for Crossover Innovation with $k = 1$. Gaussian and Cauchy mutation with the second annealing scheme are used for the 2^{nd} parent generation. Binary encoding (BR) uses Flip mutation.

Examining table 3 and observing the values for both crossover operators used with binary representation, an interesting effect is detected. Both operators are unable to produce a new offspring which is different from the parents. At first, this result could be interpreted as odd or even incorrect, however, this is the expected behavior. The individuals subject to crossover are binary encoded and the second parent, P_2, derived from the first parent, was subject to a single step of Flip mutation. The operator altered a single bit and as such, the genetic material available is not enough to produce an offspring different from the parents. In this case, binary encoding is the cause for the absence of innovation.

Moving on to the real-valued encoding and respective operators, we observe from table 3 important differences. Considering $P(CI = 0)$, Discrete crossover presents the same behavior as the crossover operators for binary encoding. This contrasts with the remaining real-valued operators. For example, Simulated Binary and Blend-α have values around 20% (26.4% and 20.4%) whilst Simple and Whole Arithmetical crossover show a percentage of 56.4 and 12.7 respectively, for a Gaussian P_2 generation (the same pattern exists the Cauchy derivation). The remaining characteristic values of CI show zero values for $E(CI|CI > 0)$, $\sigma(CI|CI > 0)$ and $Max(CI)$ for Discrete crossover. Looking at $E(CI|CI > 0)$, all operators show values close to but not zero. As for $\sigma(CI|CI > 0)$ and $Max(CI)$, the same trend occurs: maximum innovation values are small (between 0.34 and 1.87) for every crossover except Discrete crossover which is null. The reason for this reported behavior lies with the functioning of this genetic operator. The operator resembles the classic Uniform crossover for binary representation in the sense that offspring are built from interchanging genes with a given probability. Although this is a real-valued encoding, for a single mutation step only a single gene differs the second parent from the first, thus the crossover operation will always produce as offspring the same individuals, i.e., the parents, introducing no innovation. However, with multiple mutation steps, the the second parent will become considerably different and the large interaction between the genes might induce a disruptive behavior which will benefit the generation of different phenotypes. In this way, the operator could introduce large values of innovation. Moreover, this could also mean that the operator might inefficiently explore the search space if the induced locality becomes weak.

Considering the remaining crossover operators, differences in the characteristic values are more subtle, or inexistent, with the exception of $P(CI = 0)$. The arithmetical operators show $E(CI|CI > 0)$ values of 0.02, for Gaussian generation, and of 0.03 and 0.04, for Cauchy derivation. $\sigma(CI|CI > 0)$ also presents low and close values. With Simulated Binary and Blend-α crossover the pattern is similar. The conclusion to be drawn is that the operators present similar levels of high locality and thus, all of them should be local-strong operators. In spite of that, the single application of mutation is not enough to draw stronger conclusions. The analysis of the application of multiple mutation steps is essential to understand their behavior.

To establish if these differences are statistically significant, we performed the Wilcoxon rank sum test with significance value $\alpha = 0.01$. Table 4 shows the *p-value* obtained for each test. Results in bold indicated a statistically significant difference. The diagonal shows the differences between both mutation operators used to generate the second parents, the upper triangle presents the differences

	Simple	Whole	Discrete	SBX	Blend-α
Simple	0.02	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**
Whole	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**	0.03
Discrete	$\approx$ **0.0**	$\approx$ **0.0**	-	$\approx$ **0.0**	$\approx$ **0.0**
SBX	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**
Blend-α	$\approx$ **0.0**	0.12	$\approx$ **0.0**	$\approx$ **0.0**	$\approx$ **0.0**

Table 4: P-values for the Wilcoxon rank sum test, with significance $\alpha = 0.01$, for Crossover operators.

between different crossover operators using the Gaussian operator while the lower triangle shows the differences by using the Cauchy operator.

Binary representation operators are not included. A significant difference is always present. As expected, there are significant differences between all crossover operators with the exception of Whole Arithmetical and Blend-α, regardless of the type of mutation operator used for the second parent generation. For all other combinations, the *p-value* is always very close to zero. When we look at the differences between the same crossover operators but with a different parental generation, only Simple Arithmetical and Discrete crossover do not show significant differences. Although some of the values present in table 3 are very similar, statistically they are different.

Figure 3 plots the empirically obtained values for $E(CI|CI > 0)$, $\sigma(CI|CI > 0)$ and $P(CI = 0)$, applying $k \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$ number of mutations. This figure 3 contains two rows of charts. The top row refers to the data collected by using the Gaussian operator for the 2^{nd} parent's derivation while the bottom row refers to the Cauchy operator. The binary encoding operators are included on all charts for comparative and scaling purposes purposes.

Lets start the crossover analysis with the Gaussian parental derivation and binary representation. The examination of the $E(CI|CI > 0)$ plot reveals some interesting patterns. The difference between operators with binary and real-valued encodings are very large. CI increases considerable until $k = 64$, from values close to zero to 3.5 and 5.5 (1-Point and Uniform crossover respectively) and stabilizes around these values from the remainder of the k mutations. These CI values contrast with the maximum obtained by all other operators, which are below 1.0. Uniform crossover shows a larger innovation than 1-Point crossover although their behavior is very similar. However, the disruption caused by the way Uniform crossover operates justifies this difference. Uniform crossover is promoting several cut points and changes on the genotype which will result in several changes on the coordinates, after the genotype-phenotype mapping. With 1-Point crossover the changes will be smaller since in the worst case, only 50% of the genotype will be different. And even in this situation, the different half-part is still related inducing lesser innovation. Nevertheless, these variations are always superior to the ones produced by real-valued operators. In what concerns crossover for the real-valued encoding the type of behavior is similar for all operators. The values for CI are very low starting to slowly increase around $k = 32$. For the larger values of k, CI is close to 1.0 for Whole Arithmetical, approximately 0.5 for the remaining operators. These five operators exhibit a high locality. Since Discrete crossover is similar in behavior

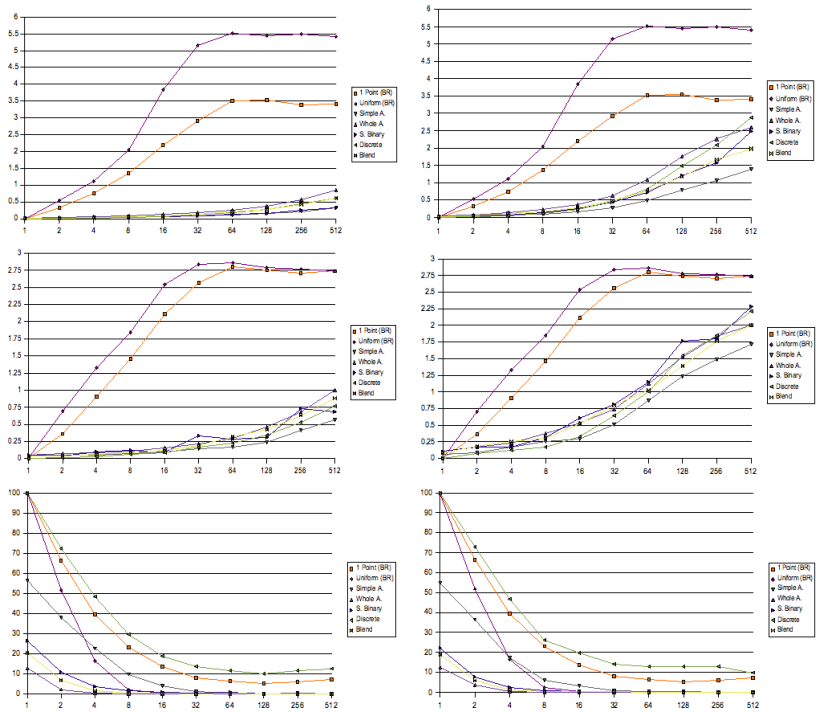

Figure 3: $E(CI|CI > 0)$, $\sigma(CI|CI > 0)$ and $P(CI = 0)$, top, middle and bottom rows respectively, over the $k \geq 1$ number of mutations (left, middle and right chart respectively). The left column displays the data using the Gaussian operator for the 2^{nd} parent's derivation while the right columns uses the Cauchy operator. The binary encoding operators are included on all charts for comparative purposes.

to Uniform crossover for binary encodings, the expected result would be a higher curve. However, this is not observed which indicates that the representation in this case is the leading reason for this difference. Moreover, the observed local strong operators generate offspring in ways that do not promote large variations on the genotypes. This might also indicate that the exploratory role is weaker in the real-valued operators. However, if this high locality correlates with good optimization results from evolutionary algorithms, then the exploration provided by these crossover operators is enough. Otherwise, the induced locality should be lower but not to the levels where no relationships between parents and offspring exist. The relation between locality and the search dynamics are analyzed in section 4.4.

Looking at the evolution of $\sigma(CI|CI > 0)$, we see a similar trend to $E(CI|CI > 0)$. The crossover operators for binary representation have the same kind of response as before: a fast increase to $k = 32$ and subsequent stabilization. The main difference relies on the proximity of the innovation deviation values. Regarding the real-valued operators, the same applies. From $k = 32$ onwards the deviation values start to increase, in a steady and slowly manner, and by $k = 512$ it is possible to distinguish the values between obtained by the different opera-

tors. $\sigma(CI|CI > 0)$ supports the identification of the two groups of operators, identified by their representation. The binary group which is characterized by some locality at the beginning but quickly displays a very weak locality. And the group of real-valued encoding, which shows high locality although it may also indicate that they can induce some excessive locality, lessening the exploratory role of crossover. However, the observation of the chart related to the evolution of $P(CI = 0)$ might indicated this effect could not be a problem. With the exception of the Simple Arithmetical crossover, which even performs worse than Uniform crossover, all the other real-valued operators quickly reach the zero probability. Comparing to the binary operators, the time to reach $CI = 0$ is long and 1-Point crossover does not even reach it. Nevertheless, this is just an indication; in section 4.4 we will discuss this issue.

Before concluding, we need to take a look at the data which relates to the generation of the 2^{nd} parent by means of Cauchy-based mutation. A brief examination of the corresponding plots, the bottom row of figure 3, we can conclude the pattern is the same for all crossover operators. In terms of the identification of the groups, it is possible to see the same information. As such, we could conclude that the choice of a mutation operator to produce the 2_{nd} parent, in order to have meaningful parental distances, is not important. However, this is not entirely correct. The plots also show a small difference between the two mutation operators used. The locally strong operators present a faster increase of innovations levels and Discrete crossover loses stability at the end. Although the pattern is similar to Gaussian-based mutation, the total amount of innovation is larger. Discrete crossover almost displays, with $k = 512$, the same amount of CI as 1-Point crossover. Looking at the $\sigma(CI|CI > 0)$ graphic, this trend is even more perceptible. What happens is that the Cauchy operator produces, for the same amount of mutation steps, parents more distant from each other. Taking into account this effect, it is natural that CI values increase to larger values. And since a Cauchy-based operator reaches a state where the semantics between the original individual and the mutated one disappear faster, this explains the observed difference. However, the important behavior of the operators remain. We also tested this hypothesis by analyzing CI using Uniform mutation for the parental derivation. Results (not shown here) confirm it: CI values increase due to the larger parental differences but the main patterns between crossover operators remain.

The charts in figure 4 show the distribution of structural distances for 1000 individuals, for all operators variants, with each column representing a mutation step. We group the distances between the original solution and the successive mutants in the same manner as before. As before, the important information is the distribution of the structural distances through the sets. Low order sets (i.e., small variations) suggest that locality is strong. The charts support our previous analysis. The genetic operators used with binary representation only provide high locality values for small values of k. For $k \geq 16$ mutation steps, the amount of individuals close to a *random* configuration is large. This is especially evident for Uniform crossover with more than 50% of the individuals in higher distance groups.

The real-valued crossover operators display in their graphics the induced high locality. Whole Arithmetical and Blend-α crossovers show a very similar group distribution with Gaussian mutation. The majority of the individuals are in the first distance group and with the increasing number of mutation

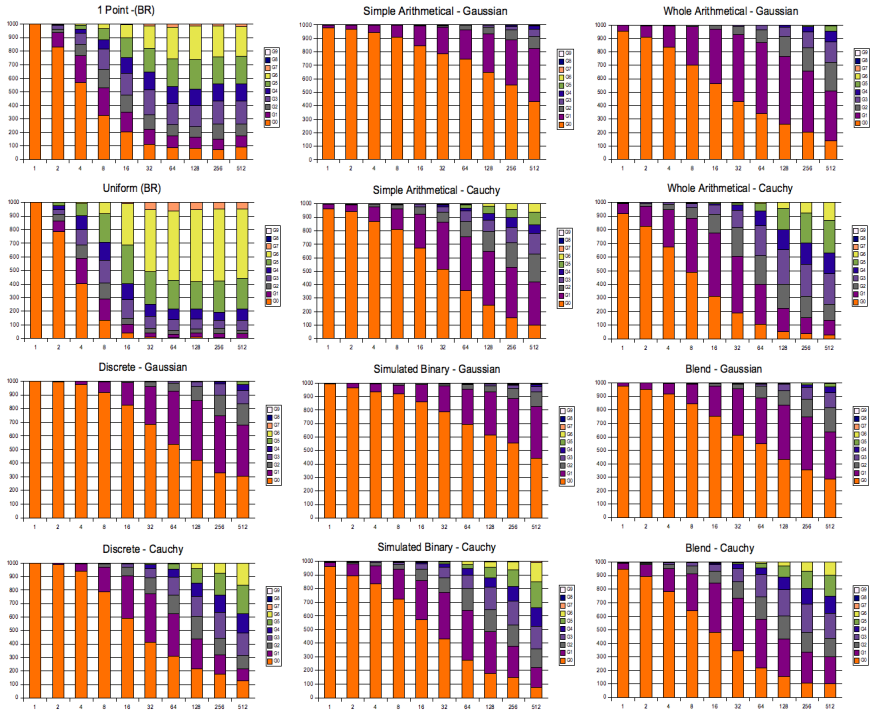

Figure 4: Distribution of structural distances for $k \geq 1$ mutations, with all mutation operators, for 1000 individuals.

steps, there is a transfer of these individuals to groups which are also close in distance. Simple Arithmetical and Simulated Binary crossovers reinforce this behavior. In fact, they appear to be the operators which induced the highest locality. This alters when we consider Cauchy-based mutation since for a higher k, the individuals are more evenly distributed to groups with larger distances. The plots are similar for all operators with the exception of Simple Arithmetical crossover, where the induced high locality is still visible.

The study of locality with crossover reinforces how important is the choice of representation and associated operators. In this case, crossover with binary encoding is expected not to perform well. It is confirmed by high locality values with parents very close to each other and moves to low values when the parental distance becomes larger. For real-valued crossover, operators have a more similar effect on locality. In fact, operators who promote subtler changes will be more locally strong, as expected. To conclude, the choice of crossover for molecular docking must be made taking into consideration the fact that this operator has a exploration role. Therefore, it is important to provide some level of innovation. Additionally, this should be balanced with a good degree of locality. However, in an evolutionary algorithm it is the dynamics of crossover and mutation together that are important.

4.2 Heritability Analysis

Empirical estimations of CL were calculated from the experiments described in the previous section, using equation 10. Figure 5 shows the estimated $E(CL|CL > 0)$ and $\sigma(CL|CL > 0)$ plotted over k number of mutations. In these plots, the second parent is generated using the Gaussian-based operator. Plots with the estimations but using the Cauchy operator follow the same pattern and as such, they are not displayed. Binary representation and associated operators are also presented in the graphics.

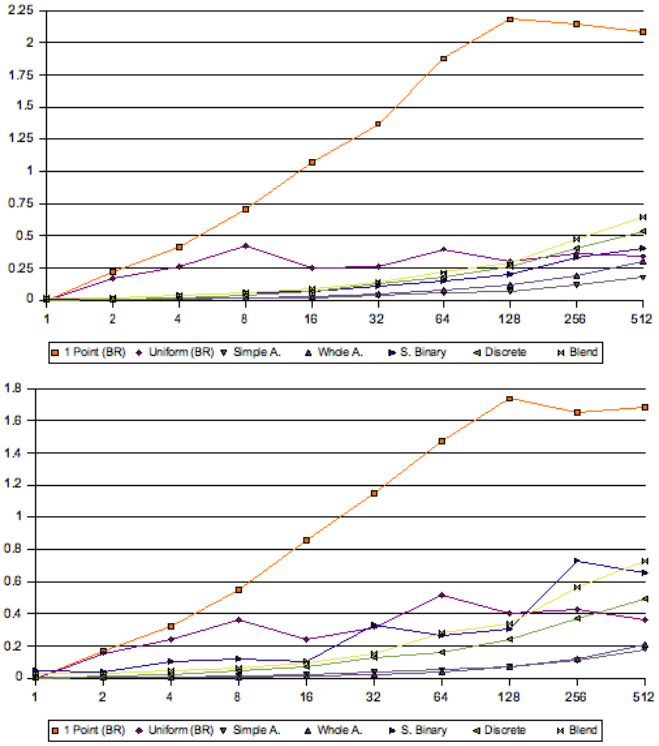

Figure 5: $E(CL|CL > 0)$ and $\sigma(CL|CL > 0)$ over the $k \geq 1$ number of mutations, left and right chart respectively, using the Gaussian operator for the 2^{nd} parent's derivation (with Cauchy operator the pattern is similar). The binary encoding operators are also included.

From the observation of the plots, the general tendency is for all operators to exhibit small values of expected crossover loss with a slight increase towards larger values of k, i.e., when the distance between parents becomes larger, heritability becomes less strong. This is particularly visible for Blend-α crossover, nevertheless, the values are still reasonable. In contrast, Simple Arithmetical crossover shows the lower CL values and has the stronger heritability. In the other extreme lies Discrete crossover. The values for $E(CL|CL > 0)$ and $\sigma(CL|CL > 0)$ are always considerable large. Heritability for the real-valued encoding with this operator is particularly poor regardless of the parental distance. As before, the reason for this behavior is the large disruption the opera-

tor applies when generating the offspring. The inherited sub-structures are not meaningful between each other since there is a dependency between the genes.

As for binary representation, 1-Point crossover starts to display small expected and standard deviation values but quickly they start to grow large when the distance between parents increases. With $k = 4$, $E(CL|CL > 0)$ and $\sigma(CL|CL > 0)$ are already superior to all the strong heritability operators and from $k = 32$, expected values are always the larger obtained by an operator. For $\sigma(CL|CL > 0)$, although they are not the larger values, by $k = 128$ they are very close to the ones attained by Discrete crossover. Since this is a binary representation, this type of operator is not the most adequate for a good mix-in and arrangement of good sub-structures. A little surprisingly, Uniform crossover has the opposite behavior. The corresponding crossover loss values are low and remain close to real-valued operators which displayed good heritability. From the figure, we see that for low values of k, $E(CL|CL > 0)$ and $\sigma(CL|CL > 0)$ are slightly larger but for high values of k, they are in the same level. The trend is stable like Discrete crossover, however, the induced heritability is better. It could have been expected that Uniform crossover would display values similar to Discrete crossover since it is a binary encoding. The interesting factor is that, the encoding is responsible for this behavior since the way this operator functions allows a good arrangement of sub-structures. In spite of showing poor locality, in terms of heritability, this operator is stronger.

4.3 Heuristic Bias Analysis

We examine now the heuristic bias of the operators. Although no problem specific knowledge is used in any of the genetic operators, we feel that it is important to observe if a bias is present when using the different distributions and control variance schemes. Even if no specific knowledge is incorporated into the operators, the way they operate might still produce a small bias. Furthermore, the initial population is randomly initialized but performed according to a grid structure in order to have a better initial quality of the first population. The heuristic bias of the grid is also measured in this section.

We start to analyze the bias in the operators by running a simple evolutionary algorithm with 100 individuals *without* selection (generational replacement), for each operator to be tested, applying each operator individually it with a 1.0 rate. We run the algorithm for 100 generations (30 runs). To measure the bias we use the following metric:

$$d_{lig-opt} = \sum_{i=1}^{n} \frac{d_{struct}(lig, opt)}{n} \tag{11}$$

where n is the population size, lig the current individual and opt the best solution. Biased operators should show an approximation for the optimal solution (or not) whilst unbiased ones should display a constant behavior.

Let's start our analysis with mutation. The left plot in figure 6 shows the measured heuristic bias for all real-valued operators (we do not consider the binary operators for this part of the analysis) without the grid heuristic initialization. As expected, all operators have a similar behavior and there are no indications of a bias. The gaussian and cauchy-based operators are unable to

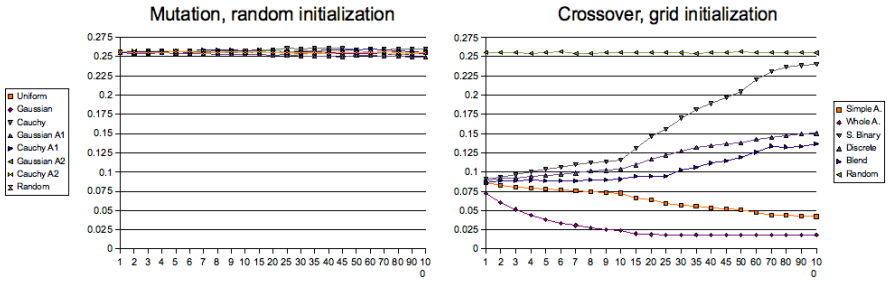

Figure 6: Plots show the mean of $d_{lig-opt}$ over 100 generations of an evolutionary algorithm with random selection, population with 100 individuals. Left plot uses shows mutation with random initialization, and center plot with grid initializaton. The right plot presents crossover with grid initialization.

bias the process to more fitter individuals. The operation of the different distributions do not alter in any significant manner the type of generated individuals. However, when the grid initialization is used, the results are slightly different. The quality of the initial population improves with the grid. However, since there is no specific heuristic encoded in the operators, they are expected not keep the initial quality and modify the individuals in a manner that their quality decreases. The best example of this behavior is Uniform mutation. From the initial generations it starts to approach the a pure random status and for the last 50 generations, the population presents the same behavior as a random one. This confirms that this mutation operator is not able to preserve the quality initially introduced and acts in the sense of a negative bias, i.e., a bias not towards the better solutions but the worse ones. The simple cauchy operator gives some similar indications although in a much smaller scale. Starting from generation 50 it starts to drift from the constant behavior in the direction of the pure random population. Since these are the operators that promote larger changes in the individual's phenotypes, on a initial good quality population, these will changes will lead to a decrease of quality since the effect of biased selection is not present to preserve the good individuals. Moreover, the other operators promote small changes, especially the ones with variance control schemes, and this is enough to preserve some of the quality of the population, i.e., not losing the effect of the initial heuristic bias of the grid.

As for crossover, the case is not very different from mutation. When considering pure random populations, the performance pattern is similar to mutation. All operators do not show a significant bias towards better solutions. Again, with grid initialization results are more interesting. As the right plot of figure 6 confirms, the same improvement on the initial populations is present comparing to the random case. In addition, we also have operators that lose the initial quality. Simulated binary, blend-α and discrete crossover start to display an approximation towards worse solutions around generation 10-15. Simulated binary has the worst performance since by generation 100 is very close to random. On the other hand, both arithmetical crossover show a bias towards fitter solutions. The whole arithmetical crossover displays the larger bias from the first generations and stabilizes around generation 20. The simple arithmetical

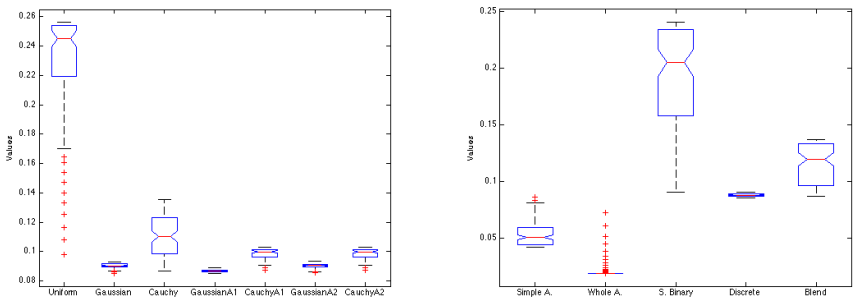

Figure 7: Box-plots for mutation (top) and crossover (bottom) showing the heuristic bias differences, with grid initialization, between the several genetic operators.

crossover behaves in a slightly different manner. It starts to converge to better solutions around generation 15 and stops in generation 70. The presence of a very small heuristic bias in these operators is surprising since they do not have any kind of specific knowledge about the problem. The main difference between these operators and the others is, once more, the degree of changes performed on the genotypes. The later operators can act more disruptively than the arithmetical operators. Nevertheless, we must point out that the differences between all these operators are small.

Figure 7 shows the box-plots for mutation and crossover after applying the Kruskal-Wallis test. This test is a non-parametric version of the classical one-way ANOVA, and an extension of the Wilcoxon rank sum test to more than two groups. It compares samples from two or more groups of data and returns the p-value for the null hypothesis that all samples are drawn from the same population, or equivalently, from different populations with the same distribution. The plots allow us to better see the differences between operators. Significant differences can be found between all crossover operators. As for mutation, differences between Gaussian operators are less clear although they existent between Gaussian and Cauchy-based operators. The same pattern can be found in-between Cauchy operators although the simple Cauchy operator shows a much larger confidence interval.

For the sake of completeness, we also analyzed the heuristic bias between the two encodings: binary and real-valued. Since semantically they are similar and no heuristics are used, no bias is expected to be found. The only effect that must be visible is the grid initialization. Bias in the encodings is examined without reference to the effect of genetic operators. We randomly generated 1000 individuals, for each one, and perform a statistical analysis. Table 5 contains the results and it is clear that no differences exist between both encodings. They display the same behavior, even with the grid initialization. As expected, the grid initialization imposes a strong heuristic bias in the first population.

Representation	Grid Initialization	Mean	Std. Deviation
Binary	no	0.25	0.03
	yes	0.08	0.05
Real-valued	no	0.25	0.04
	yes	0.08	0.05

Table 5: Bias of representations. The mean and standard deviation of the metric $d_{lig-opt}$ for 1000 randomly generated individuals (with and without grid). A higher value indicates a lower bias.

4.4 Search Dynamics Analysis

The previous sections presented our analysis concerning several genetic encoding properties and how variations operators affect them. The analysis revealed some conditions that might influence the performance of an evolutionary algorithm. However, the previous analysis were *static*, i.e., they were made without performing complete runs of an evolutionary algorithm. The exception was the heuristic bias property. Even so, it was an evolutionary algorithm without selection. Naturally, the evolutionary search process dynamics are influenced by the interplay of variation operators, selection, replacement strategy and other additional factors. In this section, we will present an analysis of the *dynamic* case, i.e., complete evolutionary runs to confirm the results of the previous analysis.

Since we are mainly interested in the behavior attained during the evolutionary runs, we use a simple generational evolutionary algorithm. The algorithm uses binary tournament selection, elitism and no local search methods. The parameters are also standard: 100 individuals, crossover rate of 0.8, mutation rate of 0.05 and 100 generations. We performed 30 runs with the same initial conditions and with different random seeds. All initial populations were randomly generated. Finally, for mutation, we will focus our analysis on the Gaussian and Cauchy operator with the second annealing scheme. For crossover, we concentrate on the real-valued crossover operators.

In order to analyze the dynamics of the evolutionary algorithm, we measure the RMSD and energy mean of the population. Since the evolutionary algorithm uses as fitness function an energy function, it is important to related it to the solution's structure. In this manner, we establish a relation between the static measures that took into account the phenotype with the search process.

Figure 8 contain the plots that show the RMSD mean of a population. The top plot uses Gaussian-based mutation and the bottom one Cauchy-based mutation. We include binary encoding with Flip mutation for comparative purposes. Figure 9 shows the plots with the energy mean of a population under the same conditions. From these figures we can detect some important differences and similarities. The plots provide a resemblance in terms of RMSD and energy level, although there is a fast convergence of the energy level compared to RMSD, for all types of operators. In addition, it is visible the differences of performance for all genetic operators. In fact, there are no major differences between groups of operators. The most striking difference of performance we find is related to Discrete crossover: it shows the worst RMSD behavior whereas for energy the binary operators perform worse. Regarding the best levels for

energy and RMSD the plots are clear: the arithmetical crossovers have the best performance, especially the Whole Arithmetical operator. In terms of mutation, the differences are not perceptible but we must add that mutation rate is low and crossover is the main variation force in this scenario.

The important aspect from these figures is that supports the previous analysis, mainly in terms of crossover. Looking at the energy and RMSD plots, arithmetical crossover operators with gaussian-based mutation are faster to converge towards fitter solutions. Moreover, it clearly shows that the choice of variation operators, and encoding, is important in the context of molecular docking. However, the plots of these evolutionary runs show only the performance of the genetic operators in the instance used during the analysis. It is important to test and relate the performance attained by these components on more instances.

Additional experiments were also made with other instances to assess the validity of this analysis in other instances, confirming the results present here. It is not possible to present a full analysis for all the studied molecules, with every single component and parameters. We focus on the overall picture across several instances. It is important to notice that our goal is not to obtain new best solutions or be competitive with current approaches. The displayed results are only for analysis and have the main objective of showing patterns of performance, i.e., to establish if the analysis presented in this paper reflects in other instances.

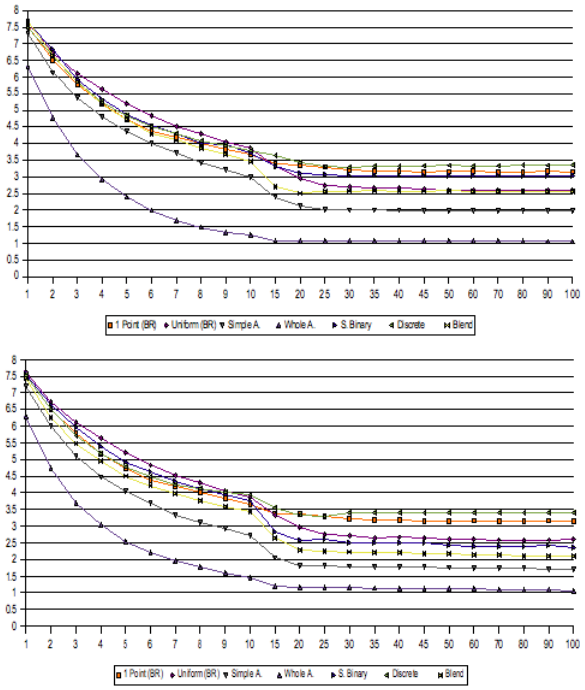

Figure 8: Plots show the RMSD mean of a population with 100 individuals, over 100 generations (30 runs). Top plot uses Gaussian-based mutation and bottom plot shows Cauchy-based mutation.

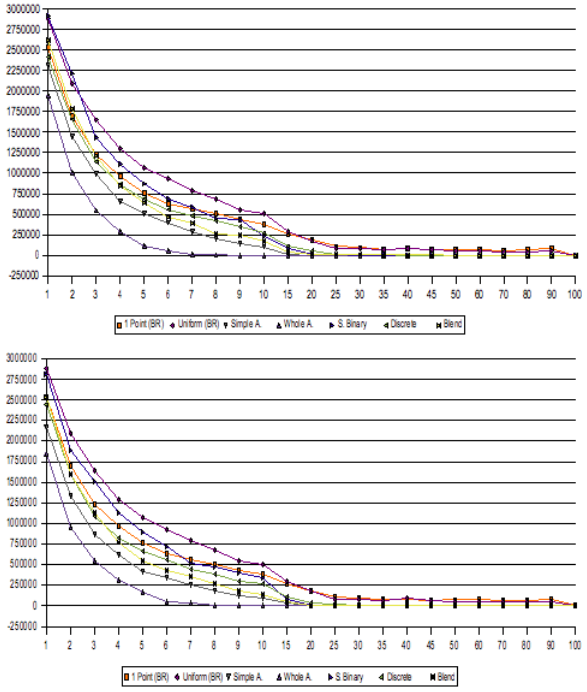

Figure 9: Plots show the Energy mean of a population with 100 individuals, over 50 generations (30 runs). Top plot uses Gaussian-based mutation and bottom plot shows Cauchy-based mutation.

Table 6 contains the results evolutionary algorithm optimization runs on eight molecules. These instances are included in the *Autodock* suite and their dimension is indicate in the row below each molecule designation. The algorithm configuration is the same used before with the same parameters values. The table displays the information according to each crossover operator. Gaussian mutation with the second annealing scheme is used with the real-valued encoding. The *best* and *mean* rows after label E contains the Energy average of the best solutions found and the mean of the population. The label R is for the RMSD values, i.e., the population's mean and the Improvement Rate (IR). This last measure indicates the RMSD improvement in the population. It measures the percentage of structural improvement from the first generation to the last. Values in bold indicate best results. Finally, the last column *Avg* contains the average of each measure on all problem instances.

		Instance	1adb	1bmm	1hvr	1nnb	1tnh	2dbl	3ptb	7abp	Avg
		Dimension	21	19	17	16	9	13	7	11	
Simple A.	E	best	**331.40**	**3.55**	165.41	-1.83	-2.07	-2.10	-2.36	8.90	62.61
		mean	**399.79**	52.61	168.16	-1.82	-2.06	-2.05	-2.35	9.31	77.70
	R	mean	4.80	6.65	1.99	6.13	6.62	4.87	5.24	3.57	4.98
		IR	41.91	23.06	72.59	17.42	26.74	34.71	38.55	54.05	38.63
Whole A.	E	best	425.95	4.80	**-8.48**	**-3.20**	**-2.57**	**-5.90**	**-3.22**	**4.03**	**51.43**
		mean	440.24	**5.23**	**-8.28**	**-3.17**	**-2.56**	**-5.85**	**-3.21**	**4.51**	**53.36**
	R	mean	**2.92**	**4.25**	**1.00**	**2.73**	**3.54**	**2.36**	**2.75**	**1.30**	**2.61**
		IR	**61.62**	**45.75**	**84.10**	**58.28**	**56.49**	**64.22**	**64.16**	**81.58**	**64.52**
S. Binary	E	best	4413.52	105.25	844.64	-0.35	-1.69	-1.09	-1.98	-0.03	669.78
		mean	4543.16	111.43	1009.39	-0.21	-1.67	131.42	-1.96	46.95	729.81
	R	mean	3.02	6.16	1.78	6.04	8.40	5.38	7.40	5.68	5.48
		IR	42.52	31.94	57.24	21.84	19.48	24.64	26.30	39.80	32.97
Discrete	E	best	5730.73	38.37	1592.20	-1.33	-1.61	-2.58	-1.82	-0.79	919.15
		mean	6123.97	357.74	1713.57	-1.31	48.24	232.35	-1.81	-0.69	1059.01
	R	mean	6.151	7.219	3.342	6.917	8.347	6.037	8.120	7.718	6.73
		IR	27.27	20.05	55.25	9.50	11.70	20.92	9.83	6.63	20.14
Blend	E	best	4451.07	105.15	1116.21	1.58	-1.76	0.84	-2.35	4.95	709.46
		mean	4555.35	799.53	1158.92	1.78	-1.75	336.78	102.01	5.39	869.75
	R	mean	4.42	6.46	2.96	6.72	7.12	6.24	6.30	5.71	5.74
		IR	47.94	28.15	60.05	12.83	25.20	19.93	29.67	30.53	31.79
1-Point	E	best	2742.53	-2.03	303.72	-2.68	-1.70	-2.75	-2.14	2.71	379.71
		mean	1.46E+05	1.14E+05	7.44E+04	3.64E+04	2.21E+04	8.09E+04	2.02E+04	2.64E+04	6.50E+04
	R	mean	5.50	6.85	3.15	6.81	7.86	6.16	7.41	6.17	6.24
		IR	36.15	23.87	58.18	10.79	16.37	19.90	18.29	24.18	25.97
Uniform	E	best	298.38	9.83	605.95	-2.49	-1.53	-2.86	-1.59	2.10	113.47
		mean	8.48E+04	9.03E+04	5.81E+04	2.93E+04	2.04E+04	4.89E+04	2.57E+04	1.92E+04	4.71E+04
	R	mean	5.22	6.50	2.54	6.00	8.18	5.58	8.10	5.00	5.89
		IR	41.18	28.58	67.34	21.65	12.15	29.29	10.52	39.81	31.32

Table 6: Results for the evolutionary algorithm optimization runs on 8 molecules.

A quick look at the table allow us to detect some important differences. The most important one is that Whole Arithmetical crossover with Gaussian-based mutation consistently performs well on all the docking instances. In terms of RMSD values, the best population means and improvement rate are displayed. As for the energy levels, with the exception of instance *1adb* and *1bmm*, it also shows the same performance. Moreover, the *Avg* simply confirms this behavior. A very important detail to notice is the improvement rate. The use of Whole Arithmetical crossover and Gaussian mutation with an adaptive scheme attained not only the best values but rates above 56% on every instance. The sole exception was instance *1bmm* which only achieved a rate of 45.75%. The global average is 64.52%. By contrast, all the other operators present much lower improvement rates and in most cases, below 40%. This is supported by looking at the global averages, ranging from 25.97% to 38.63%. In terms of energy, Simple Arithmetical crossover presents a good performance although slightly worser than the Whole Arithmetical crossover. As for the structural measures, in spite of being the second best, its performance is still far from the best crossover as indicated by the global *Avg* column: 38.63% versus 64.52%, a difference of 25.89%!

The performance of the remaining operators also follow the expected behavior: Discrete crossover and the binary operators show overall the weakest performance. The exception to this situation is the relatively good capability for Discrete crossover to attain good best solutions in terms of energy. The observation of the row containing the average of the best solutions exhibits values which are better than Simulated Binary and Blend-α crossover operators. Nevertheless, when examining the population mean we quickly see the performance is considerably worse. Before concluding this section, we must add that changing mutation didn't alter the patterns displayed in table 6, even though the overall best and mean values have lower quality. As table 7 shows, the differences in results between Gaussian and Cauchy mutation with the second annealing scheme are not very large. The observation of the overall average gives a general overview. In terms of energy, the best mean are near, 23.94 for Gaussian and 25.87 for Cauchy, but the gap is wider for the population's mean: 25.45 to 106.06. As for the structural measure, the mean value is also close since 2.29 is slightly lower then 2.51. However, the improvement rate is considerable better for Gaussian mutation: 68.72% when compared to 59.47% of Cauchy. Finally, a perusal of the table quickly reveals that the majority of the best values belong to the Gaussian operator.

In addition, other factors could influence search dynamics, e.g., local search methods. However, results shown here help to establish the role and performance of each operator and their contribution, in the context of molecular docking and in accordance with the previous static analysis and external investigations. In addition, local search methods can have a great impact on the locality of operators and therefore, influence considerably the search performance. The effect can be positive or negative. This raises the question if a study should include from the start these methods. In this problem, several evolutionary approaches do not use local search and as mentioned before, some studies made, e.g., [4], state that the use of local search can be harmful to search in this problem. Since competitive methods can be developed without local search [2], an investigation with and without local search, makes perfect sense.

		Instance	1adb	1bmm	1hvr	1nnb	1tnh	2dbl	3ptb	7abp	Avg
		Dimension	21	19	17	16	9	13	7	11	
Gaussian A2	E	best	**199.24**	4.05	**9.27**	**-3.94**	**-3.07**	-4.91	-3.66	-5.50	**23.94**
		mean	**210.60**	4.30	**9.65**	**-3.89**	**-3.06**	**-4.87**	-3.64	-5.46	**25.45**
	R	mean	**2.87**	3.92	1.17	**1.97**	**2.44**	**2.54**	**1.95**	1.44	**2.29**
		IR	**62.07**	49.60	81.63	**70.52**	**70.36**	**61.26**	**74.78**	79.55	**68.72**
Cauchy A2	E	best	217.53	**-2.26**	16.40	-3.58	-2.82	**-7.55**	**-3.91**	**-6.86**	25.87
		mean	289.25	**-1.98**	103.91	-3.51	332.36	138.64	**-3.88**	**-6.32**	106.06
	R	mean	3.19	3.95	**1.08**	2.58	3.36	2.78	2.38	**0.74**	2.51
		IR	6.86	48.59	**83.25**	60.52	59.43	58.14	69.72	**89.28**	59.47

Table 7: Results showing the differences between both mutation operators for the evolutionary algorithm optimization runs on the eight molecules. Values in bold indicate best results.

5 Conclusions

We investigated the influence of several crossover and mutation operators, with two representations, when applied to molecular docking optimization. The most successful evolutionary algorithms for this problem use a real-valued representation as the basis for their model. Different variation operators have been used, however, no studies were performed to conclude about their efficiency and performance with the exception of [4]. It is important to understand the behavior and influence of these components in order to design new and more efficient evolutionary algorithms for the molecular docking problem. Our investigation focused on encoding properties and how the different variation operators affect them. Locality, heritability and heuristic bias were analyzed in this work. We used the framework proposed by [7] to study these properties in a static and inexpensive way, and later related the findings with actual runs from evolutionary algorithms.

Results confirm that high locality is important and explain the behavior of different crossover and mutation operators. Gaussian mutation provides locally strong operators and this is especially true when used in conjunction with an annealing scheme. This is an indication that more fine-tuning of the conformations is allowed. On the other hand, Cauchy-based operators show a lesser degree of locality. The operator with the annealing scheme shows a locality similar to Gaussian mutation with fixed variance. Thus, these operators can provide a more exploratory role. Regarding crossover, the class of Arithmetical operators have shown to be more balanced in terms of locality, providing a good level of locality and innovation. A similar pattern is also displayed with other operators with the exception of Discrete crossover and operators for the binary encoding. Nevertheless, when used in conjunction with mutation, Simulated Binary and Blend-α crossovers do not perform so well as the Simple and Whole Arithmetical operators. The same kind of behavior is also visible with the heritability property: the operators who generate offspring with more substructures from the parents are the Simple and Whole Arithmetical crossover. Discrete crossover is the complete opposite, followed by 1-Point crossover (for the binary representation). In what concerns the heuristic bias, the different mutation operators do not exhibit a bias towards fitter solutions since no specific problem knowledge is included. However, when in presence of a population initialization heuristic, Uniform mutation is not able to preserve the introduced bias. The same occurs with crossover. The main difference is that a small bias is present in the Arithmetical crossovers in the direction of better solutions. Finally, actual optimization runs in different instances of the problem support the analysis findings. The performance and behavior of the variation operators are consistent.

Although specific analysis results were shown for one single problem instance, it is natural that, for other instances the actual absolute values are different. In spite of that, additional tests were performed on different instances and the same qualitative trends have been found. Previous studies have mainly applied evolutionary techniques to molecular docking and investigations on *why* the components are able to achieve the results are not provided. As such, the useful outcome from this work is twofold: 1) it explains in terms of encoding properties the operators under investigation; 2) it provides hints on how future genetic operators can be developed. To conclude in this work, local search methods

were not considered. These techniques will be the focus of future research, with some initial work already described here [32] , since the impact of local search is an important aspect of an evolutionary algorithm.

References

[1] Neumaier, A.: Molecular modeling of proteins and mathematical prediction of protein structure. SIAM Review **39** (1997) 407–460

[2] Thomsen, R.: Protein-ligand docking with evolutionary algorithms. In Fogel, G.B., Corne, D.W., Pan, Y., eds.: Computational Intelligence in Bioinformatics. Wiley-IEEE Press (2008) 169–195

[3] Morris, G.M., Olson, A.J., Goodsell, D.S.: Protein-ligand docking. In Clark, D.E., ed.: Evolutionary Algorithms in Molecular Design. Wiley-VCH (2000) 31–48

[4] Thomsen, R.: Flexible ligand docking using evolutionary algorithms: investigating the effects of variation operators and local search hybrids. Biosystems **72** (2003) 57–73

[5] Sendhoff, B., Kreutz, M., Seelen, W.V.: A condition for the genotype-phenotype mapping: Casualty. In: 7th Int. Conf. on Genetic Algorithms. (1997) 73–80

[6] Rothlauf, F.: On the locality of representations. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003). (2003) 1608–1609

[7] Raidl, G.R., Gottlieb, J.: Empirical analysis of locality heriability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem. Evolutionary Computation Journal **13** (2005) 441–475

[8] Julstrom, B.A.: The blob code: A better string coding of spanning trees for evolutionary search. In: GECCO '01: Proceedings of the 2001 conference on Genetic and evolutionary computation, San Francisco, CA, USA (2001) 256–261

[9] Soak, S.M., Corne, D.W., Ahn, B.H.: The edge-window-decoder representation for tree-based problems. IEEE Trans. Evolutionary Computation **10** (2006) 124–144

[10] Morris, G.M., Goodsell, D.S., Halliday, R.S., Huey, R., Hart, W.E., Belew, R.K., Olson, A.J.: Automated docking using a lamarckian genetic algorithm and and empirical binding free energy function. Journal of Computational Chemistry **19** (1998) 1639–1662

[11] Pereira, F.B., Marques, J., Leitão, T., Tavares, J.: Analysis of locality in hybrid evolutionary cluster optimization. In: Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, Canada, IEEE Press (2006) 8049–8056

[12] Tavares, J., Tantar, A.A., Melab, N., Talbi, E.G.: The influence of mutation on protein-ligand docking optimization: a locality analysis. In: Proceedings of the 10th International Conference on Parallel Problem Solving From Nature. (2008)

[13] Dixon, J.S.: Flexible docking of ligands to receptor sites using genetic algorithms. In: Proc. of the 9th European Symposium on Structure-Activity Relationships, Leiden, The Netherlands, ESCOM Science Publishers (1993) 412–413

[14] Xiao, Y.L., Williams, D.E.: Molecular docking using genetic algorithms. In: SAC '94: Proceedings of the 1994 ACM symposium on Applied computing, New York, NY, USA, ACM (1994) 196–200

[15] Meza, J.C., Judson, R.S., Faulkner, T.R., Treasurywala, A.M.: A comparison of a direct search method and a genetic algorithm for conformational searching. Journal of Computational Chemistry **17** (1996) 1142–1151

[16] Gehlhaar, D.K., Verkhivker, G., Rejto, P.A., Fogel, D.B., Fogel, L.J., Freer, S.T.: Docking conformationally flexible small molecules into a protein binding site through evolutionary programming. In: Evolutionary Programming. (1995) 615–627

[17] Moitessier, N., Englebienne, P., Lee, D., Lawandi, J., Corbeil, C.: Towards the development of universal, fast and highly accurate docking/scoring methods: a long way to go. British Journal of Pharmacology **153** (2007) 1–20

[18] Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)

[19] Deb, K., georg Beyer, H.: Self-adaptive genetic algorithms with simulated binary crossover. Evol. Comput. **9** (2001) 197–221

[20] Lozano, M., Herrera, F., Krasnogor, N., Molina, D.: Real-coded memetic algorithms with crossover hill-climbing. Evol. Comput. **12** (2004) 273–302

[21] Korb, O., Stützle, T., Exner, T.: An ant colony optimization approach to flexible protein-ligand docking. Swarm Intelligence **1** (2007) 115–134

[22] Manderick, B., de Weger, M., Spiessens, P.: The genetic algorithm and the structure of the fitness landscape. In: 4th International Conference on Genetic Algorithms. (1991) 143–150

[23] Wright, S.: The roles of mutation, inbreeding, crossbreeding and selection in evolution. In: Proceedings of the VI International Conference on Genetics, Vol. 1. (1932) 356–366

[24] Jones, T.: Evolutionary Algorithms, Fitness Landscapes and Search. PhD thesis, University of New Mexico, Albuquerque, New Mexico (1995)

[25] Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman, L., ed.: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, Morgan Kaufmann (1995) 184–192

[26] Weinberger, E.D.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics **63** (1990) 325–336

[27] Merz, P., Freisleben, B.: Fitness landscapes and memetic algorithm design. In Corne, D., Dorigo, M., Glover, F., eds.: New Ideas in Optimization. McGraw-Hill, London (1999) 245–260

[28] Merz, P.: Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Efective Search Strategies. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany (2000)

[29] Tavares, J.: Evolvability in Optimization Problems: the Role of Representations and Heuristics. PhD thesis, Department of Informatics Engineering, University of Coimbra, Portugal (2007)

[30] Tavares, J., Pereira, F.B., Costa, E.: Multidimensional knapsack problem: A fitness landscape analysis. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cynernetics **38** (2008) 604–616

[31] Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms. Springer (2002)

[32] Tavares, J., Tantar, A.A., Melab, N., Talbi, E.G.: The impact of local search on protein-ligand docking optimization. In: Proceedings of the 8th International Conference on Hybrid Intelligent Systems. (2008)

Centre de recherche INRIA Lille – Nord Europe
Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399