Journal of Computer-Aided Molecular Design (2005) 19: 585–601
DOI 10.1007/s10822-005-9015-1

ENPDA: an evolutionary structure-based *de novo* peptide design algorithm

Ignasi Belda[a], Sergio Madurga[a], Xavier Llorà[b], Marc Martinell[a], Teresa Tarragó[a], Mireia G. Piqueras[c], Ernesto Nicolás[c] & Ernest Giralt[a,c,*]
[a]*Institut de Recerca Biomèdica de Barcelona, Parc Científic de Barcelona, Universitat de Barcelona, Josep Samitier, 1-5, Barcelona, E 08028 Spain;* [b]*Illinois Genetic Algorithms Laboratory, National Center for Supercomputing Applications, University of Illinois at Urbana–Champaign, 117 Transportation Building, 104 South Mathews Urbana, Illinois, IL, 61801-2996, USA;* [c]*Departament de Química Orgànica, Universitat de Barcelona, Martí i Franquès, 1-11, Barcelona, E 08028 Spain*

Received 5 April 2005; accepted 14 August 2005

Key words: *de novo* design, evolutionary algorithms, peptide design

Summary

One of the goals of computational chemists is to automate the *de novo* design of bioactive molecules. Despite significant advances in computational approaches to ligand design and binding energy evaluation, novel procedures for ligand design are required. Evolutionary computation provides a new approach to this design endeavor. We propose an evolutionary tool for *de novo* peptide design, based on the evaluation of energies for peptide binding to a user-defined protein *surface patch*. Special emphasis has been placed on the evaluation of the proposed peptides, leading to two different evaluation heuristics. The software developed was successfully tested on the design of ligands for the proteins prolyl oligopeptidase, p53, and DNA gyrase.

Introduction

A primary goal in computational chemistry – as well as in general drug development – is the automation of *de novo* drug design [1, 2]. The recent surge in proteins identified as targets of pharmacological interest, a result of major advances in experimental structure determination [3, 4] and high-throughput modeling [5], has created a demand for new, drug-like ligands. Although significant advances have been made in computational approaches to ligand design [6–8], there continues to be an interest in novel design approaches.

Peptides offer tremendous therapeutic potential for myriad diseases [9]. However, the reality of peptidic drugs has been slow to arrive owing to the poor ADME (absorption, distribution, metabolism, excretion) profiles of most peptides. Strategies to overcome the aforementioned limitations include drug delivery of peptides [10], and the incorporation of metabolically robust residues (e.g., D amino acids) into peptide sequences.

Several research groups are currently trying to develop methodologies for the design of peptidic drugs. Examples of effective methodologies are structure-based drug design [11], whereby the design process is tackled as an engineering problem, and high-throughput screening (HTS) of numerous compounds from combinatorial libraries against a known target [12]. We propose herein a new *in silico* approach, dubbed ENPDA (Evolutionary structure-based *de Novo* Peptide Design Algorithm), that is a hybrid of the two aforementioned strategies; it allows the screening

*To whom correspondence should be addressed. Phone: +34-934037125; Fax: +34-934037126; E-mail: egiralt@pcb.ub.es of large numbers of candidate peptides that are derived from a semi-rational process implying evolutionary computation.

We use evolutionary algorithms to generate potential peptide ligands of a given protein, by minimizing the docking energy between the candidate peptide ligand and a user-defined area of the target protein surface, or *surface patch* [13]. To achieve this goal, an algorithm must address two main tasks. First, a competent search method must be provided to explore this high-dimensional chemical space. Second, the search space (i.e., the set of all algorithmically treatable molecules) must be divided into regions of higher and lower quality to allow the prediction of desired properties [14]. In this paper, we present results from testing four different evolutionary algorithms as candidates for the search task: (1) Darwinist genetic algorithm (GA) – also known as the original GA [14], (2) Lamarckian genetic algorithm (LGA) [15], (3) population-based incremental learning (PBIL) [16], and (4) Bayesian optimization algorithm (BOA) [17].

The evaluation focused on ordering the search space into regions of higher and lower quality by implementing two different heuristics – one of which emphasized speed, the other of which emphasized accuracy – to calculate the fitness (docking energy) of each individual (peptide) proposed by the evolutionary algorithms. We used AutoDock 3.0.5 [18] to execute the docking calculations required by the heuristics.

We then tested the developed methodology in several test cases: the proteins prolyl oligopetidase, p53, and DNA gyrase. The results obtained are encouraging. Some of the new ligands designed *via* computational methods have better docking energies than those peptides designed using a purely chemical-knowledge based approach. Moreover, we carried out a validation process based on the known complexes of the protein MHC H-2K^b with peptide ligands.

Related work

Several approaches to computer-based ligand design have previously been reported, including growing [19–22], linking [23–25], physico-chemical property guided search [26, 27], and experimental-aided evolution [28, 29]. In the growing strategy, the process starts from a seed structure that has been pre-placed on the surface patch. The user then designates certain growth sites on the seed structure, at which point the program tries to replace each growth site with a candidate fragment. The newly formed structure serves as the seed structure for the next growing cycle. In the linking strategy, the process also starts form a pre-placed seed structure. However, in this case, the structure consists of several separate pieces that have been positioned to maximize interaction with the target protein. The pieces grow simultaneously, and the program ultimately tries to link them in an acceptable way. This process continues until all of the pieces have been unified into a single molecule. In the physico-chemical approach, the main goal is to obtain a peptide exhibiting an optimal set of physico-chemical properties associated with known good ligands. A peptide is then designed by considering this obtained set of physico-chemical properties. Finally, in experiment-aided evolution approaches, an evolutionary algorithm evolves a set of molecules that are subsequently synthesized and experimentally evaluated.

Schneider et al. [30] developed an evolutionary algorithm for optimizing peptides by computing their fitness using neural networks. The difference between Schneider's approach and our approach is that their tool requires an initial *seed* compound, or pre-optimized lead peptide compound. In our case, we design peptides – *de novo* – without needing any seed to start the search, avoiding any initialization bias.

ADAPT [31] is a program for the total design of small organic molecules. The underlying mechanism uses docking as part of the fitness measure. The main difference between ADAPT and our approach is that our system is specifically optimized for the synthesis of sequential compounds and, more specifically, much larger compounds than those for which ADAPT has been optimized.

Previously reported strategies for peptidic ligands are only amenable to sequences based on L-amino acids. This limitation could thus yield compounds with high protease liability, and consequently, poor bioavailability [32]. Overcoming this drawback was another motivation for the title project. Although we are currently working with natural amino acids (L-amino acids), the framework has been designed to accommodate

D-amino acids. Albeit results with D-amino acids are not presented here, it should be noted that the algorithms and the evaluation of ligand energy function independently of peptide stereochemistry.

Methods

ENPDA is based on the use of evolutionary algorithms; several of which were explored for the present work as detailed below. The evaluation of fitness values (i.e., the docking energy of a test peptide to a given surface patch) is especially difficult in our case due to the fact that the high flexibility of peptide ligands implies a huge conformational space to be explored. As a response to this challenge we developed two different heuristics that are also described below.

Evolutionary computation

Evolutionary computation is currently being applied to several areas of chemoinformatics [33], and has recently been utilized in drug and compound library design [25, 27, 34–38]. However, despite the numerous evolutionary computation tools available, those used in drug design are primarily genetic algorithms and to a lesser extent, Lamarckian genetic algorithms [18].

Evolutionary algorithms are ideal for cases in which deterministic or analytic methods fail, for instance, problems in which the underlying mathematical model is not well defined or the search space is too large. These obstacles are commonly encountered in our research. The mathematical model being optimized in the present work, provided by AutoDock 3.0.5, is based on docking and is imperfect [39]. Also, the search space is too big to be systematically explored and, moreover, each docking energy computation is a very slow process. The appeal of using evolutionary algorithms for this model becomes apparent when one considers that docking a hexapeptide onto a protein surface may normally require more than 30 min of calculations on a 1.60 GHz Pentium IV. These considerations have also guided work previously reported by other researchers [21, 22, 25, 40].

Evolutionary algorithms are population-based search methods. However, the solutions (in our case, peptide ligands) must be represented in a way that the evolutionary algorithm can handle. In the jargon of evolutionary algorithms, solutions are known as individuals, and they can be represented by *chromosomes*, although other forms of representation are available [41]. Each chromosome is a sequence of *genes*. In our case a chromosome is a peptide, and each gene represents an amino acid. The position of a gene has in a chromosome corresponds to the position of an amino acid within a peptide. In our case the genotype, the amino acid sequence of a peptide, is distinct from the phenotype, the 3D structure of the peptide when bound to a protein surface patch.

The remainder of this section describes the evolutionary algorithms used.

Darwinistic genetic algorithm

The Darwinistic genetic algorithm (GA) is the original genetic algorithm proposed by Holland [14]. The Darwinistic GA evolves a population of individuals by progressively adapting them to the environment (i.e., the problem being optimized, which in our case is the minimization of the docking energy between a peptide ligand and a target protein). During the evolutionary process, individuals are selected, crossed over and mutated to generate new individuals. GA's consist of five steps: (1) initialization, (2) evaluation, (3) selection, (4) crossover, and (5) mutation (see Figure 1).

For the first step, a population can be initialized in several ways, however, the usual method is a random function. Afterwards, the GA enters a loop formed by the evaluation, selection, crossover, and mutation phases. This evolutionary loop runs until the end criteria are satisfied.

The loop commences with the evaluation, whereby the fitness of each individual is calculated using a function – or model – to assess the degree of adaptation of that individual to the environment.

Once the fitness of each peptide is computed, the best individuals are selected by the *Rank Selection* method [42], which ranks individuals by fitness. Then *Stochastic Universal Sampling* [43] selects the individuals to be crossed over, by mapping individuals onto contiguous segments of a line, such that the size of each individual segment is proportional to its rank. Pointers are then placed over the line at regular intervals – one for

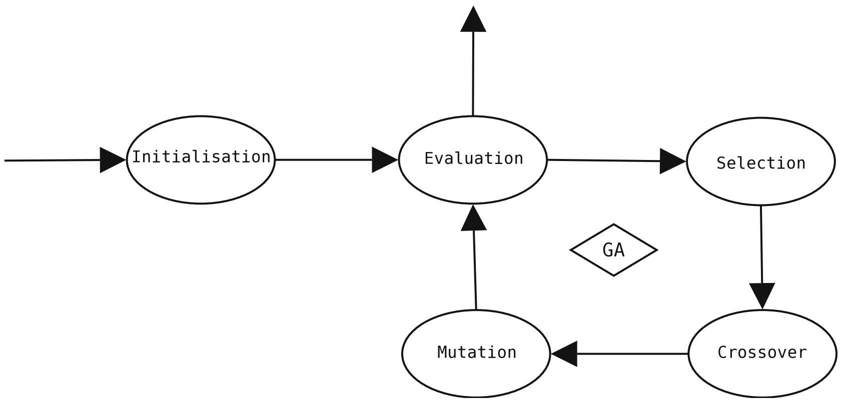

Figure 1. General overview of GA steps.

each individual to be selected. This method has been described as a good selection operator for small populations [43].

Once the individuals that are going to be reproduced have been selected, crossover begins. Two parents are chosen randomly from the selected individuals. A random cut point is generated in each parent, at which tail fragments are exchanged. Thus one of the offspring will be comprise the head fragment of the chromosome of one parent and the tail fragment of the chromosome of the other parent, and *vice-versa* for the second individual.

The following step in the loop is mutation. Small, random changes are introduced into the population with a pre-defined probability. Mutation introduces genetic diversity into the search, assuring that all the search space could be explored.

The next generation is built before restarting the loop. Every new generation contains a certain percentage of individuals selected from the high fitness members of the previous generation. This technique is known as *elitism* [44], and the percentage is normally set to one individual per generation in order to avoid premature convergence of the population.

Lamarckian genetic algorithm

The Lamarckian genetic algorithm (LGA) [15] follows a similar life cycle to that presented above. The only difference is the introduction of a local search applied to each individual before being selected. Figure 2 shows the steps of this algorithm. Throughout this description we will focus on the local search applied in this algorithm.

Local search implementation can be performed in many ways. In our work we used a (1 + 1) *evolutionary strategy* (ES) [45] – see Figure 3. This ES performs a mutation over an entire chromosome. The new individual is then evaluated and compared to its parent, and the best of the two proceeds to the next generation. This process is repeated for each individual as many times as indicated by the user.

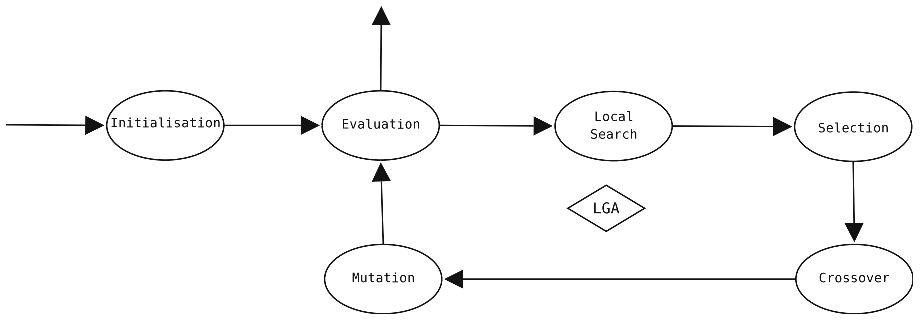

Figure 2. General overview of LGA steps.

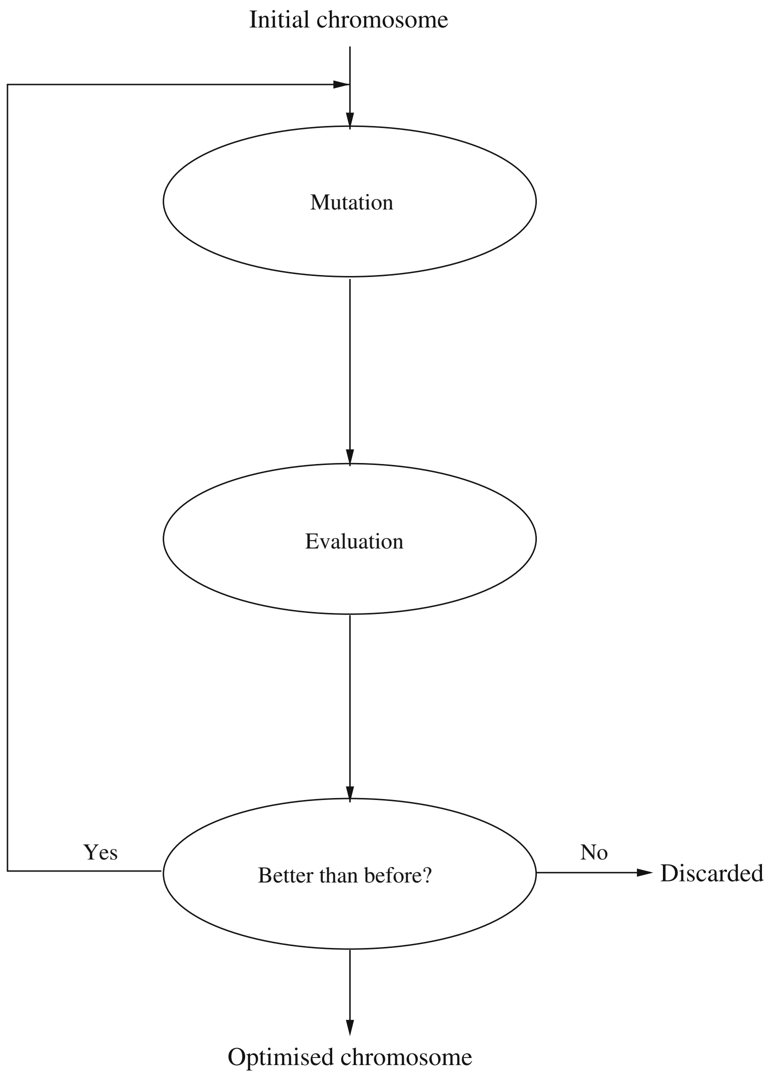

Figure 3. Steps performed in an ES.

In the ES mutation stage, a *Gaussian* mutation is performed. The chromosomes that represent each one of the proposed peptides are rebuilt. Each old gene is enhanced by another gene that encodes the standard deviation to be used by the Gaussian mutation. Hence, in the mutation step, for each gene representing a peptide amino acid, a random value with a normal distribution is taken. This value corresponds to the amino acid gene in the new individual. Figure 4 summarizes this step. It should be noted that since we are dealing with amino acids as categorical values, the only functions of the Gaussian mutation are to detect if an amino acid is favorable, and if not, to replace that amino acid.

In ES there are two kind of mutations. The gene values are mutated as described above, and in addition, the standard deviation values can also be mutated. This second mutation is carried using the following formula:

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \qquad (1)$$

σ_i' is the value of the ith standard deviation gene of the new individual, and is composed of two independent terms. The first of these terms ($\tau' \cdot N(0,1)$) takes the same value for every mutation in the step, while the second s ($\tau \cdot N_i(0,1)$) is computed individually for each standard deviation gene mutation. σ_i represents the value of the ith standard deviation parent gene. $N(0,1)$ is a random

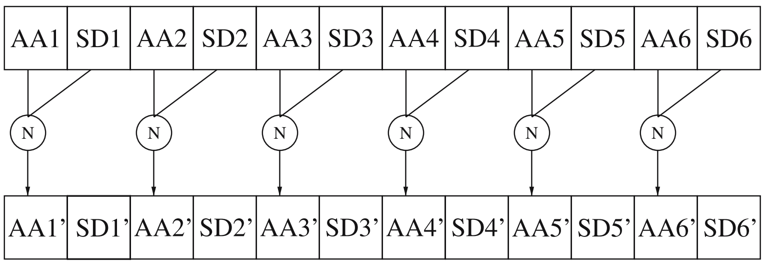

Figure 4. Gaussian mutation performed over the genes representing amino acids. The AA$_i$ term is the value of the amino acid gene representing the *i* amino acid of the peptide. The SD$_i$ is the gene representing the standard deviation of the amino acid gene *i*. Those terms with the prime symbol are the values of the new individual built using AA$_i$ as expectation of the Gaussian function (N) and SD$_i$ as standard deviation.

number with a normal distribution with a mean of 0 and a standard deviation of 1. Finally, the parameters τ and $\tau\prime$ are recommended to be proportional to (2) and (3) [46].

$$\tau \propto \frac{1}{\sqrt{2\sqrt{n}}} \tag{2}$$

$$\tau' \propto \frac{1}{\sqrt{2n}} \tag{3}$$

where n is the number of amino acid genes in the chromosome.

Population-based incremental learning

Baluja and Caruana [16] defined the population-based incremental learning (PBIL) algorithm as an abstraction of a GA by explicitly maintaining the statistics contained in a GA population. In simple words, PBIL is a GA having a statistical vector in place of a crossover or mutation. By statistical vector we mean a data structure in which we compute the frequency with which each gene allele appears in the best individuals of the population. If more than two alleles are possible, PBIL generates a matrix computing the appearance frequencies for each vector and allele. The new individuals are thus created by sampling the inferred statistical matrix. Figure 5 shows the PBIL life cycle.

PBIL performs incremental learning, meaning that the statistical matrix is not built from scratch for each generation. The values of this table are computed as follows:

$$\forall_i \forall_j V_{\text{prob}}[i][j] := ((1-\alpha) * V_{\text{prob}}[i][j]) + (\alpha * V\prime_{\text{prob}}[i][j])$$

where, $V_{\text{prob}}[i][j]$ is the matrix in which the appearance frequency is characterized and α is the learning rate. It should be noted that if α is set to 1 for each generation, the matrix is built from scratch. $V\prime_{\text{prob}}[i][j]$ is the matrix containing the appearance frequency for the present generation only. $\forall i$ refers to all of the genes in the chromosome, and $\forall j$ represents the possible values of each gene.

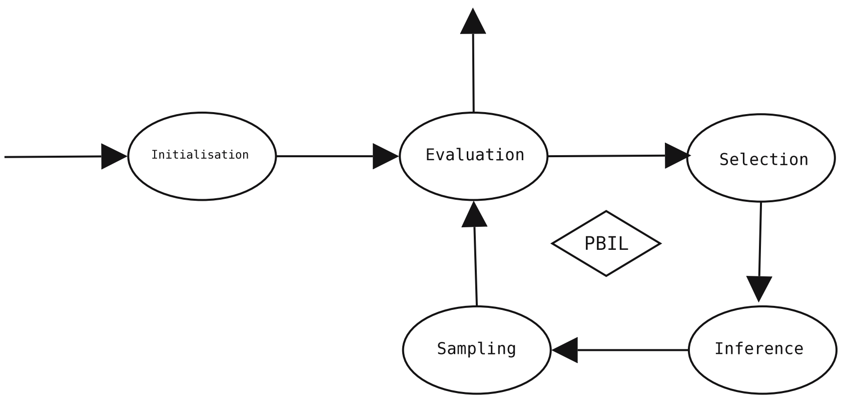

Figure 5. Life cycle of the PBIL algorithm.

A new individual is generated from sampling of the inferred distribution. We performed this step as many times as there are individuals in the population.

This algorithm is based on the assumption that values which can adopt the genes – i.e., the alleles – are independent. This underlying premise is not necessarily true in real drug design problems. For this reason we decided to implement another estimation of the distribution algorithm that correlates the frequencies with which each genes in the chromosome appears.

Bayesian optimization algorithm

The Bayesian optimization algorithm (BOA) [17] is quite similar to PBIL. Its life cycle is identical, but instead of using a statistical matrix to perform the population inference it uses a *Bayesian network* [47, 48].

A Bayesian network is an annotated, directed graph that encodes probability relationships among distinctions of interest in an uncertain-reasoning problem. In our work, it encodes the conditional probabilistic relationships among the possible values of the amino acids of each gene. The Bayesian network is modeled with only the best individuals of the population. This task is performed via the K2 algorithm in tandem with Dirichlet metrics [49].

The K2 algorithm is a Bayesian network building algorithm. It is a greedy algorithm that tries to maximize the Dirichlet metrics of a Bayesian network by adding, removing, or changing the direction of the edges that interconnect the nodes. Once K2 has found an optimal network topology, conditional probabilities of the conditional relationships represented by the edges can be computed. For example, if in a given network, node *gene1* is connected to the node *gene5*, then a conditional relationship between these two genes has been observed. An example of a Bayesian network is illustrated in Figure 6.

In the sampling stage, the conditional relationships inferred in the Bayesian network must be taken into account when creating the individuals of the next generation. A detailed explanation of the sampling algorithm used for this task can be found in [50].

Heuristics

As stated above, the use of evolutionary algorithms requires a tool to structure the search space into regions of higher and lower quality. We

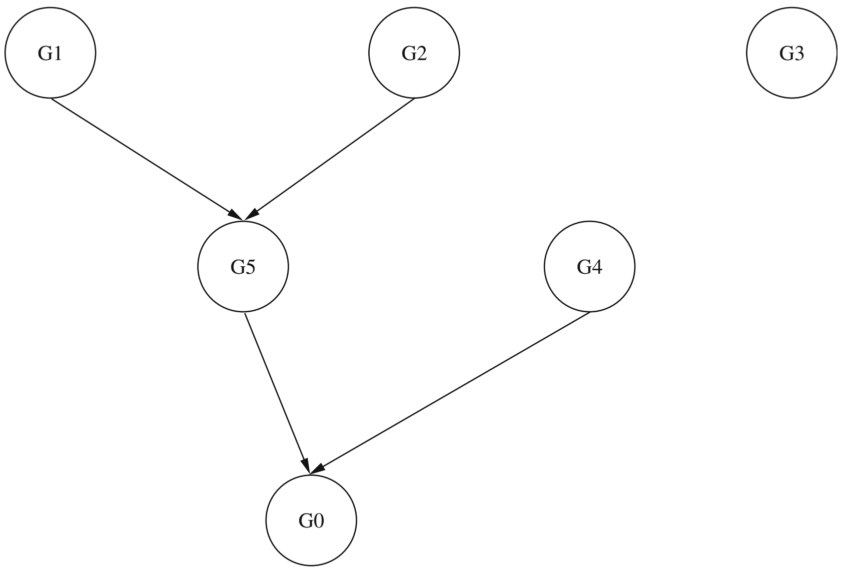

Figure 6. A graphical example of a Bayesian network. Each node represents one gene in the evolutionary algorithm. In this example, the value of *G5* depends on the values of *G1* and *G2*, the value of *G0* depends of the values of *G5* and *G4*, and *G3* is independent.

developed two heuristics to compute the energy for docking each peptide to a user-defined surface patch. These calculations provide us with an estimation of the binding energy between the peptide and the protein. The two heuristics are complementary. Whereas the first one is quicker, the second is more accurate, and, hence, computationally more expensive. The researcher can choose the heuristic according to the needs of the problem in question.

Docking

Docking algorithms are *in silico* tools that search for the best mode of interaction between a small, flexible ligand and a large, usually rigid, macromolecular receptor. Docking programs usually consist of two key elements: an energy function to be minimized, and a search algorithm. The search algorithm is used to determine which three-dimensional structure minimizes the energy function. However, as an increase in ligand flexibility results in an exponential increase in search space, the search algorithm is rarely able to find the global optima. Moreover, the energy functions are typically an approximation of the real binding energy, which could lead to artifactual results in the docked complexes. However, recent reviews of the topic are encouraging in terms of docking [51].

In the specific case of AutoDock 3.0.5 [18], the energy value being minimized is the sum of the *final intermolecular energy* and the *final internal energy* of the ligand. In our work, the small flexible ligands are the peptides proposed by the evolutionary algorithms, and the macromolecular receptor is the surface patch of the protein defined by the researchers.

The heuristics first entail preparation of the information needed by AutoDock, which is then used to perform the docking experiments. The program's output is subsequently used to extract the docking energy. We used AutoDock set at the majority of its default parameters. We used an LGA as a search algorithm within AutoDock, with 50 individuals per generation, 10,000 generations, 300 steps of local optimization and a maximum of 2,500,000 energy evaluations. The process is independently executed (i.e., with different initializations for each docking experiment) ten times for each docking run.

Systematic exploration of the search space is impractical because of the time required for docking calculations. The computing expense is a consequence of two factors: peptide size and peptide flexibility. The former stems from the fact that the peptides we designed are six amino acids in length, and any shorter peptides would be useless for our application, (i.e., to act as protein–protein interaction inhibitors). The latter is due to the fact that a high degree of flexibility had to be introduced into the peptides to obtain reliable results.

Rapid heuristics

The more rapid heuristics only involves one docking run, which itself includes ten docking experiments, however, as explained below, the more accurate technique involves five. There are five stages in the heuristics: (1) three-dimensional reconstruction, (2) energy minimization, (3) flexible angle definition, (4) docking and (5) Boltzmann averaging of docking energies.

Three-dimensional reconstruction

The internal representation of the peptides in the evolutionary algorithms is not a three-dimensional structure, but rather a sequence of amino acids. We developed a program in NAB language [52] which reads these sequences and outputs a PDBQ file (PDB with charges) [53] that describes the extended structure of the peptide and can be subsequently used for docking experiments.

Energy minimization

The next step is to perform a short energy minimization over the extended structure obtained in the previous step. We implemented this into the program developed for the three-dimensional reconstruction [52], using a conjugate gradient minimization until the root-mean-square of the components of the gradient was less than 1.0.

Flexible angles definition

Before starting the docking calculations we redefined our ligand (peptide) to be flexible, by fixing the backbone but providing flexibility to the side chains. To perform this task we use AutoTors, which is an auxiliary script of AutoDock.

Docking
Using the ligand built in the previous stages and the user-defined surface patch, a docking run takes place using the previously described configuration.

Boltzmann averaged binding energy
Finally, we computed a Boltzmann binding energy by averaging the most stable structure found in each of the 10 runs of the docking algorithm. The value obtained in this step is the fitness value for the peptide in question.

Accurate heuristics
The second heuristics implicitly implements backbone flexibility, making it more accurate than that described above, in which the backbone was fixed after the energy minimization stage. The second heuristic involves five different dockings, each of which corresponds to a different peptide backbone structure. Once the docking experiments have been carried out, we keep the most stable of the structures generated. This heuristics comprises five steps: (1) secondary structure prediction, (2) rotamers construction, (3) definition of flexible angles, (4) docking, and (5) Boltzmann averaging of docking energies.

Secondary structure prediction
We first use the Chou-Fasman method [54] for secondary structure prediction of peptides, which takes into account their amino acid sequence. This method assigns to each conformation (α-helix, β-strand or random coil) a number representing the "probability" of adopting that conformation.

Rotamer construction
Using the numbers obtained from the Chou-Fasman prediction, we built five rotamers of the same peptide. In each rotamer, all the ϕ and ψ angles of the peptide are randomly chosen from the area of a Ramachandran map which represents the secondary structure being built. Angle ω is fixed at 180°.

Flexible angle definition
Using AutoTors, each side-chain of each of the five peptide structures built is defined as flexible. Hence, in each docking experiment, the backbones are fixed and the side-chains are flexible.

Docking
A single flexible docking run is carried out for each rotamer built. Each of the docking calculations is carried out as in the first heuristics. As each docking run implies ten independent docking experiments, fifty docking experiments are ultimately performed in this stage.

Boltzmann averaged binding energy
Finally, we determined which of the rotamers is most stable when docked, then calculated the Boltzmann averaged binding energy for this rotamer as previously described.

Results

Reported here are the results from tests in which ENDPA was used to design ligands for three proteins of high therapeutic interest. Making an exhaustive comparison of the performance of the evolutionary algorithms developed is beyond the scope of this paper. Moreover, said comparison would be impossible using typical statistical analysis due to the high computational cost of ENDPA – a typical run takes around 2 weeks using a 8 process elements of an Origin 3400. In this section, we explored the performance of PBIL, LGA and BOA in three different problems. We considered it unnecessary to test GA, as it behaves similarly to LGA.

We tested ENDPA on a wide range of proteins, namely POP, p53, and DNA gyrase. The surface patch of POP is located inside a long tunnel, while the surface patch of p53 is a large hydrophilic surface, and that of DNA gyrase is an area of DNA interaction. To ensure a fair comparison, we carried out four experiments for each of the systems studied. In the first, we performed docking experiments with peptides designed manually by researchers in our group, as well as with some natural ligands. In the second experiment, we ran ENPDA. In the third, we evaluated as many random peptides as each algorithm had evaluated in each subsystem and we compared the top-five random peptides to the peptides obtained by the other sources. And finally, we evaluated five totally random peptides. In this manner we were able to compare results derived from three very different sources: human-guided methods, artificial intelligence and random search. It is worth to say that the random peptides were built using an uniform residues frequencies among the selected subset of amino acids for each system.

Since ENPDA allows to select the amino acids that can be used in the peptide design, we selected a small reduced set for each system. It should be remembered that the aim of the present work is the identification of peptide ligands starting from scratch. Hence, optimization of the peptides will be the subject of future work.

In addition, in order to compare and validate the peptides obtained with ENPDA, we have designed ligands for a protein that has been reported to co-crystalize with peptides. The chosen system is the protein MHC H-2K^b complexed to the high affinity peptide dEV8. Hence, we wanted to test if ENPDA was able to design ligands similar in sequence and secondary structure to peptides having a known affinity for MHC H-2K^b.

Prolyl oligopeptidase

Prolyl oligopeptidase (POP, EC 3.4.21.26) is a cytosolic serine peptidase characterized by oligopeptidase activity. POP hydrolyzes peptide bonds at the C-terminal site of prolyl residues within polypeptides of less than 30 amino acids [55]. POP is highly conserved in mammals and distributed throughout the body, with higher concentrations in the brain [56]. The three-dimensional structure of POP reveals two distinct domains: catalytic and structural. The catalytic domain contains residues Ser554, His680, and Asp641, which form the catalytic triad of the protease. These residues are essential for the catalytic activity of the enzyme and are located in a large cavity at the interface of the two domains [57].

Based on this information and analysis of the crystal structure of POP, we defined a surface patch around the residues of the catalytic center Ser554, His680 and Asp641. The surface patch also comprises the cavity formed by the structural domain that corresponds to the interior of the β-propeller domain. Inside this cavity the substrate interacts with many residues of the enzyme and is directed to the active site. The surface patch is ample enough to accommodate peptides composed of six residues such as those tested in this study.

POP is involved in the maturation and degradation of proline-containing neuropeptides that are implitated in learning and memory [58]. Physiological studies of human plasma have shown that POP activity is increased in mania, schizophrenia, post-traumatic stress and anxiety, while it is decreased in depression, anorexia and bulimia nervosa [59, 60]. Modulation of POP activity with specific peptide inhibitors could be useful for the treatment of these diseases.

The initial POP structure used was that corresponding to PDB code 1QFS. A brief energy minimization was performed and hydrogen atoms were added to the structure using the program Insight II.

Ligand peptides found in the literature

Since the peptides found in the literature were longer than six amino acids, we used fragments of known POP substrates to perform the experiments. As POP hydrolyzes bonds adjacent to prolines, only fragments containing a minimum of one proline residue were chosen. We tested the following peptidic fragments of known POP substrates: neurotensin fragment (KPRRPY), substance P fragment (RPKPQQ), oxytocin fragment (QNCPLG), and vasopresin fragment (QNCPRG). Furthermore, the following peptides cited as POP inhibitors were also tested: GKPPIG, GKPPVG, GVEIPE, GYPIPF, HLPPPV, LLSPFW, LSPFWN, MPPPLP, MTPPLP, TPPLPA and SPFWNI. Table 1 shows the energies of all these peptides.

Table 1. Docking energy of fragments of POP inhibitors described in the literature.

Peptide	Docking energy
LSPFWN	−9.5
QNCPRG	−8.9
MPPPLP	−8.9
HLPPPV	−8.5
QNCPLG	−8.0
TPPLPA	−7.9
RPKPQQ	−7.5
SPFWNI	−7.5
MTPPLP	−7.4
GKPPIG	−7.2
GKPPVG	−7.1
LLSPFW	−6.3
GYPIPF	−5.8
GVEIPE	−5.1
KPRRPY	−3.6

In silico designed peptides

In this stage we ran an LGA with 50 individuals per generation, 10 generations and 2 local optimization steps. We used a crossover probability of 0.9, a mutation probability of 0.05 and an elitism of one individual per generation. The more accurate of the two heuristics was used, and a total of 975 peptides were explored using a reduced set of amino acids: alanine, serine, proline, tryptophan, glutamic acid, isoleucine and arginine. Table 2 shows the energies of the best five individuals found by the algorithm. All of these peptides are proline rich, in the same manner than the natural inhibitors. Figure 7 displays an image of POP interacting with one of the proposed peptides.

Random peptides

We also created 975 random peptides and tested their docking energies using the accurate heuristics. We used the same amino acid set used in the *in silico* process above. Table 3 shows the top five peptides found. Table 4 lists five additional completely random peptides.

Table 2. Docking energy of the best five individuals found by the algorithm for the POP.

Peptide	Docking energy
WWPWPP	−13.7
WWPSWA	−13.2
WSPSWP	−13.0
PWPEWA	−13.0
WWPWSP	−12.9

Table 3. Docking energy of the best five peptides found for POP among 975 random peptides.

Peptide	Docking energy
SWPWWS	−13.0
WAPWWS	−12.1
WAPWAA	−12.1
SWPAWS	−12.0
PARWEP	−12.0

p53

p53 is a transcription factor involved in different cellular functions, such as cell cycle control, programmed cell death (or apoptosis), and differentiation. The p53 gene was the first tumor-suppressor gene to be identified, and is dysfunctional in most human cancers. In ca. 50% of cancers, p53 is deactivated as a direct result of mutations in its corresponding gene. The deactivation of the protein may also stem from mutations in genes whose products interact directly with p53 or are involved in the p53 pathway. In other cases, p53 is shut down upon binding of viral proteins to the protein itself [61].

Table 4. Docking energy of five random peptides bound to POP.

Peptide	Docking energy
SSWWRP	−9.6
AIPPWE	−8.2
AASAIP	−7.0
PAIRAA	−6.9
RWSRSP	−5.6

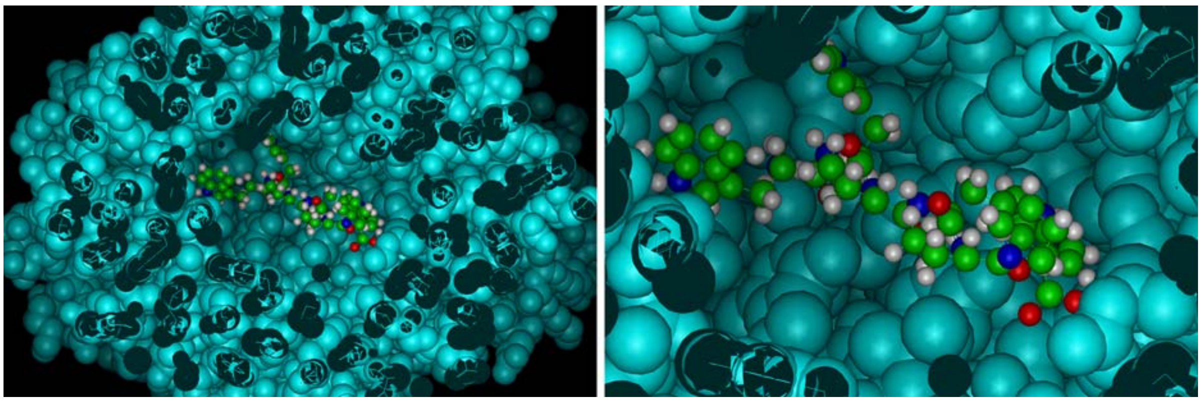
Figure 7. Proposed peptide, WWPWPP, docked in the user-defined surface patch of the protein prolil oligopeptidase.

p53 can be divided into four domains: (1) an N-terminal trans-activation domain, (2) a DNA-binding domain (DBD), (3) a tetramerization domain (TD), and (4) a C-terminal regulatory domain. The functioning of p53 is mediated by protein-DNA and by protein-protein interactions. The tetramerization domain plays an essential role in the protein's activity as only the tetramer is active [62]. This domain can be described as a dimer of dimers. Each primary dimer is formed by an antiparallel β-sheet and two antiparallel α-helices, and the two dimers are arranged orthogonally [63]. Peptides that recognize this tetramerization domain and stabilize its native conformation could be of great interest in cancer research.

For our work with p53, we chose the structure corresponding to PDB code 1SAL, which was originally derived from NMR studies. Hydrogen atoms were then added to the structure using the Insight II software.

Peptides designed by researchers

In our group, we have studied recognition of the p53 tetramerization domain by a tetraguanidinium compound [64]. Based on this molecule, we have designed and synthesized a peptide which is also able to interact with the domain. The recognition can be attributed to arginines in the peptide, which are known to favor interaction with the carboxylate-rich surface of the protein. In order to improve the affinity of the peptide for the protein surface, we designed a peptide library in which arginine was widely substituted for other residues, primarily through modifications of the chain length and/or functional group. The library was synthesized and is currently evaluated in our laboratory. Table 5 shows the docking energies of the best five peptides found in the 47 peptide library. More details on the design and evaluation of these peptides can be found in [65].

Table 5. Docking energy of the top-five peptides designed as p53 ligands.

Peptide identification	Docking energy
Rpa3R_46	−11.1
Rpa4_47	−10.9
Oxaa_33	−10.2
Rab4_51	−9.7
Dapa3R_20	−9.5

In silico designed peptides

In this stage we ran PBIL using 50 individuals per generation, and 20 generations. The learning rate (α) was set to 0.5. The best 20 individuals from each generation were taken to build the statistical matrix and an elitism of one individual was allowed. Accurate heuristics was used, and a total of 715 peptides were explored based on the following amino acids: alanine, serine, tryptophan, glutamic acid, isoleucine and arginine. Table 6 lists the energies of the five best individuals found by the algorithm. Figure 8 shows p53 interacting with one of the proposed peptides.

Random peptides

In this stage we created 715 random peptides and tested their docking energy using the accurate heuristics. We used the same amino acid set as in the *in silico* design process. Table 7 shows the best five peptides found among the 715 and Table 8 shows five fully random peptides.

DNA gyrase

The enzymes responsible for maintaining the topological state of DNA are termed DNA topoisomerases [66], although type II topoisomerases are also known as DNA gyrases. These proteins influence basic *in vivo* processes such as RNA transcription as well as DNA replication and recombination. DNA gyrases are essential to all cells and, as such, are important targets for many antibacterial drugs.

Many discoveries regarding gyrase function, such as the determination of the nature of the ATP binding site, as well as the relationship between gyrase activity and DNA replication and transcription, have been made through the use of

Table 6. Docking energy of the best five individuals found by the algorithm for p53.

Peptide	Docking energy
WWPWWW	−13.3
WWWWWW	−13.3
AWWWWW	−13.2
WWWWWA	−13.1
AWRWWW	−12.8

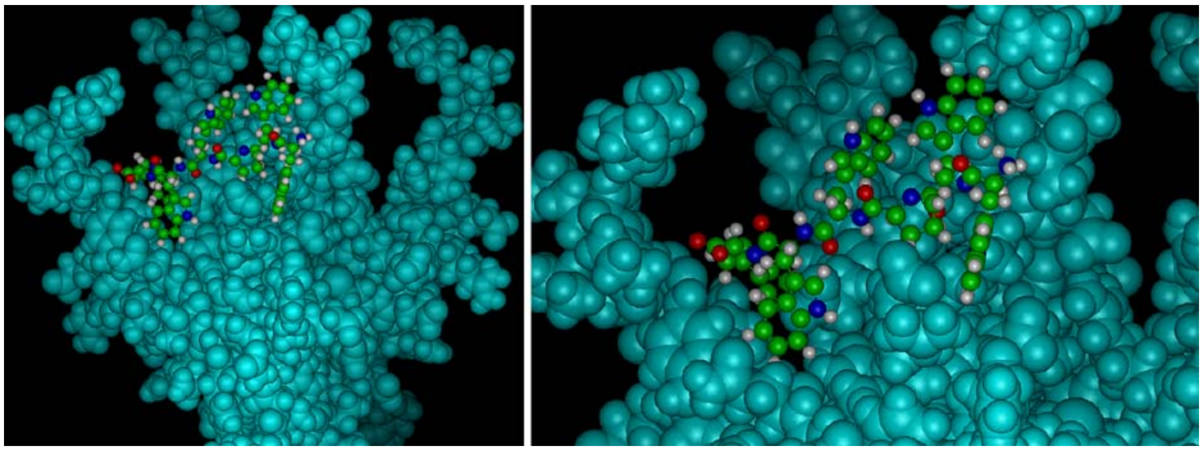

Figure 8. Proposed peptide, `WWPWWW`, docked in the user-defined surface patch of the protein p53.

Table 7. Docking energy of the best 5 peptides found for p53 among 715 random peptides.

Peptide	Docking energy
WIWWWW	−11.2
SWWWWS	−11.1
AWWEAW	−10.7
SERWWW	−10.6
WWWRWE	−10.5

Table 8. Docking energy of five random peptides bound to p53.

Peptide	Docking energy
WEAASW	−9.2
WIESSE	−6.5
RARRWA	−5.9
SWIEEE	−5.7
ERIAAR	−5.4

gyrase inhibitors [67, 68]. This is a crucial step to designing new antibiotic drugs that might prevent bacterial infection and overcome resistance.

The initial structure of the protein used was that corresponding to the PDB code 1AB4. In the interest of preserving computing resources, we only used the 59 kDa fragment.

In silico designed peptides

In this stage we ran a BOA using 100 individuals per generation, and 40 generations. The best 50 individuals from each generation were used to build the Bayesian network, and an elitism of one individual was allowed. We applied rapid heuristics and explored a total of 352 peptides based on alanine, serine, tryptophan, glutamic acid, isoleucine, and arginine. Table 9 shows the energies of the best five individuals found. Figure 9 shows DNA gyrase interacting with one of the proposed peptides.

Random peptides

In this stage we created 352 peptides, the same number explored by BOA to obtain a solution, and tested their docking energies using the rapid heuristics. We used the same amino acid set as in the *in silico* design process. Table 10 shows the best five peptides found among the 352. In Table 11 the energies of five completely random peptides bound to DNA Gyrase are listed.

MHC H-2K^b

This new peptide design methodology was applied to the murine H-2K^b major histocompatibility complex (MHC) class I molecule. MHC molecules

Table 9. Docking energy of the best five individuals found by BOA for the DNA gyrase.

Peptide	Docking energy
WWWWWS	−3.7
AAWWRA	−3.4
RWWEWW	−3.4
WAWAWS	−3.3
AAWWRE	−3.3

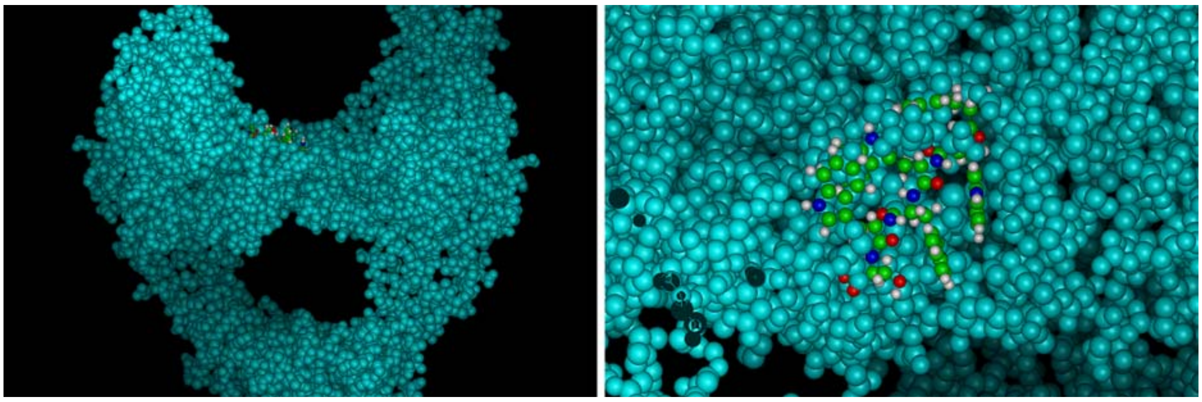

Figure 9. Proposed peptide, WWWWWS, docked in the user-defined surface patch of the protein DNA gyrase.

Table 10. Docking energy of the best five peptides found for DNA gyrase among 352 random peptides.

Peptide	Docking energy
WWSAES	−3.1
SAWWRS	−3.0
SWSWWA	−3.0
WWSSAS	−2.9
EWWWEA	−2.9

Table 11. Docking energy of five random peptides bound to DNA gyrase.

Peptide	Docking energy
WASRSW	−1.4
SSSEWS	−1.4
WSAEEW	−1.3
AAARAE	−0.4
ISWEII	1.7

are highly polymorphic cell-surface proteins that present antigenic peptides for recognition by T-cell receptors, thereby triggering an immune response. The high resolution crystal structure of the protein H-2K^b complexed to the high affinity dEV8 ligand (EQYKFYSV) was selected to compare the docking structures of the predicted hexapeptide ligands. The 1LEG PDB code was used to obtain the structure of the H-2K^b protein and dEV8 ligand [69].

The ENPDA program was run using the PBIL algorithm with 50 individuals per generation, and a total of 30 generations. The amino acids used for the design process comprised: alanine, glutamate, glutamine, tyrosine, lysine and phenylalanine. It should be noted that, in this particular validation test, the secondary structure of the peptides was assigned according to the crystallographic coordinates of the EQYKFYSV ligand bound to the protein.

From a structural point of view, the docked structures of the best peptides obtained by the algorithm are similar to the pose of the EQYKFY hexapeptide fragment in the crystal of the complex (Table 12). The root mean square deviation (RMSD) values for the backbone atoms of the five best designed peptides fall below 1 Å. With respect to the side-chain structures, comparison of the QQFFFA designed peptide with the crystal structure of dEV8 peptide reveals a good superposition of the Gln with the Glu at position P1, Gln with the Gln at position P2, the Phe with the Tyr at position P3 and the Phe with the Phe amino acid at position P5 (Figure 10). In the literature [70], high-affinity binding to H-2K^b is described to usually be associated with aromatic residues at positions P3 and P5. Our best five designed

Table 12. Comparison of the binding position of the best five designed peptides ligands to that observed in the crystal structure of the EQYKFY hexapeptide ligand fragment bound to H-2K^b protein.

Peptide	RMSD[a]
QQFFFA	0.80
QFFFFE	0.80
FFFFFA	0.98
QKFFFA	0.81
AFFFFK	0.84

[a]Backbone RMSD values, units in Å.

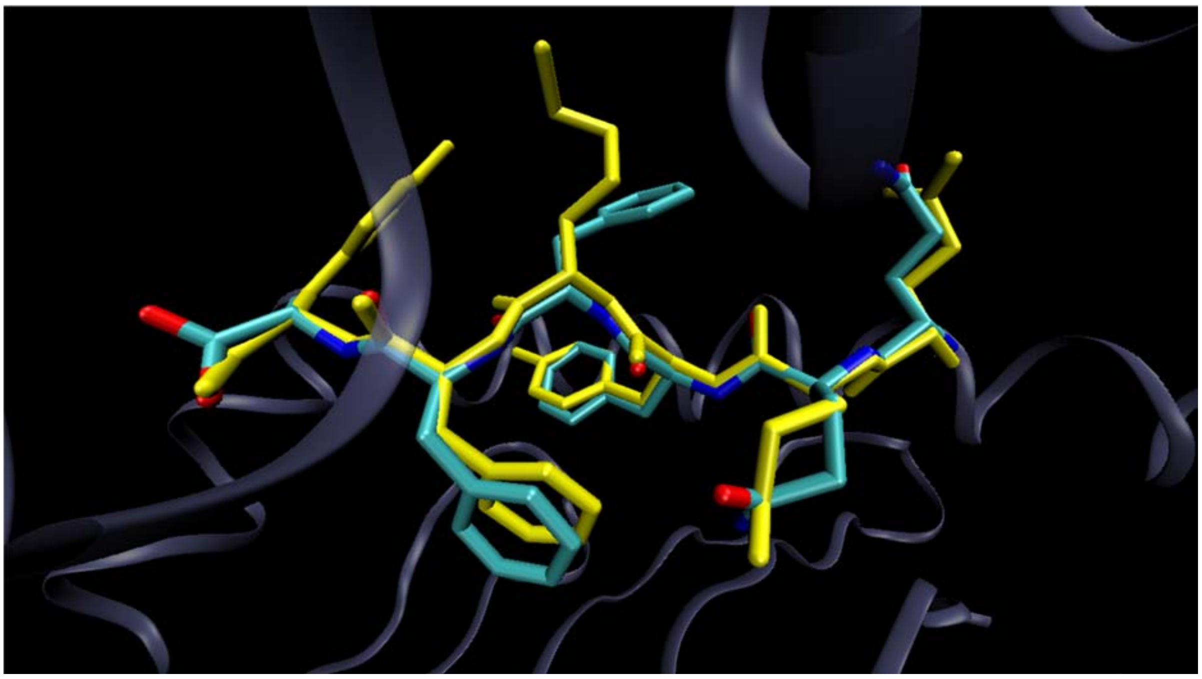

Figure 10. Comparison of the predicted structure of the QQFFFA peptide (atom type color) with the known crystal structure of the EQYKFY hexapeptide ligand fragment (yellow) bound to the H-2K(^{b}) protein.

peptides are in agreement with this experimental trend since these two positions are optimized with the amino acid Phe (Table 12).

Discussion

Recent advances in evolutionary computational methods have allowed us to develop a new approach for the *in silico* design of peptide ligands. We have implemented four evolutionary algorithms to direct this combinatorial search. Due to the computational cost of the heuristics, insufficient data was available to perform a reliable statistical comparison of the performance of these programs in the drug design problems addressed. However, we can make some comments about the general behavior of each algorithm and the quality of the peptides evolved in each individual problem domain. For example, LGA performs better than GA, since the local search performed by LGA over each individual helps build an appropriate sequence. We have also observed, in agreement with the theory [17, 71], that BOA requires a large population to infer a proper Bayesian network. Hence, since our evaluation methods are computationally expensive, we can only use the BOA algorithm when using the *rapid heuristics* that is described in the methods section. Regarding PBIL, we should point out that for simple estimations of distribution assuming gene interaction independence, population diversity is maintained (results not shown). Such diversity ensured that PBIL did not converge prematurely, even when working with small populations. Finally it must be said that LGA produced peptide ligands with very high binding affinities. Nevertheless, this may be an obvious consequence since it is the algorithm that, due to the local search mechanisms, tends to perform the most evaluations.

For three distinct problems, we compared the designed peptides to random peptides, natural ligands or peptides designed by medicinal chemists. The *in silico* designed peptides are generally better in *in silico* comparisons (i.e., they exhibit better docking energies compared to the other peptides). In the validation study carried out on the MHC H-2K^b, we observed that ENDPA is able to identify the most important residues of the molecular recognition described in the literature. We have also observed that our docking procedures are able to obtain a binding structure of the designed ligands and the protein similar to that observed in crystal structures of known high affinity ligands attached to the same protein.

In summary, the present paper reports the development of ENPDA, a new computational evolutionary tool that has been shown to be efficient for the design of high affinity peptidic ligands of protein surfaces *in silico*. Nevertheless, the Achilles heel of the method is the docking step, which is slow and inaccurate. Therefore, our future work plans entail further improvement of our evolutionary algorithms as well as optimization of the docking procedure for relatively large, flexible peptide ligands.

Acknowledgments

The authors thank I. Traus for his creative input on the computational aspects of this work, and Prof. X. Vilasis for his mathematical support in the implementation of Bayesian network learning algorithms. The authors would also like to thank the Parc Científic de Barcelona for providing the computational resources used for this research. This work was partially supported by grants from Fundación BBVA, Fundació Marató TV3 and the Ministerio de Ciencia y Tecnología FEDER (BIO2002-2301 and EET2001-4813), the Air Force Office of Scientific Research, Air Force Materiel Command, USAF (F49620-03-1-0129), and by the Technology Research, Education, and Commercialization Center (TRECC), at the University of Illinois at Urbana–Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the Technology Research, Education, and Commercialization Center, the Office of Naval Research, or the U.S. Government.

References

1. Böhm, H.J., J. Comput.-Aid. Mol. Design, 12 (1998) 309.
2. Kubinyi, H., J. Recept. Signal Transduction Res., 19 (1999) 15–39.
3. Codina, A., Gairí, M., Tarragó, T., Vigueras, A.R., Feliz, M., Ludevid, D. and Giralt, E., J. Biomol. NMR, 22 (2002) 295.
4. Chiva, C., Barthe, P., Codina, A., Gairí, M., Molina, F., Granier, C., Pugniere, M., Inui, T., Nishi, H., Nishiuchi, Y., Kimura, T., Sakakibara, S., Albericio, F. and Giralt, E., J. Am. Chem. Soc., 125 (2003) 1508.
5. Thormann, M. and Pons, M., J. Comput. Chem., 22 (2001) 1971.
6. Ajay, A., Walters, W.P. and Murko, M.A., J. Medical Chem., 41 (1998) 3314.
7. Zou, X., Sun, Y. and Kuntz, I.D., J. Am. Chem. Soc., 121 (1999) 8033.
8. Apostolakis, J. and Caflish, A., Combinatorial Chem. High Throughput Screen., 2 (1999) 91.
9. Loffet, A., J. Pept. Sci., 8 (2002) 1.
10. Malmsten, M., Surfactants and Polymers in Drug Delivery. Marcel Dekker, 2002.
11. Henry, C.M. and Washinton, E., Chem. Eng. News., 79 (2001) 69–74.
12. Pinilla, C., Appel, J.R., Borras, E. and Houghten, R.A., Nat. Medicine, 9 (2003) 118.
13. Jones, S. and Thornton, J.M., J. Mol. Biol., 272 (1997) 121.
14. Holland, J.H., Adaptation in Natural and Artificial Systems. University of Michigan Press, 1975.
15. Krasnogor, N., Studies on the Theory and Design Space of Memetic Algorithms. Ph.D. dissertation at University of the West England, Bristol, 2002.
16. Baluja. S. and Caruana, R., Proceedings of the International Conference on Machine Learning, Morgan Kaufmann (1995) pp. 112–128.
17. Pelikan, M., Goldberg, D.E. and Cantú-Paz, E., Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99, Morgan Kaufmann (1999).
18. Morris, G., Goodsell, D., Halliday, R., Huey, R., Belew, R. and Olson, A., J. Comput. Chem., 19 (1998) 1639.
19. Moon, J.B. and Howe, J.W., Proteins: Structure, Function and Genetics, 11 (1991) 314.
20. Gillet, V.J., Newell, W., Mata, P., Myatt, G., Sike, S., Zsoldos, Z. and Johnson, A.P., J. Chem. Inform. Comput. Sci., 34 (1994) 207.
21. Douglet, D., Thoreau, E. and Grassy, G., J. Comput.-Aid. Mol. Design, 14 (2000) 449.
22. Budin, N., Majeux, N., Tenette, C. and Caflisch, A., J. Comput. Chem., 22 (2001) 1956.
23. Frenkel, D., Clark, D.E., Li, J., Murray, C.W., Robson, B., Waszkowycz, B. and Westhead, D.R., J. Comput.-Aid. Mol. Design, 9 (1995) 213.
24. Böhm, H.J., Prog. Biophys. Mol. Biol., 3 (1996) 197.
25. Wang, R., Gao, Y. and Lai, L., J. Mol. Model., 6 (2000) 498.
26. Mandell, A., Selz, K. and Shlesinger, M., Algorithmic design of peptides for binding and/or modulation of the funcions of receptors and/or other proteins, Patent No. 767460, 2002.
27. Teixido, M., Belda, I., Roselló, X., Gonzalez, S., Fabre, M., Llorà, X., Bacardit, J., Garrell, J.M., Vilaró, S., Albericio, F. and Giralt, E., QSAR Combinatorial Sci., 22 (2002) 745.

28. Weber, L., Wallbaum, S., Broger, C. and Gubernator, K., Angewantde Chemical International Edition English, 34 (1995) 2280.
29. Singh, J., Ator, M.A., Jaeger, E.P., Allen, M.P., Whipple, D.A., Soloweij, J.E., Chowdhary, S. and Treasurywala, A.M., J. Am. Chem. Soc., 118 (1996) 1669.
30. Schneider, G., Schrodl, W., Wallukat, G., Muller, J., Nissen, E., Ronspeck, W., Wrede, P. and Kunze, R., Proc. Nat. Acad. Sci., 95 (1998) 12179.
31. Pegg, S., Haresco, J. and Kuntz, I., J. Comput.-Aid. Mol. Design, 15 (2001) 911.
32. Haack, T., González, M.J., Sánchez, Y. and Giralt, E., Lett. Peptide Sci., 4 (1997) 377.
33. Fogel, G.B. and Corne, D.W., Evolutionary Computation in Bioinformatics, Elsevier Science, 2002.
34. Patel, S., Stott, I., Bhakoo, M. and Elliott, P., Patenting Evolved Bactericidal Peptides. In Bentley, P. and Corne, D.W. (Eds.), Creative Evolutionary Systems. Morgan Kaufmann Publishers, 2001.
35. Kamphausen, S., Höltgen, N., Wirsching, F., Morys-Wortmann, C., Riester, D., Goetz, R., Thürk, M. and Schwienhorst, A., J. Comput.-Aid. Mol. Design, 16 (2002) 551.
36. Michaud S.R., Zydallis J.B., Lamont G.B. and Pachter, R., Technical Proceedings of the 2001 International Conference on Computational Nanoscience and Nanotechnology, 2001, pp. 29–32.
37. Goh, G.K.-M. and Foster, J.A., Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2000, Morgan Kaufmann, 2000, pp. 27–33.
38. Yamashita, F., Wanchana, S. and Hashida, M., J. Pharmaceutical Sci., 91 (2002) 2230.
39. Shoichet, B.K., McGovern, S.L., Wei, B. and Irwin, JJ., Current Opinion in Chem. Biol., 6 (2002) 439.
40. Scheider, G., Lee, M., Stahl, M. and Schneider, P., J. Comput.-Aid. Mol. Design, 14 (2000) 487.
41. Koza, J.R., Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, 1992.
42. Baker, J.E., Proceedings of the First International Conference on Genetic Algorithms, Erlbaum, 1985, pp. 101–111.
43. Baker, J.E., Proceedings of the Second International Conference on Genetic Algorithms, Erlbaum, 1987, pp. 14–21.
44. Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs. Springer, 1992.
45. Back, T., Evolutionary Algorithms in Theory and Practice. Oxford University Press, 1997.
46. Schwefel, H.P., Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Birkhaeuser, 1977.
47. Howard, R. and Matheson, J., Readings on the Principles and Applications of Decision Analysis, Vol. III. Howard, R. and Matheson, J. (Eds.), Strategic Decisions Group, 1981, pp. 721–762.
48. Pearl, J. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann, , 1988.
49. Heckerman, D., Geiger, D. and Chickering, D.M., Technical report of Microsoft Research, MSR-TR-94-09, 1995.
50. Pelikan, M., Goldberg, D.E and Cantú-Paz, E., Technical report of IlliGAL, No. 98013, 1998.
51. Vajda, S. and Camacho, C.J., Trend. Biotechnol., 22 (2004) 110.
52. Macke, T.J. and Case, D.A., NAB User's Manual. Departament of Molecular Biology, The Scripps Research Institute, La Jolla, California, 1999.
53. Berman, H.M, Westbrook, J., Feng, Z., Gilliland, G., Bhat, T.N, Weissig, H., Shindyalov, I.N. and Bourne, P.E., Nucleic Acid Res., 28 (2000) 235.
54. Chou, P.Y. and Fasman, G.D., Advanced Enzymol., 47 (1978) 45.
55. Yoshimoto, T., Fischl, M., Orlowski, R. and Walter, R., J. Biol. Chem., 10 (1978) 3708.
56. Goossens, F., De Meester, I., Vanhoof, G. and Scharpé, S., Eur. J. Clin. Chem. Clin. Biochem., 34 (1996) 17.
57. Fülöp, V., Bocskei, Z. and Polgár, L., Cell, 94 (1998) 161.
58. Mentlein, R., FEBS Lett., 234 (1988) 251.
59. Maes, M., Goossens, F., Scharpé, S., Calabrese, J., Desnyder, R. and Meltzer, H.Y., Psychiatry Res., 58 (1995) 217.
60. Maes, M., Lin, A.H., Bonaccorso, S., Goossens, F., Gastel, A.V., Pioli, R., Delmerie, L. and Scharpé, S., J. Affect. Disord., 53 (1999) 27.
61. Vogelstein, B., Lane, D. and Levine, A.J., Nature, 408 (2000) 307.
62. Chene, P., Oncogene, 20 (2001) 2611.
63. Clore, M., Ernst, J., Clubb, R., Omichinski, J.G., Kennedy, W.M.P., Sakaguchi, K., Appella, E. and Gronenborn, A.M., Nat. Struct. Biol., 2 (1995) 321.
64. Salvatella, X., Martinell, M., Gairí, M., Mateu, M.G., Feliz, M., Hamilton, A.D., de Mendoza, J. and Giralt, E., Angewantde Chemie International Edition, 43 (2004) 196.
65. Martinell, M., Disseny, síntesi i estudi de lligands peptídics capaços de reconèixer la superfície de la p53, Ph.D. dissertation at Universitat de Barcelona, 2004.
66. Gellert, M., Ann. Rev. Biochem., 50 (1981) 879.
67. Vizan, J.L., Hernandez-Chico, C., del Castillo, I. and Moreno, F., EMBO J., 10 (1991) 467.
68. Yorgey, P., Davagnino, J. and Kolter, R., Mol. Microbiol., 9 (1993) 897.
69. Luz, J.G., Huang, M., Garcia, K.C., Rudolph, M.G., Apostolopoulos, V., Teyton, L. and Wilson, I.A., J. Exp. Med., 195 (2002) 1175.
70. Falk, K., Rotzschke, O., Stevanovic, S., Jung, G. and Rammensee, H.G., Nature, 351 (2001) 290.
71. Pelikan, M., Goldberg, D.E. and Cantú-Paz, E., Technical report of IlliGAL, No. 2000001, 2000.