Frankenstein Junior: a relational learning approach toward protein engineering

Elisa Cilia[1,2], Andrea Passerini*[1]

[1] Department of Computer Science and Information Engineering, University of Trento, via Sommarive 14, I-38123 (Povo) Trento, Italy

[2] FEM-IASMA Research & Innovation Center, via Edmund Mach 1, I-38010 S. Michele all'Adige (TN), Italy

Email: Elisa Cilia - cilia@disi.unitn.it, elisa.cilia@iasma.it; Andrea Passerini*- passerini@disi.unitn.it;

*Corresponding author

Abstract

Background: Mining relevant features from protein mutation data is fundamental for understanding the characteristics of a protein functional site. The mined features could also be used for engineering novel proteins with useful functions.

Results: We propose a simple relational learning approach for protein engineering. First, we learn a set of relational rules from mutation data, then we use them for generating a set of candidate mutations that are most probable to confer resistance to a certain inhibitor or improved activity on a specific substrate. We tested our approach on a dataset of HIV drug resistance mutations, comparing it to a baseline random generator. Statistically significant improvements are obtained on both categories of nucleoside and non-nucleoside HIV reverse transcriptase inhibitors.

Conclusions: Our promising preliminary results suggest that the proposed approach for learning mutations has a potential in guiding mutant engineering, as well as in predicting virus evolution in order to try and devise appropriate countermeasures. The approach can be generalized quite easily to learning mutants characterized by more complex rules correlating multiple mutations.

Background

The mining of relevant features from protein mutation data has its first aim in understanding the properties of functional sites, for instance, which residues are more likely to have a functional role. The same mined information can be used to engineer mutants of a protein with an improved activity on a certain substrate or resistance to a certain inhibitor.

Rational design is an engineering technique modifying existing proteins by site directed mutagenesis. It assumes the knowledge or intuition about the effects of specific mutations on the protein function. The process typically involves extensive trial-and-error experiments and is also used with the aim of improving the understanding mechanisms of a protein behavior and taking the necessary countermeasures.

In this work we moved the first steps towards the use of a relational learning approach to protein engineering by focusing on learning relevant single mutations (mutations that change the amino acid type in the protein sequence) that can affect the behavior of

a protein. Predicting the effect of single mutations helps reducing the dimension of the search space for rational design.

In the proposed approach an Inductive Logic Programming (ILP) [1] learner is trained to extract general rules describing mutations relevant to a certain behavior (e.g. drug resistance of HIV) and the learned rules are then used to infer novel potentially relevant mutations.

We focused on learning relevant mutations of the HIV reverse transcriptase (RT). The HIV RT is a DNA polymerase enzyme that transcribes RNA into DNA, allowing it to be integrated into the genome of the host cell and replicated along with it, and it is therefore crucial for virus propagation. Viruses typically have a very high mutation rate and a mutation can confer the mutant resistance to one or more drugs, for instance by modifying the inhibitor target site on the protein. In the case of the HIV drug resistance, which we addressed in the present work, building artificial mutants that can resist to the inhibitor could help to early predict the virus evolution and thus design more effective drugs.

Many machine learning methods have been applied in the past to mutation data for predicting single point mutations on protein stability changes [2] and the effect of mutations on the protein function [3] [4] or drug susceptibility [5]. At our knowledge this is the first approach proposal for learning relational features of mutations affecting a protein behavior and use them for generating novel relevant mutations. Furthermore, even if we focus on single point mutations in our experimental evaluation, our approach can be quite straightforwardly extended to multiple point mutations, and we are actively working in this direction. Conversely, the up-mentioned predicting approaches would immediately blow up for the explosion in the number of candidate configurations to evaluate.

Results and Discussion

Dataset

We applied our approach to the same dataset of mutations already used in [6] for extracting relational rules among mutations. The dataset is derived from the Los Alamos National Laboratories (LANL) HIV resistance database[1] and reports mutations of the HIV reverse transcriptase (RT). Richter et al. [6] formulated the learning problem as a mining task and applied a relational association rule miner to derive rules relating different mutations and their resistance properties.

In previous work [7] we used the same dataset for learning a relational model of mutant resistance with the hierarchical kFOIL algorithm, a statistical relational learner [8]. Here we use the dataset for inferring rules that can characterize a single mutation as resistant to a certain class of RT inhibitors. Those drug classes include: a) Nucleoside RT Inhibitors (NRTI); b) NonNucleoside RT Inhibitors (NNRTI); c) NonCompetitive RT inhibitors (NCRTI); d) Pyrophosphate Analogue RT Inhibitors (PARTI). The four classes of inhibitors differ in the targeted sites and rely on quite different mechanisms. NNRTI and NCRTI inhibit the reverse transcriptase by binding to the enzyme active site, therefore directly interfering with the enzyme function. NRTI is instead incorporated into the newly synthesized viral DNA for preventing its elongation. Finally the PARTI targets the pyrophosphate binding site and it is employed, as part of a salvage therapy, on patients in which the HIV infection shows resistance to the other classes of antiretroviral-drugs. The final dataset is composed of 164 mutations labeled as resistant over a set of 581 observed mutations (extracted from 2339 mutants). Among the 164 mutations, 95 are labeled as resistant to NRTI, 56 to NNRTI, 5 to NCRTI and 8 to PARTI.

Learning in first order logic

Our aim is to learn a first-order logic hypothesis for the target concept, i.e. mutation conferring resistance to a certain drug, and use it to infer novel mutations consistent with such hypothesis. We rely on definite clauses which are the basis of the Prolog programming language. A definite clause is an expression of the form `h` $\leftarrow$ `b1 AND ... AND bn`, where `h` and the `bi` are atoms. Atoms are expressions of the form `p(t1, ..., tn)` where `p/n` is a predicate symbol of arity `n` and the `ti` are terms, either constants (denoted by lower case) or variables (denoted by upper case) in our experiments. The atom `h` is also called the head of the clause, typically the target predicate, and `b1 AND ... AND bn` its body. Intuitively, a clause represents that the head `h` will hold whenever the body `b1 AND`

[1] http://www.hiv.lanl.gov/content/sequence/RESDB/

... AND bn holds. For instance, a simple hypothesis like res_against(A,ncrti) ← mutation(A,C) AND close_to_site(C) would indicate that a mutation C in the proximity of a binding site confers to mutant A resistance against a ncrti. Learning in this setting consists of searching for a set of definite clauses $H = \{c_i, ..., c_m\}$ covering all or most positive examples, and none or few negative ones if available. First-order clauses can thus be interpreted as relational features that characterize the target concept. The main advantage of these logic-based approaches with respect to other machine learning techniques is the expressivity and interpretability of the learned models. Models can be readily interpreted by human experts and provide direct explanations for the predictions.

Background knowledge

We built a relational knowledge base for the domain at hand. Table 1 summarizes the predicates we included as a background knowledge. We represented the amino acids of the wild type with their positions in the primary sequence (aa/2) and the specific mutations characterizing them (mut_prop/4). Target predicates were encoded as resistance of the mutation to a certain drug (res_against/2).

Additional background knowledge was included in order to highlight characteristics of residues and relationships between mutations:

color/2 indicates the type of the natural amino acids according to the coloring proposed in [9]. For example the magenta class includes basic amino acids as lysine and arginine while the blue class includes acidic amino acids as aspartic and glutamic acids.

same_type/2 indicates whether two residues belong to the same type, i.e. a change from one residue to the other conserves the type of the amino acid.

same_type_mut/2 indicates that a residue substitution at a certain position does not modify the amino acid type with respect to the wild type. For example mutation d123e conserves the amino acid type while mutation d123a does not (i.e. different_type_mut/2 holds for it).

Other background knowledge facts and rules were added in order to express structural relations along the primary sequence and catalytic propensity of the involved residues:

close_to_site/1 indicates whether a specific position is distant less than 5 positions from a residue belonging to a binding or active site. In our specific case, the background theory incorporates knowledge about a metal binding site and a heterodimerization site.

location/2 indicates in which fragment of the primary sequence the amino acid is located. Locations are numbered from 0 by dividing the sequence into a certain number of fragments.

catalytic_propensity/2 indicates whether an amino acid has a high, medium or low catalytic propensity according to [10].

mutated_residue_cp/3 indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high).

Algorithm overview

The proposed approach is sketched in Figure 1.

The first step is the learning phase, in which an ILP learner is fed with a logical representation of the data $\mathcal{D}$ (the mutations experimentally observed to confer resistance to a certain inhibitor) and of the domain knowledge $\mathcal{B}$ we want to incorporate, and it returns a first-order logical hypothesis H for the concept of mutation conferring resistance to a certain drug.

The hypothesis is derived using the Aleph (A Learning Engine for Proposing Hypotheses) ILP system[2]. Aleph allows also to learn from positive example only. This is the most suitable approach in our case as the positive examples are the mutations experimentally proved to confer resistance to a drug but no safe claim can be made on the other mutations if there is no sufficient evidence due for example to the lack of an exhaustive set of laboratory experiments.

Aleph incrementally builds a hypothesis covering all positive examples guided by a Bayesian evaluation function, described in [11], scoring candidate

[2]http://www.comlab.ox.ac.uk/activities/machinelearning/Aleph/aleph.html

solutions according to an estimate of the Bayes' posterior probability that allows to tradeoff hypothesis size and generality.

In Figure 2 we show an example of learned hypothesis covering a set of examples. The learned hypothesis models the ability of a mutation to confer the resistance to NCRTIs and is composed of three first-order clauses, each one covering different sets of mutations of the wild type as highlighted in colors: blue for the first clause, yellow for the second and red for the third one. Some mutations are covered by more than one clause as shown by the color overlaps.

Our background knowledge incorporates information about the RT metal binding site, which is composed of the aspartic acids D110, D185 and D186 and, about the heterodimerization site composed of W401 and W414 (in bold in Figure 2).

The second step is the generative phase, in which the learned hypothesis is employed to find novel mutations that can confer drug resistance to an RT mutant. A set of candidate mutations can be generated by using the Prolog inference engine starting from the rules in the learned model. The rules are actually constraints on the characteristics that a mutation of the wild type should have in order to confer resistance to a certain inhibitor.

Algorithm 1 details the mutation generation procedure. We here assume, for simplicity, to have a model H for a single drug class. The procedure works by querying the Prolog inference engine for all possible variable assignments that satisfy the hypothesis clauses. In order to generate novel mutations, the `mut_prop/4` predicate is modified here in order to return all legal mutations instead of only those appearing in the training instances. The set of mutations identified by the variable assignments and not present in the training set is ranked according to a scoring function S_M before being returned by the algorithm. In this work we defined S_M as the number of clauses in H that a candidate mutation m satisfies.

Referring to the model in Figure 2 some generated candidate mutations with highest score are: 101P, 102A, 103F, 103I, 104M, 181I, 181V, 183M, 188L, 188F where for example the notation 101P indicates a change of the wild type amino acid, located in position 101, into a proline (P). This list includes also known surveillance mutations [12].

Performance evaluation

We divided the dataset of mutations into a training and a test set (70/30) in a stratified way, which means by preserving, both in the train and test set, the proportion of examples belonging to one of the four drug classes. The resulting training set is composed of a total of 116 mutations while the test set is composed of 48 mutations.

We trained the ILP learner on the training set and we evaluated on the test set the set of mutations generated using the learned model. The evaluation procedure takes the generated mutations and computes its enrichment in test mutations by counting how many generated mutations are actually observed in at least one test example. We compare the recall of the approach with the recall of an algorithm that choose at random a set (of the same cardinality) of possible mutations among all legal ones.

Recall or *sensitivity* or *TP rate* is $\frac{t^+}{t^+ + f^-}$ where t^+, the true positives, is the number of test set mutations and $t^+ + f^-$, true positives plus false negatives, corresponds to the number of generated mutations.

Results averaged on 30 random splits of the dataset are reported in Table 2. On each split we performed 30 runs of our algorithm and of the random generation algorithm in each one of the different learning tasks (NNRTI, NRTI, NCRTI and PARTI). For each task we also reported in column 3 and 4 the mean number of generated mutations over the 30 splits and the number of test set mutations for reference.

We evaluated the statistical significance of the performance differences between the two algorithms by paired Wilcoxon tests on the averaged recall reported on each split. We employed a confidence level α of 0.05.

The improvement of the algorithm with respect to the random generation of mutations is statistically significant on the NRTI and NNRTI tasks, which are the tasks on which we can learn the hypothesis from a largest set of training examples.

Figure 2 shows the trend of the mean recall over all splits when cutting the number of generated mutations (from 1 generated mutation to more than 8000). The advantage of our approach remains quite stable when reducing the set of candidates, producing almost nine times more test mutations than random in the 100 highest scoring ones for NNRTI (Figure 3 shows the trend of the recall curves of the plot in Figure 3(a) for the highest ranked 150 mutations).

We finally learned a model on the whole set of training mutations in order to generate a single set of mutations for further inspection. Below we reported five examples of novel mutations with the highest rank for each one of the tasks:

NNRTI 90I 98I 103I 106P 179I

NRTI 60A 153M 212L 229F 239I

NCRTI 183V 183L 188V 188F 188I

PARTI 84R 86E 88Y 88V 89N

In [13], the authors found a set of novel mutations conferring resistance to efavirenz and nevirapine, which are NNRTIs. Our mutation generation algorithm partially confirmed their findings. Mutation 90I was ranked high (5/5), mutation 101H was generated with a rank of 3/5, mutations 196R and 138Q with rank 1/5, while mutation 28K was not generated at all by our system as a candidate for conferring resistance to NNRTI.

Example of learned hypothesis

An example of learned hypothesis is reported in Figure 5. For instance, according to the model, among the features a mutation should have for conferring resistance to a NNRTI, there is the change of a basic (magenta) residue of the wild type, e.g. lysine or arginine, into a residue with completely different phisico-chemical characteristics (rule 16).

Another example for the resistance to NNRTI is that a non conserved mutation is present in positions between 98 and 106 of the wild type sequence (rule 8).

Conclusions

In this work we proposed a simple relational learning approach toward protein engineering. Starting from HIV reverse transcriptase mutation data we built a relational knowledge base and we mined relevant relational features for modeling mutant resistance by using an ILP learner. Based on the learned relational rules we generate a set of candidate mutations satisfying them.

Albeit preliminary, our results suggest that the proposed approach for learning mutations has a potential in guiding mutant engineering, as well as in predicting virus evolution in order to try and devise appropriate countermeasures. A more detailed background knowledge, possibly including 3D information whenever available, is necessary in order to further focus the set of generated mutations, and possibly post-processing stages involving mutant evaluation by statistical machine learning approaches [2]. In the next future we also plan to generalize the proposed approach to jointly generate sets of related mutations shifting the focus from single point mutations to entire mutants.

Authors contributions

EC built the background knowledge, implemented the algorithm and conducted the experimental evaluation. AP suggested the overall idea and coordinated the study. Both authors contributed in writing the article. Both authors read and approved the final manuscript.

References

1. Muggleton S, De Raedt L: **Inductive Logic Programming: Theory and Methods**. *University Computing* 1994, :629–682.
2. Capriotti E, Fariselli P, Rossi I, Casadio R: **A three-state prediction of single point mutations on protein stability changes.** *BMC bioinformatics* 2008, **9 Suppl 2**:S6.
3. Needham CJ, Bradford JR, Bulpitt AJ, Care Ma, Westhead DR: **Predicting the effect of missense mutations on protein function: analysis with Bayesian networks.** *BMC bioinformatics* 2006, **7**:405.
4. Bromberg Y, Rost B: **SNAP: predict effect of non-synonymous polymorphisms on function.** *Nucleic acids research* 2007, **35**(11):3823–35.
5. Rhee SY, Taylor J, Wadhera G, Ben-Hur A, Brutlag DL, Shafer RW: **Genotypic predictors of human immunodeficiency virus type 1 drug resistance.** *Proceedings of the National Academy of Sciences of the United States of America* 2006, **103**(46):17355–60.
6. Richter L, Augustin R, Kramer S: **Finding Relational Associations in HIV Resistance Mutation Data**. In *Proceedings of Inductive Logic Programming (ILP), Volume 9* 2009.
7. Cilia E, Landwehr N, Passerini A: **Mining Drug Resistance Relational Features with Hierarchical Multitask kFOIL**. In *Proceedings of BioLogical@AI*IA2009* 2009.
8. Landwehr N, Passerini A, Raedt L, Frasconi P: **Fast learning of relational kernels**. *Machine Learning* 2010, **78**(3):305–342.
9. Taylor WR: **The classification of amino acid conservation.** *Journal of Theoretical Biology* 1986, **119**(2):205–218.

10. Bartlett G, Porter C, Borkakoti N, Thornton J: **Analysis of catalytic residues in enzyme active sites**. *Journal of Molecular Biology* 2002, **324**:105–121.

11. Muggleton S: **Learning from Positive Data**. In *Inductive Logic Programming Workshop* 1996:358–376.

12. Bennett DE, Camacho RJ, Otelea D, Kuritzkes DR, Fleury H, Kiuchi M, Heneine W, Kantor R, Jordan MR, Schapiro JM, Vandamme AM, Sandstrom P, Boucher CaB, van de Vijver D, Rhee SY, Liu TF, Pillay D, Shafer RW: **Drug resistance mutations for surveillance of transmitted HIV-1 drug-resistance: 2009 update.** *PloS one* 2009, **4**(3):e4724.

13. Deforche K, Camacho RJ, Grossman Z, Soares Ma, Van Laethem K, Katzenstein Da, Harrigan PR, Kantor R, Shafer R, Vandamme AM: **Bayesian network analyses of resistance pathways against efavirenz and nevirapine.** *AIDS (London, England)* 2008, **22**(16):2107–15.

Algorithms

Algorithm 1 - Mutation generation algorithm.

Algorithm for novel relevant mutations discovery.

Algorithm 1 Mutation generation algorithm.

input: *dataset of training mutations* $\mathcal{D}$*, background knowledge* $\mathcal{B}$*, learned model* H
output: *rank of the most relevant mutations* $\mathcal{R}$
procedure GENERATEMUTATIONS($\mathcal{D}, \mathcal{B}, H$)
    Initialize $\mathcal{D}_{\mathcal{M}} \leftarrow \emptyset$
    $A \leftarrow$ find all assignments a that satisfy at least one clause $c_i \in H$
    for $a \in A$ **do**
        $m \leftarrow$ mutation corresponding to the assignments $a \in A$
        $score \leftarrow S_M(m)$ ▷ number of clauses c_i satisfied by a
        if not $m \in \mathcal{D}$ **then** ▷ discard mutations observed in the training set
            $\mathcal{D}_{\mathcal{M}} \leftarrow \mathcal{D}_{\mathcal{M}} \cup \{(m, score)\}$
        end if
    end for
    $\mathcal{R} \leftarrow$ RANKMUTATIONS($\mathcal{D}_{\mathcal{M}}, \mathcal{B}, H$) ▷ rank relevant mutations
    return $\mathcal{R}$
end procedure

Figures

Figure 1 - Mutation engineering algorithm.

Schema of the mutation engineering algorithm.

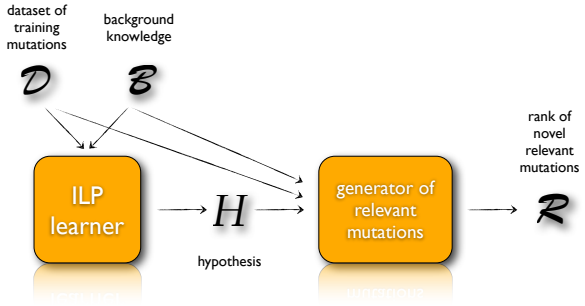
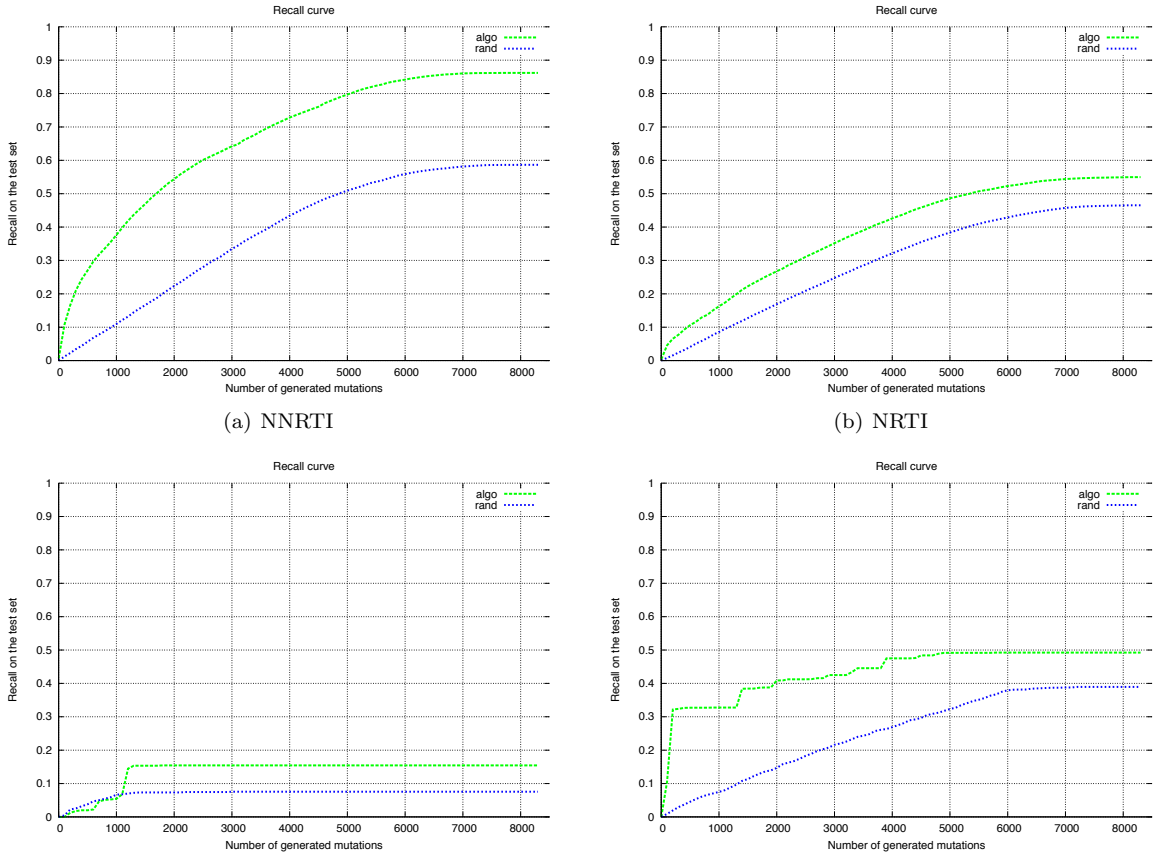


Figure 1: Schema of the mutation engineering algorithm.

Figure 2 - Model for the resistance to NCRTI
An example of learned hypothesis for the NCRTI task with highlighted examples covered by the hypothesis clauses.

Figure 3 - Mean recall by varying the number of generated mutations
Mean recall by varying the number of generated mutations.



(a) NNRTI

(b) NRTI

(c) NCRTI

(d) PARTI

Figure 2: Mean recall by varying the number of generated mutations.

Figure 4 - Recall curves of the NNRTI task for the highest ranked mutations.
Detail of the mean recall curves of the NNRTI task for a number of generated mutations below 150.

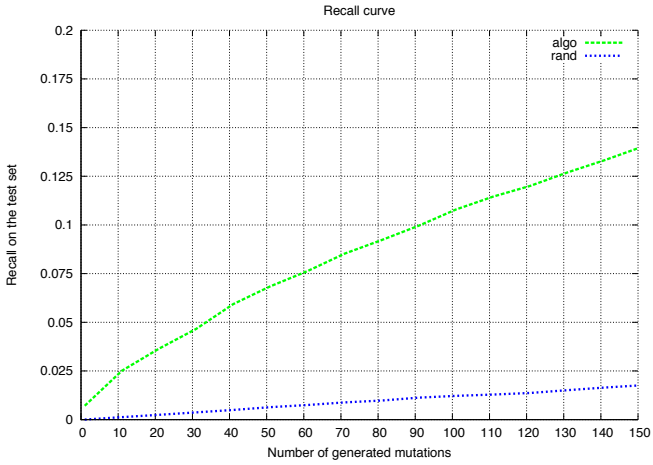


Figure 3: Detail of the mean recall curves of the NNRTI task for a number of generated mutations below 150.

Figure 5 - Example of learned model

```
1-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND color(red,D)
2-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND color(red,D) AND color(red,B)
3-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND location(7.0,C) AND mutated_residue_cp(C,medium,medium)
4-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND location(7.0,C)
5-res_against(A,nrti) ⟵ same_type_mut(A,B)
6-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND mutated_residue_cp(C,medium,high)
7-res_against(A,nrti) ⟵ mut_prop(A,B,C,D) AND mutated_residue_cp(C,medium,small)
8-res_against(A,nnrti) ⟵ different_type_mut(A,B) AND location(11.0,B)
9-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND mutated_residue_cp(C,small,small)
10-res_against(A,nnrti) ⟵ same_type_mut(A,B)
11-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND aminoacid(B,v)
12-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND location(15.0,C)
13-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND aminoacid(D,i)
14-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND location(11.0,C)
15-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND color(red,D)
16-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND color(magenta,B) AND different_type_mut(A,C)
17-res_against(A,nnrti) ⟵ mut_prop(A,B,C,D) AND location(21.0,C)
18-res_against(A,ncrti) ⟵ mut_prop(A,B,C,D) AND location(11.0,C)
19-res_against(A,ncrti) ⟵ mut_prop(A,B,C,D) AND mutated_residue_cp(C,high,small)
20-res_against(A,ncrti) ⟵ mut_prop(A,B,C,D) AND color(green,B) AND close_to_site(C)
21-res_against(A,parti) ⟵ mut_prop(A,B,C,D) AND location(9.0,C)
```

Tables

Table 1 - Background knowledge predicates

Summary of the background knowledge facts and rules.

Background Knowledge Predicates	
`aa(Pos,AA)`	indicates a residue in the wild type sequence
`mut_prop(MutationID,AA,Pos,AA1)`	indicates a mutation: mutation identifier, position and amino acids involved, before and after the substitution
`res_against(MutationID,Drug)`	indicates whether a mutation is resistant to a certain drug
`color(Color,AA)`	indicates the type of a natural amino acid
`same_type(R1,R2)`	indicates whether two residues are of the same type
`same_type_mut(MutationID, Pos)`	indicates a mutation to a residue from the same type
`different_type_mut(MutationID, Pos)`	indicates a mutation changing the type of residue
`close_to_site(Pos)`	indicates whether a specific position is close to a binding or active site if any
`location(L,Pos)`	indicates in which fragment of the primary sequence the amino acid is located
`catalytic_propensity(AA,CP)`	indicates whether an amino acid has a high, medium or low catalytic propensity
`mutated_residue_cp(Pos, CPold, CPnew)`	indicates how, in a mutated position, the catalytic propensity has changed (e.g. from low to high)

Table 2 - Results

Statistical comparisons of the performance of the proposed algorithm with an algorithm generating mutations at random. The average recall has been computed for each one of the learning tasks over the 30 splits by averaging recall over 30 repeated runs of the two algorithms. Results of a paired Wilcoxon test ($\alpha = 0.05$) on the statistical significance of the performance differences are also reported. A black bullet indicates a statistical significant improvement of our algorithm over a random generator.

	Mean recall % on 30 splits			
	Algorithm	**Random Generator**	*Mean n. generated mutations*	*n. test mutations*
NNRTI	86 •	58	5201	17
NRTI	55 •	46	5548	28
NCRTI	16	8	1042	1
PARTI	49	39	3425	2