# Stackelberg Games for Vaccine Design

Swetasudha Panda and Yevgeniy Vorobeychik
Electrical Engineering and Computer Science
Vanderbilt University
Nashville, TN
{swetasudha.panda, yevgeniy.vorobeychik}@vanderbilt.edu

## ABSTRACT

Stackelberg game models have recently seen considerable practical and academic success in security applications, with defender as the leader, and attacker the follower. The key conceptual insight of Stackelberg security games is that defense needs to be proactive, optimally accounting for attacker's response to a defensive posture. We propose that this insight has relevance in another important application domain: vaccination. Vaccination therapies are important tools in the battle against infectious diseases such as HIV and influenza. However, many viruses, including HIV, can rapidly escape the therapeutic effect through a sequence of mutations. We propose to design vaccines, or, equivalently, antibody sequences, that make such evasion difficult. Formally, we model the interaction between a vaccine and a virus as a Stackelberg game in which the vaccine designer chooses an antibody, and the virus chooses a minimal sequence of mutations to escape it.

Our crucial observation is that we can leverage protein modeling software, Rosetta, as an oracle to compute binding score for an input virus-antibody pair. This observation enables us to develop a fully automated bi-level stochastic optimization algorithm for optimal antibody "commitment" strategy. A key technical challenge is that score calculation for each possible antibody-virus pair is intractable. We therefore propose a novel simulation-based bi-level optimization algorithm to address this, which consists of three elements: first, application of local search, using a native antibody sequence as leverage, second, machine learning to predict binding for antibody-virus pairs, and third, a Poisson regression to predict escape costs as a function of antibody sequence assignment. We demonstrate the effectiveness of the proposed methods, and exhibit an antibody with a far higher escape cost (7) than the native (1).

## Categories and Subject Descriptors

G.1.6 [**Numerical Analysis**]: Optimization-Stochastic programming; I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems-Medicine and science; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search-Heuristic methods

**Appears in:** *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

## Keywords

Stackelberg games; Heuristic search and optimization; Machine learning

## 1. INTRODUCTION

Infectious diseases that defy definitive treatment are a major public health challenge. Millions of people worldwide are infected with HIV, many of them expected to die from AIDS [21]. Antibiotic resistant bacterial strains have raised much alarm, and show no signs of abating [3]. Annual influenza cycles, while not usually deadly, cause substantial productivity loss and much temporary pain. Finally, a recent Ebola outbreak in Africa has killed thousands so far, and may kill many more before it runs out of steam [4].

An important problem faced by drug and vaccine designers is evolution: diseases, particularly those that can mutate rapidly, such as HIV and influenza, invariably escape specific vaccine/drug treatment.

Vaccination, which is the focus of our work, stimulates the immune system to produce antibodies that bind to the vaccine substance. The goal is that the produced antibodies are subsequently capable of binding relevant strains of the live pathogen, rapidly militating immune response against disease. Consequently, a core question in vaccination research is how to *design* or *discover* an antibody that is effective against a particular pathogen. To simplify terminology, we henceforth focus our discussion on viruses, although our methods are general. Antibody effectiveness has two forms: the first is that it actually binds the virus it is meant to bind; the second is that the virus does not easily mutate and escape such binding. This latter design criterion typically takes the form of designing/discovering *broadly binding antibodies*, or antibodies that bind to many different known strains of the same virus, for example, to many different influenza or HIV strains found "in the wild" [11].

We formulate antibody design as a Stackelberg game between the vaccine designer (drug designer, etc), who stimulates an antibody with particular binding characteristics (this is the binding site in the antibody sequence), and the virus subsequently responds to the antibody by attempting to evade it (evade binding to it, that is) through a series of local mutations. So, the "designer" chooses an antibody, and the virus responds through a shortest sequence of mutations leading to escape. In nature, evasion models natural selection where fitness criterion principally includes not binding to the antibody (since otherwise the virus is killed by the immune system).

The designer-virus game poses two challenges: 1) enormous search space for both the designer and the virus ($\geq 10^{50}$ in each case), and 2) determination whether an arbitrary antibody-virus pair bind. To tackle the former challenge, we propose, and compare the performance of, several stochastic local search heuristics, using the native antibody as a "springboard". Even for computing virus escape alone, this approach scales poorly. The major bottleneck is the second challenge: binding evaluation. For this purpose we make use of Rosetta, a premier computational protein modeling tool [8]. Rosetta, however, can be extremely time consuming even for a single evaluation (which could take nearly an hour, as it makes use of its own sophisticated amalgam of local search techniques to simulate a binding complex). To significantly speed up the search, we use classification learning to predict whether or not an antibody-virus pair bind, limiting Rosetta evaluations only to cases in which the classifier predicts that they do not. While this makes the virus escape search practical, the bi-level nature of the problem means that antibody design is still quite time consuming. To address this, we make use of Poisson regression to predict virus escape cost. Making use of the resulting predictions now makes antibody design viable, with "inner loop" (virus escape) evaluations restricted to a small set of candidate antibodies predicted to be difficult to escape.

In summary, we make the following contributions:

1. A novel Stackelberg game model of antibody design and virus escape interaction,
2. stochastic local search techniques to determine optimal virus escape, with classifier-in-the-loop used to speed up the evaluations, and
3. stochastic local search techniques for optimal antibody design, making use of Poisson regression to predict minimal virus escape time.

Our methods ultimately exhibit antibodies that are far more robust to mutation than the native antibody.

## 2. RELATED WORK

Conceptually, our work follows on the steps of Stackelberg game modeling efforts in security [19, 13, 14]. However, the specific models developed for security are completely inadequate for our domain: there are no meaningful targets and no defense resources in vaccine design; rather, the specific details of the antibody-virus interaction build on biochemistry and computational protein modeling.

Our work bears superficial similarity to game theoretic models of vaccination decisions [2, 5, 17]. However, this line of work aspires to model human decisions about being vaccinated, relative to socially optimal choices, whereas our model involves molecular-level interactions between immunity and pathogen; the two models therefore have virtually nothing in common. A somewhat more similar model by Huang et al. [12] considers, at a very high level, the symbiotic-pathogenic spectrum of microbial-host relationship through the lens of several simple game models (pure cooperation, zero-sum, and prisoners' dilemma). Another model [1], uses a high-level public goods model and evolutionary game theory to capture population-level evolution of cancer cells, with implications for resistance to therapies targeting growth factor production. However, our work, to our knowledge, is the first game theoretic model of molecular-level interaction between infectious disease treatment and disease.

Previous work most similar to ours was
in combinatorial drug design, which involved the design of rule-based expert systems to recommend individualized treatment strategies for patients, based on individual mutation history [16]. Rules are applied to infer mutation pattern and this rule-directed search finds possible mutations. The corresponding optimal drug combination is found by solving a triply nested combinatorial optimization problem [15]. The primary drawback of this approach (common to all approaches using broad immunity as a criterion) is that any unseen mutations are assumed non-existent. Moreover, the search space is limited to the possible combination of a few drugs, and is thus very small (can be searched exhaustively). In addition, domain expertise is required to formulate the rules. Decision is made only based on frequently observed mutations. Another set of research focuses on understanding the dynamics of appearance of mutations using dynamic probabilistic graphical models to predict viral evolution [9], while [20] use descriptive mining methods to understand correlations and associations in mutations. In these cases, mutation process is studied in the context of a specific environment (e.g., drugs) using available data. Our investigation requires a model of the mutation process for arbitrary antibodies (or drugs), and therefore cannot make direct use of such approaches.

## 3. ANTIBODY DESIGN AS A STACKELBERG GAME

When an antibody is present in the system, it effectively reduces the fitness of all virus mutations that bind to it. This exerts selective pressure on virus mutants, ultimately leading to survival of those which escape binding. The *native* viral strains (also called *wild type*, in that they are typically found "in the wild") have, by definition, an evolutionary advantage *in the absence of the antibody* (vaccine), and can be presumed to initially dominate. Consequently, mutations that exhibit greater differences from the native (wild type) are increasingly unlikely, both because three or more point mutations are unlikely, and because general selective pressures on the virus [18]. Thus, the virus in the presence of an antibody that binds the native faces two opposing pressures: one which pushes it to escape the antibody, and the other to retain most of the native type protein structure.

Let $v^0$ denote the native virus, which we treat simply as a sequence (vector) of amino acids, and $v$ and $a$ arbitrary virus and antibody sequences, respectively. Let $O(a, v)$ represent binding energy for the antibody-virus pair $(a, v)$, which is computed by Rosetta. We stylize the "dilemma" faced by the virus as the following constrained optimization problem:

$$\min_{v \in V} \|v^0 - v\|_0 \tag{1a}$$

$$\text{s.t.} : O(a, v) \geq \theta, \tag{1b}$$

where $V$ is the space of virus sequences under consideration, and $\theta$ is a threshold on binding energy which designates escape (that is, once binding energy is high enough, the proteins will no longer bind[1]); this threshold is typically domain-dependent. The $l_0$ norm simply computes the number of sequence positions in $v$ that are different from $v^0$. While in principle we could consider the space of all possible virus sequences in this subproblem, since virus structure and, consequently, its binding properties can be affected by a change in any residue (amino acid) in its sequence. However, first-order effect in regard to its antibody binding properties is determined by the sequence that is a part of the native virus binding site. Therefore, we only consider the problem of virus escape in terms of binding site mutations.

The optimization problem (1) can be viewed as a *best response* of the virus to a fixed antibody $a$. Now we consider the problem of designing an antibody, $a$, that is robust to virus escape. The target, virus escape, is now precisely defined by the virus optimization problem (1). Let $v(a)$ be the solution to this problem—naturally, a function of the antibody choice $a$. The designer's decision problem is then

$$\max_{a \in A} \|v^0 - v(a)\|_0, \tag{2}$$

where $A$ is the antibody design space, which we restrict to the native binding site for the same reasons as for the virus. Alternatively, we can write this is a bi-level optimization problem composing (2) with (1):

$$\max_{a \in A} \min_{v \in V} \|v^0 - v\|_0 \tag{3a}$$

$$\text{s.t.}: O(a, v) \geq \theta, \tag{3b}$$

Note that the antibody-virus interaction in our model is a Stackelberg game in which the designer (antibody) is the leader, and the virus is the follower, who chooses an alternative virus sequence in response to the antibody chosen by the designer. Moreover, this game is zero-sum: the designer wishes to maximize the number of escape mutations, a quantity which is minimized by the virus. This interaction bares more than surface similarity to Stackelberg security games [19, 13, 14]; the nature of the model, of course, is entirely distinct. Game theoretically, our focus is on commitment to *pure strategies* (i.e., a fixed antibody sequence), and the solution is therefore not necessarily equivalent to a Nash equilibrium of the corresponding simultaneous move game, unlike games in which commitment to a mixed strategy is possible [14].

Returning to the bi-level optimization program that is the core of our antibody design problem, we face two primary challenges: 1) enormous search space for both the designer and the virus, and 2) determination whether an arbitrary antibody-virus pair bind. In the case of the former, even if we restrict the search to the binding sites, the search space for the antibody is $20^{52}$ and it is $20^{45}$ for the virus, since there are 20 amino acids and the binding sites include 52 and 45 residues (sequence "slots"), respectively. Before we begin with the associated algorithmic questions, we reduce the search space significantly by abstracting amino acids into 7 groups that share common chemical properties, with each group represented by a single prototype amino acid. We label these groups with letters $\{C, P, A, W, R, D, N\}$. In the context of the second challenge, we note that even a single evaluation of binding energy for an arbitrary antibody-virus pair using Rosetta can take up to 40 minutes. However, local search, which is one of our core techniques below, can help with this. In particular, if we start with a known antibody-virus binding structure and keep the antibody fixed, we can evaluate the effect of single-point mutations in the virus an order of magnitude faster (i.e., in several minutes). Several minutes is still extremely slow if we consider the search space size, so clearly it is not in itself sufficient, but is a considerable help when coupled with our search methods described below. Setting the challenges aside for the moment, at the high level the problem can be solved as shown in Algorithm 1, where $a^0$ and $v^0$ are the native antibody-virus pair. Algorithm 1 takes as a black box our ability to compute virus escape (which in turn relies on Rosetta as a black box to evaluate binding strength), and is in the form of a very general stochastic local search algorithm [10], which proceeds through a sequence of iterations, choosing and evaluating candidate antibodies $a$ in the process, returning the most effective antibody found at the end.

**Algorithm 1** High-level algorithm for antibody design

```
function ABDesign($a^0, v^0$)
    $s$ = initializeState($a^0, v^0$)
    for $K$ iterations do
        $a$ = chooseNext($s$)
        $e$ = findEscape($a, v^0$)    // $e$ = escape time
        $a^*$ = updateOpt($a, e$)
    return $a^*$
```

## 4. ROSETTA PROTOCOL

An important component of our simulation-based optimization procedure is the evaluation of binding energy for a given antibody-virus pair using Rosetta. We now describe the specific protocol used to this end, developed with the aid of a Rosetta co-creator, striving to minimize the amount of time spent evaluating binding energy.

The native virus-antibody complex PDB[2] is obtained from the protein database and is first cleaned. The *fast relax procedure*[3] is performed on this complex, which works by iteratively making side chain repack and energy minimization steps. The structure can change up to 2-3 Å from the starting conformation during this process. We output 10 structures and choose the one with minimum ddg[4] as the starting relaxed complex. This process requires about 40 minutes per structure output. Ddg of this chosen relaxed complex is the binding score between the native virus and native antibody.

[1]This idea may seem counterintuitive at first, but it is a reflection of the well-known tendency of chemical compounds towards low-energy states.

[2]The Protein Data Bank (PDB) format provides a standard representation for macromolecular structure data derived from X-ray diffraction and NMR studies. The initial antibody-virus complex is obtained in this format and all 3D structures are output in this format after the relax/repack steps described in the protocol.

[3]See `https://www.rosettacommons.org/manuals/archive/rosetta3.4_user_guide/d6/d41/relax_commands.html` for details.

[4]ddg is the energy of the antibody-virus complex less the total energy of the two in isolation. Thus, when ddg is negative it implies that the complex has a lower energy, i.e., is more stable, than individual proteins.

To obtain 3D structures corresponding to single point mutations, we make an appropriate amino acid change in the virus/antibody part of the sequence. This is followed by 1 repack and 1 energy minimization step (as opposed to many cycles of these two steps until some limit is reached required by fast relax), for faster results. This takes about 6 minutes per structure output. Each such procedure is made to output three 3D structures (about 20 minutes total time) corresponding to the mutated sequence. Ddg score of each of the structures is evaluated and the minimum ddg score is recorded as the score corresponding to that particular mutation. For all such quick repack and energy minimization steps, the starting PDB is already relaxed, so there is only a small difference compared to running the much slower fast relax protocol each time.

## 5. COMPUTING MINIMAL VIRUS ESCAPE

Given that computing virus escape is a core subproblem—the "inner loop" of the antibody design process—we begin our endeavor with this subproblem.

### 5.1 Greedy Local Search

Our baseline approach for computing an escape sequence for the virus, given an antibody $a$, is a greedy local search algorithm initialized with the native virus $v^0$ (Algorithm 2). Before even undertaking the search, we check that $v^0$ binds to $a$; if it does not, we can immediately return 0 (that is, there are 0 mutations needed to escape). At the high level, the algorithm proceeds as follows. Starting with $v^0$, the binding score is evaluated for all the neighbors of $v$. The single-point mutation causing the largest increase in binding energy score from the native is chosen at each iteration until this score exceeds the threshold $\theta$. The escape cost is computed simply as the number of greedy iterations, $e$.

**Algorithm 2** Greedy local search for a virus escape sequence minimizing $\|v^0 - v\|_0$.

```
function VirusEscapeGreedy($(a, v^0)$)
    $v \leftarrow v^0$
    $e \leftarrow 0$
    while $O(a, v^0) < \theta$ do
        $v \leftarrow \arg\max_{w \in V : \|v-w\|_0 = 1} \mathcal{O}(a, w)$
        $e \leftarrow e + 1$
    return $e$
```

As mentioned earlier, greedy local search has an important feature that the binding score in each iteration can be computed much faster by Rosetta, given the structure from previous iteration, than if it were computed for an arbitrary antibody-virus pair. An example run of greedy search is shown in Figure 1 for $\theta = 0$, where escape takes 5 mutations.

Local search has two important disadvantages. First, it is quite possible that by considering combinations of mutations one can achieve much faster escape time. An arguably more severe issue is that it still requires extremely slow evaluations by running Rosetta in each iteration. Next, we tackle the latter problem by using classification learning as a means to avoid costly evaluations.

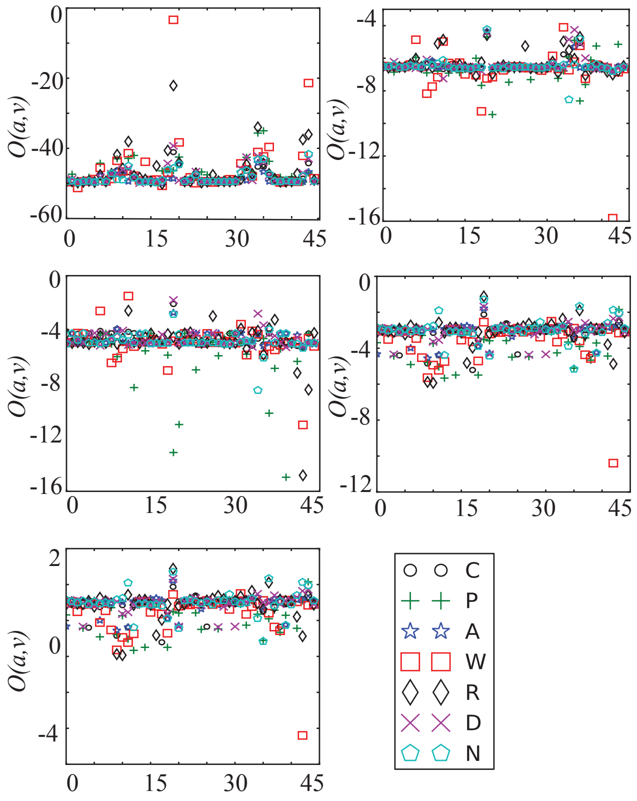

**Figure 1: Example greedy search to compute escape cost. Horizontal axes correspond to point mutations in each virus sequence position, relative to the sequence from previous iteration, and vertical axis is the corresponding binding score. C,P,A,W,R,D,N correspond to the 7 amino acid classes that are candidate mutations.**

### 5.2 Speeding Up Search through Learning

Figure 1 reveals an interesting piece of structure about the problem: most candidate mutations in any iteration make little difference in binding score, but there are typically a few that make a rather significant difference. Conceptually, this is an opportunity: if we could restrict our evaluations only to those that are likely to matter, we can save much time in the execution of the greedy search.

To operationalize this observation, we train a classifier that predicts for a given $(a, v)$ pair whether the virus sequence $v$ will cause a significant change in binding score relative to other single-point mutations from its neighbor (since the said neighbor is left unspecified, we are effectively assuming that significant deviation from baseline score is primarily a property of the evaluated virus sequence, rather than the sequence for which we are considering single-point mutations). To generate training data for this classifier, we collect a set of actual greedy search runs for alternative antibodies. For each $(a, v)$ pair, the feature vector consists of binary indicators whether a particular position is different from the native $(a^0, v^0)$ sequences, as well as a collection of 15 amino acid features defined with the help of domain experts for each position in the sequence pair.

For a given feature vector (i.e., a given $(a, v)$ pair), we assign a label $+1$ if $O(a, v) > M(a, \bar{v}) + 0.9(g(a, \bar{v}) - M(a, \bar{v}))$, and $-1$ otherwise, where $\bar{v}$ is the virus sequence for which $v$ is a single-point mutation, $M(a, \bar{v})$ is the median binding score of all single-point mutations from $\bar{v}$, and $g(a, \bar{v})$ is the highest score among these. We denote the resulting classifier by $\Omega(a, v)$.

In addition, we train using the same data and same features a classifier $\Psi(a, v)$ which predicts whether $a$ and $v$ bind (labeled as $+1$) or not (labeled as $-1$).

In each case, we train a linear SVM classifier with $l_2$ loss and $l_1$ penalty, ensuring sparsity to cope with our rather large feature space. Also, since in both cases the two classes are highly unbalanced (very few mutations cause large increase in the score and we stop searching as soon as there is escape, so most pairs bind), we assign class weights for the two classes that are inversely proportional to their frequencies in the training dataset.

Armed with the two classifiers just constructed, we can now significantly speed up the greedy search. The new algorithm (Algorithm 3) works as follows. In each iteration of the virus escape search and for each possible neighbor $w$ (i.e., single-point mutation) of the current virus iterate $v$, we first check whether $w$ will effect a significant difference from the baseline score for $v$ using the classifier $\Omega(a, w)$. If so, we also check using $\Psi(a, w)$ whether $w$ will still bind to $a$; if we expect that it will not, we verify this prediction by actually evaluating the binding using Rosetta. If it is confirmed, we can now stop the search. Otherwise, $w$ is added to the consideration set of next virus iterates. Finally, we only evaluate those possible single-point mutations from $v$ which we expect to make a significant difference, and which are not predicted to have already escaped (if they are, but were verified to bind, we can simply reuse the corresponding binding score here, so there is no need to evaluate this mutation again).

**Algorithm 3** Classifier-guided greedy search.

```
function ClassifierGuidedSearch($(a, v^0)$)
    $v \leftarrow v^0$
    $e \leftarrow 0$
    while $O(a, v^0) < \theta$ do
        $B \leftarrow \emptyset$
        $e \leftarrow e + 1$
        for $w : \|v - w\|_0 = 1$ do
            if $\Omega(a, w) = +1$ then
                if $\Psi(a, w) = -1$ then
                    if $O(a, w) \geq \theta$ then
                        return $e$
                    else
                        $B \leftarrow B \cup w$
        $v \leftarrow \arg\max_{w \in B} O(a, w)$
    return $e$
```

## 6. ANTIBODY DESIGN

Having considered the problem of computing virus escape for an arbitrary antibody $a$, we now turn to the "outer loop" of the bi-level optimization problem: antibody design. We begin by considering two alternative local search heuristics, taking the evaluation function (virus escape) as given. We then proceed to shortcut virus escape evaluation altogether through another application of machine learning.

### 6.1 Stochastic Local Search for Antibody Design

**Random with a Native Antibody Bias (BiasedRandom):** Our simplest algorithm is a random search which is biased towards the native antibody sequence $a^0$ (and restricted to changes in its binding site alone, as all other methods), so as to take advantage of the structure in the native antibody $a^0$. In particular, we first choose the number of mutations $n$ to $a^0$ uniformly at random in the interval $[1, 52]$ (that is, randomly changing between 1 and all residues in the binding site of the native antibody). Then we choose a random subset of $n$ residues, $R$, in the $a^0$ binding site. Finally, independently for each residue (slot) $r \in R$, we pick an amino acid group distinct from $a^0$ uniformly at random from all the 7 groups we consider. This yields a candidate antibody $a \in A$. We proceed through this search by drawing $I$ such candidate antibodies $\{a_i\}_{i=1,\dots,I}$. Here, we leverage the classifier $\Psi(a, v)$ to predict whether the $(a, v)$ pair bind. In particular, if $a_i$ drawn according to the procedure above is predicted not to bind to the native virus $v^0$, it is simply discarded, and another is drawn in its place, until one is found which binds to the native virus. Each $a_i$ is evaluated by calling the findEscape$(a_i, v^0)$ evaluation function, which executes Algorithm 3.

**Simulated annealing:** A relatively widely used stochastic local search method is *simulated annealing* [10]. Our variation of simulated annealing (Algorithm 4) uses as a starting point a random antibody that is sampled in exactly the same biased way as *BiasedRandom* above. In addition, it leverages the classifier predicting binding described above to check that an antibody generated in a given step binds to the native virus $v^0$, throwing away any instance that does not.

**Algorithm 4** Simulated Annealing search

```
function ABSearchSA($(a^0, v^0, \alpha, T_0)$)
    do
        $a \leftarrow$ BiasedRandom()
    while $\Psi(a, v^0) = -1$
    $e =$findEscape$(a, v^0)$
    $a^* \leftarrow a$
    $u^* \leftarrow e$
    $T \leftarrow T_0$
    for $i$ in 1 to $I$ do
        $T \leftarrow \alpha T$
        do
            $a' \leftarrow$ random neighbor of $a$
        while $\Psi(a', v^0) = -1$
        $\Delta E \leftarrow$findEscape$(a', v^0)$ $-e$
        if $\Delta E > 0$ then
            $a \leftarrow a'$
            $e \leftarrow$ findEscape$(a', v^0)$
        else
            $a \leftarrow a'$ w.p. $\exp(\Delta E / T)$
            $e \leftarrow$ findEscape$(a', v^0)$
```

## 6.2 Speeding Up Antibody Search through Learning

Clearly, the main bottleneck of the antibody design search is evaluation. While we previously described a collection of strategies for speeding up evaluation, ultimately they are all relatively slow, each requiring multiple calls into Rosetta even in the best case. A natural question is whether we can shortcut this lengthy process altogether by predicting escape time. We implement this idea by using Poisson regression as stochastic prediction of escape times for a given antibody $a$. The advantage of using Poisson regression is that it properly captures the stochasticity of our escape evaluations, an important source of which is stochasticity in Rosetta evaluations. In Poisson regression, the escape time $Z$ is distributed as $Pr(Z = z) = \frac{e^{-\mu}\mu^z}{z!}$, where $\log(\mu) = \beta x$, with $\beta$ the parameter vector and $x$ the vector of features. We used the same set of features as for the classification tasks above.

After the Poisson regression model is learned, it can be used in place of findEscape$(a, v^0)$ in all of the design algorithms, with actual evaluations only necessary to check the final solution.

We wish to make an important final point about overall antibody design implementation. All of the learning methods described need training data, the collection of which must take place during the design process itself. Therefore, the overall algorithm would work as follows. For the first subset of iterations of antibody design, the baseline greedy approach must be used to collect sufficient training data to train the classifiers $\Omega(\cdot)$ and $\Psi(\cdot)$. In the next subset of iterations, the evaluations use the classifier-based methods, as additional training data is collected to predict escape times. Finally, we can proceed with many more iterations of antibody design by only using the predicted escape times. While this is the ideal use of the proposed approach, our evaluation below considers the different proposed pieces in isolation to enable sound practical recommendations.

## 7. EVALUATION

To evaluate our approach we used a native antibody-virus interaction for HIV.

The native structure is the co-crystal structure of the antibody VRC01 complexed with the HIV envelope protein GP120.

This structure has 3 chains, the virus chain G and the heavy and light chains in the antibody H and L. The binding site on the virus is chain G with 45 residues, while the binding site on the antibody includes chains H and L with a total of 52 residues.

The binding score for the native pair is $O(a^0, v^0) = -49.5$. The visual representation of the native binding structure is shown in Figure 2 (left).

### 7.1 Computing Virus Escape

We begin the evaluation with the subproblem of computing virus escape. To evaluate the effectiveness of using the two classifiers in the search process, we consider 346 antibodies drawn according to the Biased Random distribution described above (to mirror the distribution with which they are drawn algorithmically). For evaluation, we consider two settings: a) using 75% for training, and b) using 50% for training, with the rest used for evaluation. In our running time comparison (so that the comparison is meaningful), we

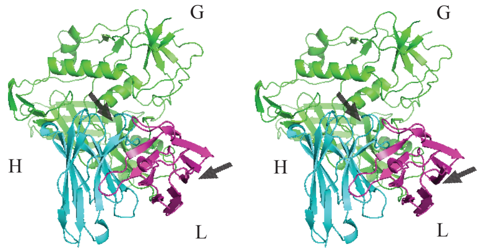

**Figure 2: The native antibody, H and L, with the native virus, G (left) and antibody with escape cost=7 (right). The arrows point at some significant differences.**

use the *combined* running time expended both in collecting the training data and the evaluation. We used the LIBLINEAR SVM implementation [6], using $l_2$ loss and $l_1$ regularization. The ratio of $+1$ to $-1$ instances in the training data is ∼0.005 for both classifiers. The average accuracy for the classifier $\Omega$ which predicts which neighbors will cause a significant change in the baseline score is 90.3% when 75% of the data is used for training and 90.7% when 50% of the data is used for training. The corresponding false negative rates are 6% and 10.2% respectively. For the classifier $\Psi$, the respective accuracies/false negative rates are 90.5%/17.6% and 90.4%/15.3%.

All these results are based on five-fold cross-validation.

The results of the comparison between the baseline and classifier-based greedy approaches for computing virus escape are shown in Figure 3. As expected, using the classifiers in the greedy loop dramatically reduces the number of Rosetta evaluations. The main question is whether it preserves the quality of the resulting solutions. The results in Figure 3 (bottom) show a scatterplot of the escape time difference ($\Delta e$) compared to baseline greedy (verified using Rosetta) for the collection of antibodies tested. Zero, of course, means that they are the same; above zero means that the classifier-based approach finds mutations with smaller escape time than greedy—that is, it actually yields a *better solution*, whereas below zero results imply that the classifier-based approach results in a worse solution than the baseline. It is clear from the figures that quite often the classifier-based approach is actually better, in part because of the randomness that the classifier inaccuracy introduces into the process (as a result of this, it is no longer strictly hill climbing). The average differences, which are $-0.02, 0.07, 0.11$, and 0.15 for (a), (b), (c), and (d) respectively, suggest that we lose very little by switching to the classifier-based search in terms of expected solution quality.

### 7.2 Antibody Design

An important contribution towards practical antibody design was the proposal of using Poisson regression in place of the full virus escape subroutine. The effectiveness of this approach for optimization purposes hinges on our ability to distinguish among antibodies in terms of escape cost, far more so than actual accuracy. Correlation is a natural measure of this. We train the Poisson regression model on the escape cost for the same 346 antibodies considered above. We use the GLMNET package in R [7] to fit Poisson regression parameters, using $l_1$ regularization. We find that the av-

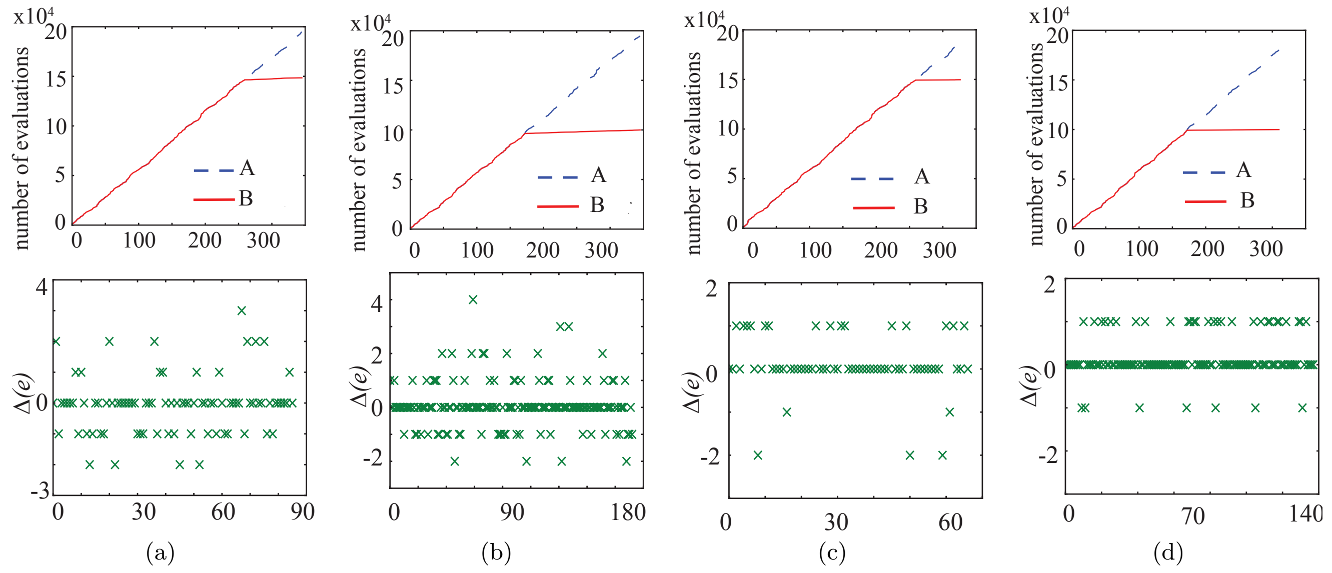

**Figure 3: Comparison between baseline (A) and classifier-based greedy (B) algorithms for computing virus escape in terms of the number of evaluations (top) and computed escape time (bottom). (a) $\theta = 0$, 75% of data for training; (b) $\theta = 0$, 50% of data for training; (c) $\theta = -15$, 75% of data for training; (d) $\theta = -15$, 50% of data for training. Horizontal axes denote antibodies.**

erage correlation between predicted and actual (computed) escape times is 0.66 (based on 10-fold cross-validation), suggesting that the idea is potentially quite viable. Next, we actually utilize the predicted escape costs in the local search algorithms proposed for antibody design: biased random (or simply "random" in the experiments) and simulated annealing. The comparison in terms of *predicted* escape time, as a function of the number of iterations, is shown in Figure 4 . The random biased approach appears clearly better than simulated annealing, perhaps somewhat surprisingly. The likely reason is that our filter that removes any candidates that do not already bind to $v^0$, combined with the bias introduced in search, already provide a good balance between global search and local structure. Next, we evaluated the quality of the final candidate antibody generated by each search after 400 iterations, averaged over 80 independent search sequences using actual greedy local search for virus escape. The results, shown in Figure 5 demonstrate both that the ordering predicted by the Poisson regression is consistent with the evaluation result: random, again, is significantly better than simulated annealing (p-value$< 0.001$).

Finally, we report the upshot: the actual set of antibodies we generated as a part of our search process, ranked in terms of evaluated escape cost (Figure 6). It is noteworthy that we found many antibodies which are much more robust to escape than the native when $\theta = 0$.

## 7.3 The Best Antibody

The best antibody discovered in our experiments has escape cost of 7 (compared to only 1 mutation needed to escape the native VRC01 antibody!), and the resulting antibody complexed with the native virus is shown in Figure 2 (right). The designed antibody has 39 amino acid changes from the native. Structurally, this antibody has two portions of the mid-H chain that are somewhat wider apart, which likely leads to a better grip on the virus chain G. Similarly

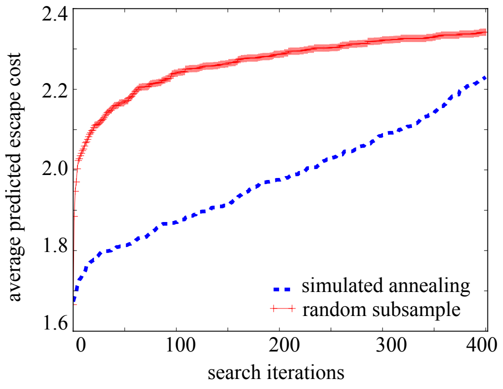

**Figure 4: Antibody design algorithms comparison ($\theta = 0$).**

there is a larger area of interaction between the L chain and G chain in the new antibody. Visually, the differences appear quite small, but make a significant difference in the ultimate breadth of binding, emphasizing the importance of a computational micro-level design approach.

## 8. DISCUSSION

We have, for the first time, formulated the virus evading antibodies problem as a Stackelberg game in which the antibody designer moves first, and the virus responds by escaping through the smallest number of protein sequence edits. We were able to exploit the problem structure to develop effective classification algorithms to significantly speed up the evaluation of escape cost for a particular antibody, as well as to predict escape cost, with little loss in solution quality. Moreover, we exhibited an antibody that is far more robust

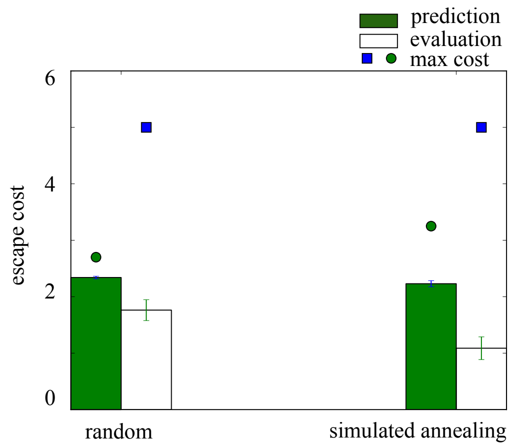

**Figure 5: Antibody design algorithms comparison after 400 iterations averaged over 80 search sequences ($\theta = 0$).**

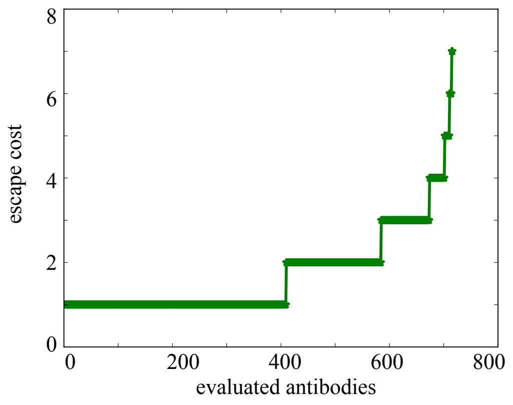

**Figure 6: Evaluated antibodies for $\theta = 0$, ranked by escape cost. The native antibody escape cost is 1.**

to virus escape than the native (i.e., the antibody found in nature to bind to the corresponding virus epitope).

## 8.1 Generalizability

While the proposed framework was developed in the context of antibody design (vaccination) for HIV viruses, nothing in it was specific to HIV, and the framework is therefore applicable directly to vaccination design for other viruses (e.g., influenza). Perhaps less obviously, it can also be applied almost without change to drug design: the primary difference is that drugs are typically not protein sequences. However, we can leverage chemical fragment databases specifically engineered for use in computational drug design as the core constituent component, turning our problem into the search over drug fragment space, and having little impact on the overall structure of the problem. Note that in the context of drug design (as well as vaccine design), nothing about our framework is unique to viruses, and can be generalized directly to other pathogens. This would, in principle, enable application, for example, to ebola vaccine design, as well as design of vaccines and antibiotic treatments for many bacterial infections. Moreover, our framework would tackle head-on the issue of antibiotic resistance, a problem of enormous and increasing concern for global health.

## 8.2 Limitations and Future Research

While our general approach shows much promise as an alternative route for antibody design to what is traditionally pursued, it has a number of limitations. First, we use protein sequence edit distance as a proxy for the difficulty of viral escape. In reality, a more meaningful measure is the number of nucleotide mutations required. This gives rise to two questions for future research: first, how good a metric is edit distance in predicting virus escape, and second, how can one map a metric based on nucleotide mutations into protein sequence edits (necessary for our search process). For the second question, a promising idea is to define a more generic cost function for virus escape, where cost of edit from one amino acid to another is measured in terms of corresponding mRNA mutations. If we could devise such a cost function, the approach developed in this paper is almost immediately applicable.

Another important consideration is that escape is not the lone survival criterion for a virus protein. Other important considerations are virus protein stability, and its ability to function and reproduce. For example, antibodies generally bind to a functional region of the virus, so that escaping an antibody will often imply weakened binding to a body protein critical for reproduction (such as CD4 in the case of HIV and sialic acid in the case of influenza). Modeling this balancing game is relatively direct in our framework: we would need to include additional binding energy constraints on virus escape, and our approach remains largely unchanged.

Yet another issue is the viability of an antibody. This involves two considerations: protein stability, and the ability to develop a vaccine that would elicit it. The first consideration can be handled directly in our framework: stability would entail an additional constraint on the energy of the antibody 3D structure, which can be evaluated using Rosetta. This additional constraint would, again, have little qualitative impact on the proposed approach. The second issue is a problem for all research in antibody design and characterization, and is not limited to our method in particular [11]. Addressing this issue requires both extensive "wet-lab" evaluation, and, ultimately, clinical evaluation, both clearly outside the scope of this paper.

A final issue worth noting is that typically we encounter a population (more precisely, a quasispecies) of viruses, rather than a single type, whenever mutation rates are high. The simplest way to integrate this aspect into the model is by considering multiple native virus proteins, and optimizing an antibody, or a collection of antibodies, that target all of these. Fundamentally, this doesn't change the overall approach, but clearly introduces additional challenges which likely require further computational advances (e.g., clustering of virus epitopes).

## 9. ACKNOWLEDGEMENTS

The authors would like to thank Dr. Jim Crowe, Dr Jens Meiler and graduate students in their group for help in understanding the data and the Rosetta software.

## REFERENCES

[1] M. Archetti. Evolutionarily stable anti-cancer therapies by autologous cell defection. *Evolution, Medicine, and Public Health*, pages 161–172, 2013.

[2] C. T. Bauch and D. J. Earn. Vaccination and the theory of games. *Proceedings of the National Academy of Sciences*, 101(36):13391–13394, 2004.

[3] CDC. Antibiotic resistance threats in the united states, 2013.

[4] CDC. 2014 ebola outbreak in west africa, 2014. http://www.cdc.gov/vhf/ebola/outbreaks/guinea/.

[5] G. Chapman, M. Li, J. Vietri, Y. Ibuka, D. Thomas, H. Yoon, and A. Galvani. Using game theory to examine incentives in influenza vaccination behavior. *Psychological Science*, 23(9):1008–1015, 2012.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[8] J. J. Gray, S. Moughon, C. Wang, O. Schueler-Furman, B. Kuhlman, C. A. Rohl, and D. Baker. Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of molecular biology*, 331(1):281–299, 2003.

[9] P. Hernandez-Leal, L. Fiedler-Cameras, A. Rios-Flores, J. A. González, L. E. Sucar, and S. M. Tonantzintla. Contrasting temporal bayesian network models for analyzing hiv mutations.

[10] H. H. Hoos and T. Stutzle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann, 2004.

[11] J. Huang, B. H. Kang, M. Pancera, J. H. Lee, T. Tong, Y. Feng, I. S. Georgiev, G.-Y. Chuang, A. Druz, N. A. Doria-Rose, L. Laub, K. Sliepen, M. J. van Gils, A. T. de la Pena, R. Derking, P.-J. Klasse, S. A. Migueles, R. T. Bailer, M. Alam, P. Pugach, B. F. Haynes, R. T. Wyatt, R. W. Sanders, J. M. Binley, and A. B. Ward. Broad and potent hiv-1 neutralization by a human antibody that binds the gp41-gp120 interface. *Nature*, 2014.

[12] S.-H. Huang, W. Zhou, A. Jong, and H. Qi. Game theory models for infectious diseases. In *Frontiers in the Convergence of Bioscience and Information Technologies*, pages 265–269, 2007.

[13] M. Jain, J. Pita, M. Tambe, F. Ordóñez, P. Paruchuri, and S. Kraus. Bayesian stackelberg games and their application for security at los angeles international airport. *SIGecom Exch.*, 7:10:1–10:3, June 2008.

[14] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[15] R. H. Lathrop and M. J. Pazzani. Combinatorial optimization in rapidly mutating drug-resistant viruses. *Journal of Combinatorial Optimization*, 3(2-3):301–320, 1999.

[16] R. H. Lathrop, N. R. Steffen, M. P. Raphael, S. Deeds-Rubin, M. J. Pazzani, P. J. Cimoch, D. M. See, and J. G. Tilles. Knowledge-based avoidance of drug-resistant hiv mutants. *AI Magazine*, 20(1):13, 1999.

[17] J. Liu, B. F. Kochin, Y. I. Tekle, and A. P. Galvani. Epidemiological game-theory dynamics of chickenpox vaccination in the usa and israel. *Journal of the Royal Society Interface*, 9(66):68–76, 2012.

[18] M. A. Nowak and R. May. *Virus Dynamics: Mathematical Principles of Immunology and Virology.* Oxford University Press, 2001.

[19] P. Paruchuri, J. P. Pearce, J. Marecki, M. Tambe, F. Ordóñez, and S. Kraus. Playing games with security: An efficient exact algorithm for Bayesian Stackelberg games. In *Proceedings of the Seventh International Conference on Autonomous Agents and Multiagent Systems*, pages 895–902, 2008.

[20] L. Richter, R. Augustin, and S. Kramer. Finding relational associations in hiv resistance mutation data. In *Inductive Logic Programming*, pages 202–208. Springer, 2010.

[21] UNAIDS. Hiv fact sheet, 2013.