Algorithms for Large-Scale Adversarial Decision Problems

By

Swetasudha Panda

Dissertation

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

in

Computer Science

August 10, 2018

Nashville, Tennessee

Dedicated to my parents, my brother and my grandparents.

ACKNOWLEDGMENTS

Working on this dissertation has been an exhilarating experience and its completion feels euphoric. First and foremost, I am very fortunate to have the supervision of Professor Yevgeniy Vorobeychik, I thank him for being such a great and inspiring mentor, for his invaluable advice, encouragement and all the research discussions and ideas that impact the foundations of this dissertation. Thank you so much for making me a part of this amazing research lab and for always creating such a positive and motivating atmosphere for research. I am really grateful to have had this opportunity.

I also want to thank my thesis committee members, Professor Jens Meiler for the very important research collaboration, for the research ideas and discussions, Professors Gautam Biswas, Benoit Dawant and Ivelin Georgiev for all the important advice and suggestions on this dissertation. I thank Professor Mark Ellingham for the valuable feedback on ideas when this research was in the initial stages.

Next, I thank all my friends, colleagues, seniors and role models who have encouraged, influenced and inspired me during this research. I thank our amazing lab members for the wonderful company and the constant encouragement. I thank Bo Li for being such a great friend and mentor, for all our research discussions, I am so very grateful that we met the first day after I joined Vanderbilt. I thank Alex Sevy for being an awesome co-author and collaborator, Haifeng Zhang for all the research discussions, for sharing knowledge on the various projects that we worked on and Chen Hajaj, for the great professional advice and valuable feedback, especially during the final stages of this dissertation. I also thank Jian Lou, Liang Tong, Ayan Mukhopadhyay, Sixie Yu and Rajagopal Venkat; I will miss the discussions ranging from science: research talks and paper reviews to history, sports and ancient medicine. I thank Nidhi Haryani for being the most dependable friend, ever since our IIT days, for our conversations on the graduate school experience (and every other possible topic). I thank my seniors Deepti Vikram and Rachana Vidhi, for the words of support and encouragement throughout the progress of this dissertation.

Finally and most importantly, I thank my family: my parents and my brother Subhadarshi, for the moral support, for always being there to share the phases of ecstasy and also the challenging times. I am so grateful that you have always motivated me to put forth the greatest effort. None of these accomplishments would have been possible without your positive reinforcement and all the time-tested advice. I also thank my dear grandparents for the priceless blessings, and for all the love and affection. Everything that I have achieved (and will achieve in the future) is dedicated to my family.

ABSTRACT

Decision making in the presence of uncertainty received a major focus of traditional Artificial Intelligence (AI) research. In an adversarial environment with competing agents, this challenge is compounded by interdependence: a combination of the agents' strategies determines their respective gains. Game theory provides a mathematical foundation to reason about competing strategic agents each pursuing their own interests. Recently, security games have seen tremendous success and actual deployment in homeland security, especially the models of interaction between the defender and the attacker as a Stackelberg (two player, one shot) game. This dissertation considers this conceptual idea of Stackelberg games as a natural modeling paradigm in domains beyond physical security; in cybersecurity and immunology, antibody design in particular. A key challenge is the enormous combinatorial search space corresponding to the strategy space of the players. This dissertation presents major algorithmic contributions for highly scalable decision-making in adversarial environments.

It begins with the framework of plan interdiction games, motivated by attack plan models of the adversary in cybersecurity domains. While interdiction of deterministic plans is reasonably scalable, the approach scales poorly when modeling uncertainty with the attacker as a markov decision process (MDP). This research presents scalable algorithms using a) factored MDP solution approaches, b) value function approximation over Boolean space with Fourier representation, c) constraint generation in linear programs and d) model-free reinforcement learning. An interesting application of plan interdiction is in designing treatment therapies (vaccines) against infectious diseases, where a virus (adversary) plans to escape the antibody (defender) through a series of mutations. This dissertation presents algorithms for robust antibody design, with combinatorial optimization in the enormous protein sequence space, using data-driven graphical models of interaction and mixed integer linear programming approaches to solve the bi-level optimization problem corresponding to the game.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Motivation

In recent years, Artificial Intelligence (AI) research has made remarkable progress in addressing high impact problems in the society. AI algorithms play a pivotal role in improving and revolutionalizing our day-to-day lives, health and well being as well as safety and security. Many of these real-world problems involve strategic planning and decision making with multiple agents interacting in an environment. These problems can be modeled using the mathematical concepts in AI planning, decision theory and game theory. A crucially important modeling criterion in such real-world problems is that of *resilience*, i.e., identifying and detecting unexpected events and emergencies and accounting for those. Indeed, resilience to the unexpected is an important theme in the area of autonomic computing [1].

In most research in autonomic computing, resilience is incorporated in terms of previously observed and recorded adverse situations [2, 3, 4], with the assumption that such situations will be encountered with a frequency consistent with past occurrences. However, events that are intentional attacks on the system, are not necessarily similar to other general and non-malicious adversities, that can be reasonably predicted from past observations. Indeed, there is substantial evidence that real-world attackers in both the cyber and the physical domains *respond* to deployed defensive measures [5, 6, 7] and often change their behavior and patterns in order to circumvent the defensive measures and to simultaneously achieve their malicious goals [7, 8]. Therefore, if attackers are considered similarly as typical adversities, the defender will function myopically and hence, will suffer from a fundamental disadvantage. In this scenario, it is imperative that the defender proactively reasons about the attacker's strategies and anticipates the attacker's reactions to any deployed defense strategies.

The motivating domain we begin with is that of cybersecurity. In recent years, the cyber space has grown with tremendous sophistication, and the complexity of cyber attacks has evolved as well. Recent incidences of massive data breach incidents have resulted in a loss of millions of dollars in addition to the loss of enormous critical data [9]. In previous literature, game theory has been used to capture the fact that the attackers respond to defensive measures, such as in intrusion detection systems [10, 11]. In cybersecurity, game theoretic approaches typically make one of the following assumptions: a) the simultaneous move assumption, i.e., the attacker acts without any knowledge of the deployed defensive measures, b) the attacker's utility function is accurately known, and c) the attacker and the defender operate on a pair of strategies that are optimal against each other. However, there are limitations since these assumptions do not necessarily hold in many situations. Many of these limitations have been alleviated by the pivotal idea that the attacker responds after observing the deployed defensive posture. These *security game* models [12] have achieved real-world deployment, recognition and success in a broad array of physical security applications, e.g., assigning security resourecs and checkpoints in airports and TSA screenings, securing the major ports and protecting wild-life from poachers [13, 14, 15].

The key modeling concept in these *security games* is the idea of a leader-follower interaction defined as a *Stackelberg game*. In a two player Stackelberg game, the leader moves first, i.e., it commits to a *strategy*. The follower observes the leader's strategy and responds with its *best response*. This framework captures the agents' interactions in security settings because of the following reasons: a) the defender's resources in a security setting are limited and b) the attacker observes any deployed defensive measures and then responds with its best strategy. The conceptual insight of Stackelberg security games is that of proactive defense. Ultimately, these games translate into adversarial optimization problems, incorporating the defender's and the attacker's objectives. *In this dissertation, our research focus is two-fold: a) to extend this concept of Stackelberg game interaction to solve adversar-*

ial decision problems in domains beyond physical security (in particular, cybersecurity as described above and also immunology which we will present subsequently), and b) to develop scalable algorithms that address the combinatorial search space in the adversarial optimization problems corresponding to these games.

We begin the dissertation with the framework of *plan interdiction* in the context of cybersecurity. In particular, cyber attacks can be modeled as plans or sequential decisions where the attacker starts with a certain configuration or set of capabilities and plans towards a malicious goal through a series of actions. The defender's goal then is to *interdict* such a multi-stage attack plan. In order to execute optimal and proactive defense, the defender must not only account for previous attacks but also consider possible alternative attack plans. We denote the defender's defensive measures as the *mitigation strategies*. Once the mitigations are deployed, the attacker actively attempts to circumvent these mitigations and simultaneously tries to accomplish its own objectives. We capture this interaction as a Stackelberg game in which the defender is the leader and commits to a mitigation strategy. The attacker is the follower that observes the defender's mitigations and optimally responds to those by constructing an optimal plan. Previous research has demonstrated algorithms for plan interdiction with the attacker modeled as a deterministic planner. In this dissertation, we consider the more general attacker model of planning under uncertainty, using the Markov Decision Process (MDP) framework, denoted *MDP interdiction*. Solving large-scale MDPs as such poses a critical challenge in terms of computational intractability. Moreover, the attacker MDP solution only constructs the 'inner loop' (details follow in the next section) of the bi-level optimization corresponding to the Stackelberg game. Therefore, there are additional challenges corresponding to the defender's combinatorial strategy space in the 'outer loop'. Consequently, there are a number of algorithmic challenges in solving MDP interdiction at scale.

A completely different domain, critically important nevertheless, is that of immunology and vaccine design. Vaccination therapies are important tools in the battle against infectious diseases such as HIV and influenza. 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 [16]. There is neither a vaccine nor a cure available to date for HIV. With an incidence rate of around 2 million each year and 1.6 million deaths in 2012, HIV infections continue to be a major global health issue. A vaccine stimulates the immune system to produce antibodies that bind to the vaccine substance. Antibodies and viruses are essentially protein sequences composed of chains of amino acids. In the primary structure, each position in the sequence is occupied by one of the 20 possible amino acids. Each protein has an active site called the *binding positions* at which it interacts and bonds with other proteins to form energy minimized configurations or *complexes*. The primary function of an antibody is to defend against external infections and pathogens e.g., the viruses. An antibody deactivates or kills a virus by *binding* at the binding positions. To simplify terminology, we henceforth focus our discussion on viruses, although our methods are general.

The primary goal in antibody design is that it binds the virus that it is designed to be effective against. In most cases, there are a number of variants of such a virus sequence. This gives rise to the idea of *broadly binding 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' [17]. Binding is usually specific and relatively small changes in structure can lead to destabilization of binding. Antibodies stimulated no longer bind to the target pathogen if it has mutated. Many viruses have the ability to exhibit diverse *mutations* to bring about such change and destabilization, such that these can *escape* binding to the antibody. In particular, some viruses e.g., HIV have alarmingly high mutation rates [18], making it extremely difficult to design effective vaccines / antibodies against the catastrophic disease. For example, a single person infected with HIV carries more variations of the virus than all the influenza strains isolated worldwide. The goal in vaccination is to ensure that the produced antibodies are subsequently capable of binding relevant strains of the live pathogen, rapidly militating immune response against disease. To summarize, a core question in vaccination research is how to *design* or *discover* an antibody that is effective against a large number of pathogens and continues to be effective in the presence of rapid mutations.

The modeling environment in vaccine design is undoubtedly completely different as compared to our earlier discussion on cybersecurity. However, we identify a common theme which is that of *proactive defense*. In this domain, the defense is against the external infection or equivalently, attack by the viruses. Given this scenario, we make the following observations. We posit that the 'series of virus mutations until escape' can be viewed as a planning problem of the virus. The immune system / antibody is the defender and its goal is to interdict such an escape plan. Therefore, the conceptual modeling frameworks in plan interdiction have direct applications in antibody design. Consequently, we present the first game theoretic model of molecular-level interaction between the immune system (antibody) and the virus. Specifically, we formulate the interaction between an antibody and a virus as a Stackelberg game in which the vaccine designer (drug designer, etc) 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 (i.e., escape binding to it) through a series of local mutations. So, the 'designer' chooses an antibody, and the virus responds through the shortest sequence of mutations leading to escape. We confront a number of technical and conceptual challenges in our algorithmic formulations for antibody design, a high-level overview of which follows in the next section.

1.2 Research Challenges and Contributions

1.2.1 MDP Interdiction

A key research challenge in cybersecurity is to model the attacker's multi-stage plan with high resolution. In the previous research on plan interdiction (Letchford and Vorobeychik et al. [19]), such evasion has been modeled in a Stackelberg framework, both in the context of deterministic (PDDL-based) planning, and planning with Markov decision processes (MDPs). The Stackelberg leader removes a subset of attack actions, and the adversary computes an optimal plan in the restricted action space. In solving the interdiction problem, the defender takes into account both the cost of mitigation strategies and the benefit in terms of preventing the attacker from reaching a subset of goals. The model is general-sum. The attacker's decision problem involves planning costs. The defender's decision problem involves costs of mitigations.

While interdiction of deterministic plans has been demonstrated to be reasonably scalable, the approach does not scale when the attacker is modeled as an MDP. The central challenge with MDP interdiction is the exponential size of the state space in the number of state variables. This leads to intractability in representation as well as in computing the optimal value function and the optimal policy. Moreover, as we have described earlier, Stackelberg plan interdiction translates to a bi-level optimization problem. The defender's optimization forms the outer level and the attacker MDP solution is the inner level in the optimization. Therefore, the computational challenge is compounded when we consider the combinatorial search space of the defender. Finally, a major challenge in plan interdiction is that the defender is typically not knowledgeable about the attacker's planning problem, such as the initial vulnerabilities and attacker's access and capabilities. Such uncertainties can be captured using a Bayesian Stackelberg game framework. However, the resulting Bayesian Stackelberg game is infeasible for state of the art plan interdiction approaches for even small problem instances.

To address the above challenges, we propose models for MDP interdiction and develop several solution approaches. We formulate MDP interdiction with factored representation of states and present two alternative interdiction approaches in terms of the a) action space and b) state space of the MDP. We develop scalable algorithms to solve the interdiction problem on realistic MDP problem domains. Scalability in factored MDPs has been achieved by two basic approaches: a) exploiting structure in the MDP transition model and reward function and b) value function approximation. Factored MDPs represent the complex state space using state variables and the transition model using a dynamic Bayesian network. This representation allows an exponential reduction in the representation size of structured MDPs. Moreover, efficient approximate solution algorithms have been proposed that exploit structure in factored MDPs.

Leveraging factored representation, we now present a high-level sketch of our first MDP interdiction approach. Starting with the approximation methods for factored MDPs, we develop a mixed-integer programming approach for factored MDP interdiction. In doing so, we face two challenges: 1) effective basis representation, and 2) a super-exponential set of constraints corresponding to alternative evasion plans for the attacker. To address the first challenge, we propose using a Fourier basis over a Boolean hypercube to represent the value function over a binary factor space. While there always exists an exact Fourier basis for functions over a Boolean space, the representation is exponential in size. We address this challenge by developing iterative basis generation methods. Addressing the second challenge of intractably large constraint space, we develop a novel constraint generation algorithm using a combination of linear programming factored MDP solvers and novel heuristics for attack plan generation. Our algorithms achieve dramatically improved scalability while resulting in near-optimal interdiction decisions. Specifically, our algorithms scale up to more than 2^{60} state space sizes on realistic MDP problem domains from the international planning competition compared to the exact interdiction baseline which does not scale beyond toy problem sizes (about 2^{8}).

While our scalable algorithms achieve significant advancement in solving MDP interdiction problems, we continue to face limitations in scalability. For example, capturing uncertainty about the attacker remains computationally expensive. Another major bottleneck is that the difficulty of solving the factored MDP grows exponentially in the number of interdependencies among state variables. An alternative approach is to use model-free reinforcement learning, which can scale significantly better when we use function approximation to represent the action-value function. We propose a novel interdiction model in which the defender modifies the initial state of the attacker. The attacker then computes an optimal policy starting from the modified initial state. This is quite general: for example, we can model prior interdiction approaches by adding action-specific preconditions as state variables. However, we show that this change enables significantly simpler and far more scalable interdiction techniques which rely on single-level integer linear programming, in contrast to difficult bi-level problems faced in prior art. A very high-level overview of the solution approach is the following. We learn a general value function for the attacker's initial state space, and then solve the interdiction problem in one step, without having to iteratively solve the planning problem. Accordingly, we present interdiction algorithms for both linear and non-linear value function approximation. Additionally, this interdiction model with the associated scalability gains allows us to incorporate uncertainty about the attacker by capturing uncertainty over a subset of initial state variables. With a linear value function, by linearity of expectation, we transform the defender's interdiction problem into an integer linear program for Bayesian interdiction. We also demonstrate Bayesian interdiction in case of non-linear value function approximation.

1.2.2 Applications in Antibody Design

Our ultimate goal is to design or optimize antibody sequences that a) bind to a broad array of diverse viruses and b) continue to bind in the face of the rapid virus mutations. First, we consider the goal of designing a broadly binding antibody. When we say 'design an antibody', our goal is to optimize the amino acid sequence at the binding positions of the antibody. In our proposed algorithms, we begin with the *native* antibody (the one that is generated by the immune system) and then, optimize the binding positions against a set of target viruses that we denote as the *virus panel*. In doing so we face the following challenges. First, typically the binding position has a length of about 30 or more amino acid residues on each side, resulting in a combinatorial search space of $\geq 20^{60}$. Second, it is difficult to quantify the criteria for binding and stability between the various antibody and virus sequences. For this purpose, we leverage ROSETTA, a premier computational protein modeling tool [20]. The major challenge is that ROSETTA evaluations are extremely expensive computationally (which could take nearly an hour for a single antibody-virus pair scoring evaluation, as it makes use of its own sophisticated amalgam of local search techniques to simulate a binding complex). Finally, in previous literature, computational protein design has been used successfully to design broadly binding proteins in general. In the simplest case, protein design involves optimizing the amino acid sequence of a protein to accommodate a desired 3-D conformation. This approach has been extended to related tasks such as protein-protein interface design, de novo design of protein binding molecules, design of self-assembling protein nano-cages, etc. However, the state of the art protein design algorithms for broadly binding antibodies (defined as multi-specificity design and decsribed in section 2.5.1.1) suffer from large energetic barriers that limit sampling in sequence space, resulting in sub-optimal designs (Sevy et al. [21]). In addition, these methods are severely limited in scale by the size and number of states that can be included. Since these approaches rely on local search, these do not guarantee a global solution to the sequence optimization problem.

To address the above mentioned challenges, we develop a method that integrates structural modeling with integer linear programming to enable a fast global search through large ensembles of target states. The combination of these methods allows us to surpass protein design limitations that have been seen up to this point. This algorithm increases the predicted *breadth* (fraction of viruses in a panel to which the antibody binds) of the naturally-occurring anti-HIV antibody VRC23 to 100% against a panel of 180 divergent HIV viral strains.

Having tackled the problem of broadly binding antibody design, we now move on to the problem of robust antibody design against escaping virus sequences. Following the adversarial modeling framework in plan interdiction, we model the interaction between the antibody and the virus as a Stackelberg game. The designer-virus game poses two challenges: 1) enormous search space for both the designer and the virus ($\geq 20^{60}$), and 2) determining whether an arbitrary antibody-virus pair bind and form a stable complex. To tackle the former challenge, we propose, and compare the performance of, several stochastic local search heuristics, again 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 and as mentioned earlier, ROSETTA can be extremely time consuming even for a single evaluation. 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 the ‘inner loop’ (virus escape) evaluations restricted to a small set of candidate antibodies predicted to be difficult to escape. To quantify the difficulty in escaping an antibody, we define *escape cost* as the minimum number of mutations that the virus needs to execute in order to escape binding to the antibody. This game-theoretic antibody design algorithm achieves a robust antibody that has a superior escape cost of 7 as compared to the native antibody that has an escape cost of 1.

Our game theoretic approach is promising for designing robust antibodies. However it suffers from a fundamental problem: it relies on local search algorithms and therefore, fails to compute the globally optimal antibody sequence. Moreover, all computations consider binding energy. However, stability energy of the complex is very important to determine the feasibility of a designed antibody sequence (stability scores can also be computed using ROSETTA modeling). To address these challenges, we formulate the bi-level optimization problem (corresponding to the antibody design game) in terms of the combined binding and stability score (denoted as z-score). We leverage problem structure in terms of a pairwise amino acids interaction model to formulate the problem as a mixed integer linear program. Typically, it is extremely challenging to compute an optimal solution to such bi-level problems with integer variables. We present a compact single-level mixed integer program formulation by relaxing the integrality constraints and then obtaining the dual linear program. We demonstrate with a proof that our compact formulation computes the optimal (integer) global solution despite the relaxation and duality transformations. In addition, it allows us to incorporate a panel of escaping viruses into the optimization problem. Through a series of simulation experiments, we demonstrate the efficacy of our proposed antibody design algorithm.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows. In the following chapter 2, we present an overview of the background and the related work. Subsequently, the dissertation is divided into two parts. Part 1 focuses on MDP interdiction. We introduce the MDP interdiction problem in chapter 3. In chapter 4, we present scalable algorithms for interdiction in factored MDPs where the defender manipulates the action space of the attacker. This research has been published in the **Conference on Uncertainty in Artificial Intelligence (UAI) 2017**. In chapter 5, we present our alternative initial state interdiction approach in factored MDPs. We present scalable algorithms for state interdiction with factored representation, value function approximation and reinforcement learning. This research has been published in the **International Joint Conference on Artificial Intelligence (IJCAI) 2018**. Part 2 focuses on the modeling concepts in plan interdiction applied to the antibody design problem. In chapter 6, we summarize the major goals in antibody design. In chapter 7, we present our first algorithm for broadly binding antibody design. This research has been published in the **PLOS Computational Biology (2018)** journal. In chapter 8, we formulate antibody design as a Stackelberg game and present solution approaches using machine learning guided stochastic local search. This research has been published in the **Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)** and **Association for the Advancement of Artificial Intelligence (AAAI) Conference 2015**. In chapter 9, we present algorithms that leverage the structure in antibody-virus binding interactions to formulate the antibody design game as a mixed integer program. Subsequently, we present the compact optimization framework for computing a globally optimal solution. This research is under review in a conference. We conclude and discuss future work in chapter 10.

Chapter 2

BACKGROUND AND RELATED WORK

In this section, we present several core concepts which constitute the preliminary and background material for this dissertation. We also outline the previous literature that is relevant to our research.

2.1 Computational Game Theory

As we have declared previously, a major goal of this dissertation is to extend the conceptual idea of Stackelberg games to various adversarial decision problems. In this section, we begin with a very high-level overview of some of the key terminology in computational game theory, followed by a very short introduction to Stackelberg games. For detailed reference, we redirect the reader to the comprehensive texts [22, 23]. Finally, we summarize some of the recent literature on Stackelberg security games.

Game theory provides a structured framework to study interactive decision-making in multi-agent settings, to analyze how different entities (the players) make decisions by taking some form of strategic interdependence into account. It is is the study of optimal decision-making under competition when one individual's decisions affect the outcome of a situation for all other individuals involved. Formally, game theory is defined as the study of mathematical models of conflict and cooperation between intelligent rational decision-makers. A game is a formal definition of a strategic situation. and has three major components: the players (the agents that make the decisions in the game), the strategies for each player, and the payoffs that estimate the costs and benefits to the players based on the potential outcomes of the game. The players can move simultaneously (e.g., rock-paper-scissors) or sequentially (e.g., chess). The payoff is a scalar quantity, that reflects the utility or desirability of an outcome to a player. When the outcome is random, payoffs are usually weighted with their probabilities. The expected payoff incorporates the player's attitude towards risk. A player's payoff depends not only on its own actions (or strategies), but also on the actions of other players in the game. A player is said to be rational if it seeks to play in a manner which maximizes its own payoff. A strategy is one of the given possible actions of a player. A *mixed strategy* is an active randomization, with given probabilities, that determines the player's decision. As a special case, a mixed strategy can be the deterministic choice of one of the given pure strategies. A player's strategy is (strictly) dominant if, for any combination of actions by other players, it gives that player a strictly higher payoff than its other strategies. A player's strategy is (strictly) dominated if there exists another strategy giving that player a strictly higher payoff for all combinations of actions by other players.

A *cooperative game* (or coalitional game) is a game with competition between groups of players ('coalitions') due to the possibility of external enforcement of cooperative behavior. A *non-cooperative game* is a game with competition between individual players in which only self-enforcing alliances (or competition between groups of players) are possible due to the absence of external means to enforce cooperative behavior, as opposed to cooperative games. The strategic form (also called normal form) is the basic type of game studied in noncooperative game theory. A game in strategic form lists each player's strategies and the outcomes that result from each possible combination of choices. An outcome is represented by a separate payoff for each player, which is a number (also called utility) that measures how much the player likes the outcome. The extensive form, also called a game tree, is more detailed than the strategic form of a game. It is a complete description of how the game is played over time. This includes the order in which players take actions, the information that players have at the time they must take those actions, and the times at which any uncertainty in the situation is resolved. A game in extensive form may be analyzed directly or can be converted into an equivalent strategic form. A game is said to be *zero-sum* if for any outcome, the sum of the payoffs to all players is zero. In a two-player zero-sum game, one player's gain is the other player's loss. A game has *perfect information* if each player, when making any decision, is perfectly informed of all the events that have previously occurred, including the 'initialization event' of the game.

A *solution concept* is a formal rule for predicting how a game will be played. These predictions called solutions describe which strategies will be adopted by the players and, therefore, predict the result of the game. In the traditional solution concept called *Nash equilibrium*, no player has an incentive to deviate from its chosen strategy, while the strategies of the other players remain unchanged. A game in strategic form does not always have a Nash equilibrium in which each player deterministically chooses one of his strategies. However, players may instead randomly select from among these pure strategies with certain probabilities. Any finite strategic-form game has an equilibrium if mixed strategies are allowed.

2.1.1 Stackelberg Games

The Stackelberg model [24] was originally introduced to capture market competition between a leader (e.g., a leading firm in some area) and a follower (e.g., an emerging start-up). In a Stackelberg (two-stage, one-shot) game, the leader commits to a strategy first, and the follower, selfishly optimizes its own reward, considering the action chosen by the leader. The two players i.e., the leader and the follower in a Stackelberg game need not represent individuals, but could also be groups that cooperate to execute a joint strategy, such as a police force or a terrorist organization. Each player has a set of possible pure strategies, or equivalently, the actions. A mixed strategy allows a player to play a probability distribution over pure strategies. Payoffs for each player are defined over all possible pure-strategy outcomes for both the players. The payoff functions are extended to mixed strategies by taking the expectation over pure-strategy outcomes. The follower observes the leader's strategy first, and then acts to optimize its own payoffs. The typical solution concept applied to these games is Strong Stackelberg Equilibrium (SSE) [25], which assumes that the leader will choose an optimal mixed (randomized) strategy based on the assumption that the follower will observe this strategy and choose an optimal response.

2.1.2 Stackelberg Security Games

The Stackelberg leader-follower paradigm fits many real-world security situations. As we have discussed earlier, Stackelberg game models have recently seen tremendous success in security applications, with the defender as the leader, and the attacker as the follower. Commonly in such models of Stackelberg Security Games (SSGs) [12] the defender (e.g., the security forces) acts first by committing to a patrolling or inspection strategy, and the attacker chooses where to attack after observing the defenders choice. The key conceptual insight is that defense needs to be proactive, optimally accounting for attacker's response to a defensive posture. These models span game-theoretic approaches to security at airports, ports, transportation, shipping and other infrastructure [26, 27, 28, 29, 30, 31]. The deployed applications include ARMOR at the Los Angeles Airport (LAX) deployed in 2007 (Pita et al. [13]) to randomize checkpoints on the roadways entering the airport; IRIS (Tsai et al. [32]), a game-theoretic scheduler for randomized deployment of the U.S. Federal Air Marshal Service (FAMS) that has been deployed since 2009; and PROTECT (Shieh et al. [33]) which is deployed for generating randomized patrol schedules for the U.S. Coast Guard in Boston, New York, Los Angeles, and other ports around the United States. Green security games (Fang et al. [15, 34]) focus on defending against environmental crimes. These problems exhibit a spatial and temporal aspect and behavioral adversary models play an important role. An example is PAWS which is a wildlife protection assistant system that has been extensively evaluated in Malaysia and the Queen Elizabeth National Park in Uganda. Stackelberg games models have also been applied to protection against urban crime (Zhang et al. [35]). Such models have been evaluated for deterring fare evasion within the Los Angeles Metro System (Yin et al. [36], TRUSTS) and for crime prevention at the University of Southern California.

2.2 Planning in Artificial Intelligence

As we have discussed earlier, in this dissertation, we study the plan interdiction problem in which the attacker solves a planning problem. Ability to accomplish desired goals despite uncertainty is a well-established area of research in artificial Intelligence, a testament to which is the rich literature on solving Markov Decision Processes [37, 38], as well as related efforts that aspire to compute robust plans [39]. Moreover, in reinforcement learning [40] (which we elaborate in the next section), the planning problem additionally involves learning about the environment based on observations. In this section, we formalize the concept of planning as studied in artificial intelligence. We start with classical planning in deterministic domains and then consider generalizations to capture uncertainty.

2.2.1 Classical Planning

Formally, a classical (deterministic) planning problem [41] is a tuple $X, A, \mathbf{x}_0, G, R, c$, where X is the set of literals (binary variables) which represent the state of the world relevant for the planning problem, A is the set of actions, $\mathbf{x}_0$ is the set of literals which are initially true (i.e., the initial state of the world), and G is the set of goals. A plan action $a \in A$ is characterized by a set of preconditions, i.e., the set of literals that must be true in the current state for the action to be applicable, and a set of effects, which either add literals from current state, or delete these, thereby determining transition from one state to another. A reward function R_l assigns a value (utility) to each goal literal $l \in G$ (assuming that the total utility is additive). Finally, c_a is the cost of taking an action a. A solution to this planning problem is a plan π, which is a sequence of actions. A number of effective approaches exist for solving such planning problems at scale [42].

2.2.2 Planning with Uncertainty: Markov Decision Processes (MDPs)

Our work builds on solution approaches for discounted infinite-horizon MDPs, and particularly for factored MDPs, which we now introduce.

Formally, a discounted infinite-horizon MDP [37] is defined as a tuple $\mathscr{D} = (\mathbf{X}, A, R, P, \gamma)$ where $\mathbf{X}$ is a finite set of $|\mathbf{X}| = N$ states; A is a finite set of actions; R is a *reward function* $R : \mathbf{X} \times A \mapsto \mathbb{R}$, in which $R(\mathbf{x}, a)$ is the reward obtained by the agent in state $\mathbf{x}$ after taking action a; P is a *Markovian transition model* where $P(\mathbf{x}'|\mathbf{x}, a)$ is the probability of moving from state $\mathbf{x}$ to $\mathbf{x}'$, after taking action a; and $\gamma \in [0, 1)$ is the discount factor which exponentially discounts future rewards. It is well-known that such MDPs always admit an optimal *stationary deterministic policy*, which is a mapping $\pi : \mathbf{X} \mapsto A$, where $\pi(\mathbf{x})$ is the action the agent takes at state $\mathbf{x}$ [43].

Each policy can be associated with a *value function* $\mathscr{V}_\pi \in \mathbb{R}^N$, where $\mathscr{V}_\pi(\mathbf{x})$ is the discounted cumulative value obtained by starting at state $\mathbf{x}$ and following policy π. Formally,

$$\mathscr{V}_\pi(\mathbf{x}) = E_\pi\left[\sum_{t=0}^{\infty} \gamma^t R(\mathbf{X}^t, \pi(\mathbf{X}^t)) \middle| \mathbf{x}^{(0)} = \mathbf{x}\right],$$

where $\mathbf{X}^{(t)}$ is a random variable representing the state of the system after t steps.

2.2.2.1 Factored Representation

Factored MDPs (Guestrin et al. [44]) exploit problem structure to compactly represent MDPs. The set of states is described by a set of *random state variables* $\mathbf{X} = \{X_1, \ldots, X_n\}$. Let $\text{Dom}(X_i)$ be the domain of values for X_i. A state $\mathbf{x}$ defines a value $x_i \in \text{Dom}(X_i)$ for each variable X_i. *Throughout, we assume that all variables are Boolean.* The transition model for each action a is compactly represented as the product of local factors by using a DBN. Let X_i denote the variable X_i at the current time and X_i' the same variable at the next time step. For a given action a, each node X_i' is associated with a conditional probability distribution (CPD) $P_a(X_i'|\text{Parents}_a(X_i'))$. The transition probability is given by

$P_a(\mathbf{x}'|\mathbf{x}) = \prod_i P_a(x_i'|\mathbf{x}[\text{Parents}_a(X_i')])$, where $\mathbf{x}[\text{Parents}_a(X_i')]$ is the value in $\mathbf{x}$ to the variables in $\text{Parents}_a(X_i')$. The complexity of this representation is linear in the number of state variables and exponential in the number of variables in the largest factor. The reward function is represented as the sum of a set of localized reward functions. Let $R_1^a, \ldots, R_r^a$ be a set of functions, where the scope of each R_i^a is restricted to the variable cluster $\mathbf{W}_i^a \subset \{X_1, \ldots, X_n\}$. The reward for taking action a at state $\mathbf{x}$ is then $R^a(\mathbf{x}) = \sum_{i=1}^{r} R_i^a(\mathbf{W}_i^a) \in \mathbb{R}$.

2.2.2.2 Linear Programming Methods for Solving MDPs

A common method for computing an optimal policy of an MDP is by using the following linear program (LP):

$$\min \sum_{\mathbf{x}} \alpha(\mathbf{x}) V(\mathbf{x}) \tag{2.1a}$$

$$\text{s.t.: } \forall \mathbf{x}, a, V(\mathbf{x}) \geq R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a) V(\mathbf{x}'). \tag{2.1b}$$

where the variables $V(\mathbf{x})$ represent the value function $\mathcal{V}(\mathbf{x})$, starting at state $\mathbf{x}$. The *state relevance weights* αs are such that $\alpha(\mathbf{x}) > 0$ and $\sum_{\mathbf{x}} \alpha(x) = 1$. The optimal policy π^* can be computed as the greedy policy with respect to $\mathcal{V}^*$, $\pi^* = \arg\max_a [R(\mathbf{x}, a) + \gamma \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a) V(\mathbf{x}')]$. The dual of this LP (dual LP) maximizes the total expected reward for all actions:

$$\max \sum_{\mathbf{x}} \sum_{a} \phi_a(\mathbf{x}) R(\mathbf{x}, a) \tag{2.2a}$$

$$\text{s.t.: } \forall \mathbf{x}, a, \quad \phi_a(\mathbf{x}) \geq 0 \tag{2.2b}$$

$$\forall \mathbf{x}, \sum_a \phi_a(\mathbf{x}) = \alpha(\mathbf{x}) + \gamma \sum_a \sum_{\mathbf{x}'} P(\mathbf{x}|\mathbf{x}', a) \phi_a(\mathbf{x}'). \tag{2.2c}$$

where $\phi_a(\mathbf{x})$ called the *visitation frequency* for state $\mathbf{x}$ and action a is the (discounted) expected number of times that state $\mathbf{x}$ will be visited and action a will be executed in this state. There is a one-to-one corresponedence between policies in the MDP and feasible solutions to the dual LP.

2.2.2.3 Factored MDPs and Approximate Linear Programming

In the case of factored MDPs, there is no guarantee that the structure extends to the value function [45] and linear value function approximation is a common approach. A factored (linear) value function $\mathscr{V}$ is a *linear function* over a set of *basis functions* $H = \{h_1, \ldots, h_k\}$, such that $\mathscr{V}(\mathbf{x}) = \sum_{j=1}^{k} w_j h_j(\mathbf{x})$ for some coefficients $\mathbf{w} = (w_1, \ldots, w_k)'$, where the scope of each h_i is restricted to some subset of variables $\mathbf{C}_i$. The approximate LP corresponding to (2.1) is given by Guestrin et al. [44]:

$$\min \sum_i \alpha_i w_i \tag{2.3a}$$

$$\text{s.t.: } \forall a, \max_{\mathbf{x}} \{R^a(\mathbf{x}) + \sum_i w_i [\gamma g_i^a(\mathbf{x}) - h_i(\mathbf{x})]\} \leq 0. \tag{2.3b}$$

where for basis h_i, $\alpha_i = \sum_{\mathbf{x}} \alpha(\mathbf{x}) h_i(\mathbf{x})$ is the factored equivalent of α and $g_i^a(\mathbf{x}) = \sum_{\mathbf{x}'} P(\mathbf{x}'|\mathbf{x}, a) h_i(\mathbf{x}')$ is the factored representation of expected future value. The non-linear constraint in LP (2.3) can be represented by a set of linear constraints using approaches similar to variable elimination in cost networks. The factored dual approximation LP [46] is defined on a set of variable clusters $\mathscr{B} \supseteq \mathscr{B}_{FMDP}$ where $\mathscr{B}_{FMDP} = \{\mathbf{W}_1^a, \ldots, \mathbf{W}_r^a : \forall a\} \cup \{\mathbf{C}_1, \ldots, \mathbf{C}_k\} \cup \{\Gamma_a(\mathbf{C}_1), \ldots, \Gamma_a(\mathbf{C}_k) : \forall a\}$, $\Gamma_a(\mathbf{C}) = \cup_{X_i \in \mathbf{C}} \text{PARENTS}_a(X_i) = \text{Scope}[g]$ is the set of parent state variables of variables in $\mathbf{C}$ (Scope$[h]$) in the DBN for action a. This factored dual LP is given by:

$$\max \sum_{a} \sum_{j=1}^{r} \sum_{\mathbf{w}_j^a \in \mathrm{Dom}[\mathbf{W}_j^a]} \mu_a(\mathbf{w}_j^a) R_j^a(\mathbf{w}_j^a)$$

s.t.:

$$\forall i = 1, \ldots, k :$$

$$\sum_{\mathbf{c} \in \mathrm{Dom}[\mathbf{C}_i]} \mu(\mathbf{c}) h_i(\mathbf{c}) = \sum_{\mathbf{c} \in \mathrm{Dom}[\mathbf{C}_i]} \alpha(\mathbf{c}) h_i(\mathbf{c})$$

$$+ \gamma \sum_{a} \sum_{\mathbf{y} \in \mathrm{Dom}[\blacksquare_a(\mathbf{C}_i')]} \mu_a(\mathbf{y}) g_i^a(\mathbf{y}) \tag{2.4a}$$

$$\forall \mathbf{B}_i, \mathbf{B}_j \in \mathscr{B}, \forall \mathbf{y} \in \mathrm{Dom}[\mathbf{B}_i \cap \mathbf{B}_j], \forall a :$$

$$\sum_{\mathbf{b}_i \sim [\mathbf{y}]} \mu_a(\mathbf{b}_i) = \sum_{\mathbf{b}_j \sim [\mathbf{y}]} \mu_a(\mathbf{b}_j) \tag{2.4b}$$

$$\forall \mathbf{B} \in \mathscr{B}, \forall \mathbf{b} \in \mathrm{Dom}[\mathbf{B}], \forall a, \mu_a(\mathbf{b}) \geq 0 \tag{2.4c}$$

$$\mu(\mathbf{b}) = \sum_{a'} \mu_{a'}(\mathbf{b}), \tag{2.4d}$$

$$\sum_{\mathbf{b}' \in \mathrm{Dom}[\mathbf{B}]} \mu(\mathbf{b}') = \frac{1}{1-\gamma} \tag{2.4e}$$

where $\mu_a(\mathbf{b}) = \sum_{\mathbf{x} \sim [\mathbf{b}]} \phi_a(\mathbf{x}), \forall \mathbf{b} \in \mathrm{Dom}[\mathbf{B}]$ is the *marginal visitation frequency* for a subset of state variables $\mathbf{B} \subset \mathbf{X}$ ($\mathbf{b} \in \mathrm{Dom}[\mathbf{B}]$ represents enumeration of the variables in $\mathbf{B}$ and $\mathbf{x} \sim [\mathbf{b}]$ are the assignments of $\mathbf{x}$ that are consistent with $\mathbf{b}$), and $\mu(\mathbf{b}) = \sum_a \mu_a(\mathbf{b})$. The constraints ensure that these μ_a variables are consistent across variable subsets. The factored dual approximation is guaranteed to be equivalent to the dual LP-based approximation [46] if the factored MDP cluster set $\mathscr{B}$ forms a junction tree. Triangulation $\mathbf{Tr}(\mathscr{B}_{FMDP})$ constructs a junction tree by adding cluster sets to $\mathscr{B}$ if needed. Approximate triangulation $\hat{\mathbf{Tr}}(\mathscr{B})$ returns some cluster set $\mathscr{B}'$ such that $\mathscr{B} \subseteq \mathscr{B}'$. The constant basis function h_0—i.e., with scope as the empty set $\{\emptyset\}$—is always included in H for feasibility of the above factored LPs [46].

2.2.2.4 Representing and Computing the Optimal Policy

Policies in factored MDPs can be compactly represented assuming the default action model [47]. Different actions often have very similar transition dynamics, only differing in their effect on a small subset of variables. In factored MDPs that follow a *default transition model* for each action a, $Effects[a] \subset \mathbf{X}'$ are the variables in the next state whose local probability model is different from the model for the default action d, i.e., $P_a(X_i'|\text{Parents}_a(X_i')) \neq P_d(X_i'|\text{Parents}_d(X_i'))$ [47]. Similarly, in the *default reward model*, there is a set of reward functions for the default action d. The extra reward of any action a has scope restricted to $\mathbf{W}_i^a$. With the above assumptions, the greedy policy relative to a factored value function can be represented as a decision list [44].

However, this default action model is often not applicable in many real world examples. In such cases, we solve the approximate factored dual LP and monitor the values of the μ_a variables as a proxy to determine if a certain action appears in the computed policy. More precisely, if ϕ_a is a feasible solution to the exact dual LP, then in a state $\mathbf{x}$, $\phi_a(\mathbf{x}) > 0$ if $a = \pi(\mathbf{x})$ and $\phi_a(\mathbf{x}) = 0$ for all other actions. In the approximate solution with a subset of basis functions, all ϕ_a variables may not be represented by the set of μ_a variables. However, we can approximately determine the set of actions in a policy by removing those actions a from the set of allowed actions, for which $\mu_a(\mathbf{b}) = 0\ \forall \mathbf{b} \in \text{Dom}[\mathbf{B}], \forall \mathbf{B} \in \mathscr{B}$.

2.3 Reinforcement Learning and Value Function Approximation

Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward. [40, 38, 48, 49]. The problems of interest in reinforcement learning have also been studied in the theory of optimal control. In machine learning, the environment is typically formulated as a Markov Decision Process (MDP), as many reinforcement learning algorithms for this context utilize dynamic programming techniques. The main difference between the classical dynamic programming methods and reinforcement learning algorithms is that the latter do not assume knowledge of an exact mathematical model of the MDP and they target large MDPs where exact methods become infeasible [50]. Reinforcement learning differs from standard supervised learning in that correct input / output pairs need not be presented, and sub-optimal actions need not be explicitly corrected. Instead the focus is on performance, which involves finding a balance between exploration (of uncharted territory) and exploitation (of current knowledge).

We now introduce some notations albeit, very briefly. At each time step t, the RL agent lands a state $\mathbf{x}^t$ in the state space and selects an action a_t from the action space, following a policy $\pi(a^t|\mathbf{x}^t)$. It receives a scalar reward r^t, and then transitions to the next state $\mathbf{x}^{t+1}$, according to the reward function $R(\mathbf{x},a)$ and the state transition probability $P(\mathbf{x}_{t+1}|\mathbf{x}^t,a^t)$ respectively. In an episodic problem, this process continues until the agent reaches a terminal state. The cumulative discounted reward with discount factor $\gamma \in (0,1]$ is given by $R^t = \sum_{k=0}^{\infty} \gamma^k r^{t+k}$. The agent's goal is to maximize the expected discounted reward.

2.3.1 Temporal Difference Learning

Temporal difference learning is a combination of Monte Carlo ideas and dynamic programming ideas, the most prominent being SARSA and Q-learning [51, 40]. The value function $V(\mathbf{x})$ is learned directly from experience in a model-free, online, and incremental manner. The update rule is given by $V(\mathbf{x}) = V(\mathbf{x}) + \alpha[r^t + \gamma V(\mathbf{x}^{‘}) - V(\mathbf{x})]$, where α is the learning rate.

SARSA, represents state, action, reward, (next) state and (next) action. It is an on-policy control method to find the optimal policy. Q-learning is an off-policy control method to find the optimal policy. The action-value Q function is directly approximated using the Bellman equation as an iterative update $Q'(\mathbf{x}^t,a) = \mathbb{E}[r^t + \gamma \max_{a'} Q(\mathbf{x}^{t+1},a')|\mathbf{x},a]$. To incorporate generalization, linear and non-linear function approximation are commonly used [40]. Combination of off-policy learning, function approximation, and bootstrapping may result in instability and divergence [52].

2.3.2 Deep Reinforcement Learning

Recently, reinforcement learning with deep neural networks algorithms has gained much popularity. The conceptual idea in these algorithms is to store the agent's data in memory and sample random batches for learning, denoted as *experience replay*. This core technique effectively alleviates the problems associated with correlated data, non-stationary distributions, and convergence issues in learning with neural networks [53, 54, 55, 56].

2.4 Plan Interdiction

Now that we have presented the concepts in computational game theory and artificial intelligence planning, we are in a position to introduce plan interdiction. We begin with an informal sketch of the core conceptual idea. At first glance, interdiction appears to be an adversarial activity. For example, an attacker may interdict the power flow on an electric power grid, resulting in widespread blackouts [57], or interdict a transportation or a supply network [58, 59]. However, in cybersecurity, interdiction manifests with good intentions since it is the defender that interdicts an attack plan. Cyber attacks can be considered as plans, or sequential decision processes by the attackers. Specifically, a plan is a sequence of actions that begins with an initial state and upon complete execution, achieves a goal. In plan interdiction, the attacker is such a planner that constructs an optimal plan to achieve its objectives. The initial state for the attacker includes the initial vulnerabilities of the target network, as well as any relevant capabilities and information. An attack is a series of actions to accomplish a malicious goal or multiple goals with different values to the attacker. The defender is the interdictor and attempts to prevent the attacker from achieving its objectives. In most of the work in cyberdefense, the defender is myopic and attempts to only interdict any past attacks. However, a proactive defender should ideally also take into account the alternative attack plans in making a decision on the optimal interdiction.

Given this overview of plan interdiction, we illustrate the concept with a specific example of an attack plan. The initial state includes the initial attacker capabilities, such as having a boot disk and port scanning utilities. Actions include physical actions (breaking and entering and booting a machine from disk) as well as cyber actions, such as performing a port scan to find vulnerabilities. Figure 2.1 shows an attack graph (with attack actions as nodes) for this situation (the actual attack plan is highlighted in red).

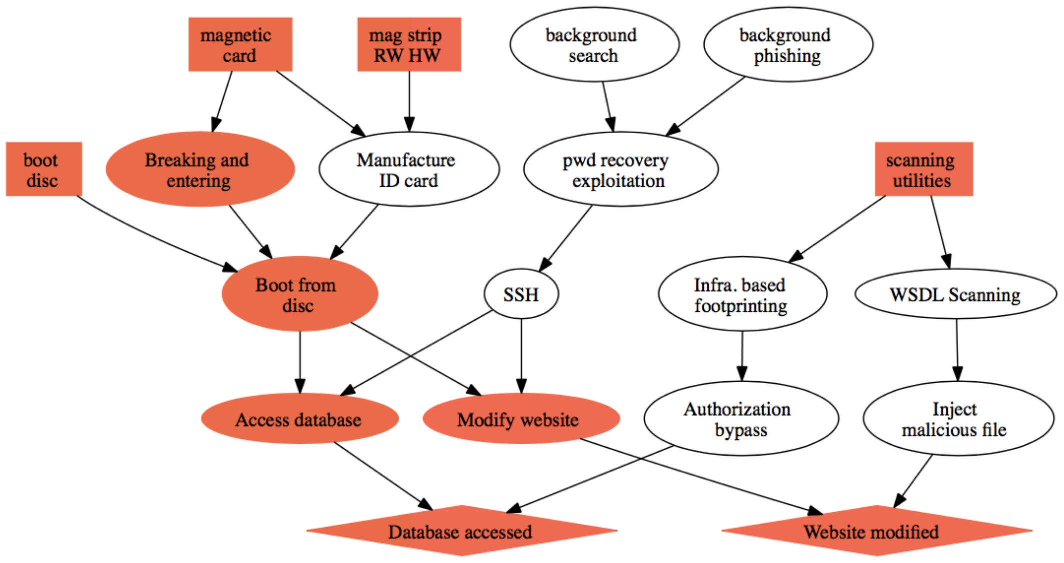


Figure 2.1: Example of an attack graph. The boxes correspond to initial attacker capabilities, ovals are attack actions, and diamonds are attack goals. An optimal attack plan is highlighted in red.

An example of interdiction is shown in Figure 2.2 in which a subset of attack actions are interdicted (for example, by patching specific vulnerabilities, or changing the network architecture). The attacker is now constrained to follow an alternative sequence of actions to achieve a suboptimal goal (in this example, it is expensive to interdict every possibility compared to any loss from a successfully executed attack).

There is a long chain of previous research that comprises graph, planning, and game theoretic approaches to cybersecurity and attacker modeling. We outline some of the most relevant previous research. The first involves graph and planning-based approaches to at-

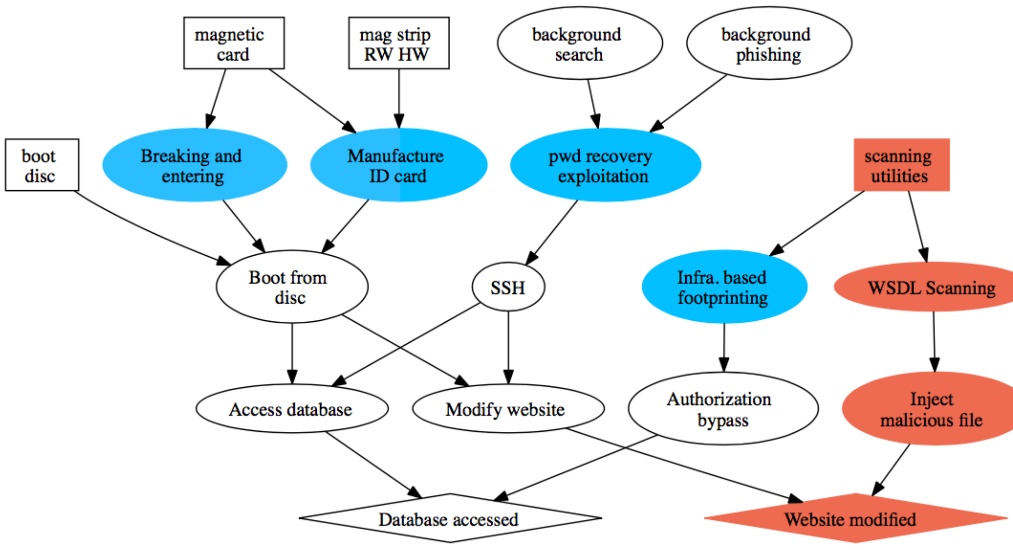


Figure 2.2: Example of attack plan interdiction: blocked actions are colored blue and the resulting alternative attack plan is highlighted in red.

tacker modeling in the form of attack trees or graphs. The main limitation of the attack graph representation is that the state space, and, thus, the graph grows exponentially in the number of state variables. Although it is common practice to construct attack graphs by hand (by experts), there are a number of efforts to automate their construction [60, 61]. To address the scalability limitations of constructing complete attack graphs, a line of research focuses only on generating specific sequences of attack steps (e.g., attack plans) that achieve a desired goal [62, 63, 64, 65, 66]. In most of the literature on attack graph analysis, mitigation is only considered as a secondary problem. Some discuss heuristic approaches, such as computing a minimum set of attacks that must be prevented to ensure that the attacker does not achieve its goals [61, 64]. Also, except for the important exception [67], uncertainty has typically not been considered in such research.

There are a number of explicit game-theoretic approaches to attacker-defender interactions in the attack graph framework [68, 69]. However, these approaches require a full specification of the attack graph, and do not scale beyond very small instances. There are several lines of research in game theoretic security games e.g., the literature on network interdiction [58, 59, 70, 57], in which the attacker typically plays the role of the interdictor (or, the defender may interdict drug traffic or border penetration). In most cases, the problem is formulated as a zero-sum game, which is then cast as a bi-level mathematical program at the core of which is some variant of a network flow problem. One important exception relevant to plan interdiction is Brown et al. [71], which offers a bi-level programming formulation for a nuclear weapons project interdiction. However, this approach is focused narrowly on maximally extending the project length, and assumes that task dependencies are given. In the plan interdiction model, attack goals are much more generic, the game admits arbitrary payoffs for the defender and attacker, and task dependencies are computed dynamically as a part of the attackers planning problem. Finally, plan interdiction is connected to the literature on computing Strong Stackelberg equilibria in leader-follower security games [30, 72]. Plan interdicton can be viewed as an approach to model circumvention games [72] where the attacker is a planning agent. This allows us to reason explicitly about attacker circumvention space, rather than abstracting it into a single circumvention action. Finally, we very briefly outline the literature relevant to planning in adversarial environments in which it is the defender that constructs a plan in the presence of an adversary. In these planning problems, the planner first commits to a plan, and the adversary subsequently attempts to interdict this plan to prevent the planner from achieving its goals. This structure arises in cyber-warfare missions, or battle planning, adversarial patrolling [73, 74, 75], intrusion detection etc. There is extensive research in the context of large zero-sum games, such as Chess and Go [76, 77], and recent literature explores Stochastic games [78].

In the next section, we formally present the plan interdiction problem.

2.4.1 The Plan Interdiction Problem

The goal in plan interdiction is to compute an optimal defender strategy to interdict the attacker's plan. In plan interdiction games, this interaction is modeled as a Stackelberg game in which the defender moves first and deploys a set of mitigations. The attacker responds by executing an optimal attack plan in the resulting mitigated environment. Formally, the *plan interdiction problem* is defined by $\{\mathscr{M}, C_m, R^D, \mathscr{P}\}$, where $\mathscr{M}$ is the set of defender's mitigations, C_m is the cost of a mitigation $M \in \mathscr{M}$, $R^D(\mathbf{x}) = \sum_j R_j(\mathbf{x}_j)$ is the defenders reward function, additive over state variables, and $\mathscr{P}$ the attacker's planning problem. A mitigation M can have the following two impacts: a) it can modify the current (initial) state x_0, and b) it can remove a subset of the attacker's actions. Consequently, if M is a subset of mitigations deployed by the defender, the attacker's planning problem $\mathscr{P}$ is modified into $\mathscr{P}(M)$. Let the resulting optimal plan for the attacker be denoted as $\pi(M)$. In the plan interdiction problem, the defender's goal is to choose the optimal set of mitigations M, taking into account the defenders utility (total discounted reward) and the cost of mitigations. Let the defender's utility be denoted as $V^D(M, \pi(M))$ and the cost of mitigations is $c(M) = \sum_{m \in M} C_m$. The defender's goal is then given by the following:

$$\max_{M \subseteq \mathscr{M}} V^D(M, \pi(M)) - c(M), \tag{2.5}$$

where $\pi(M)$ is the attackers best response, i.e., the attacker's optimal plan in the restricted planning environment with the mitigations deployed $\mathscr{P}(M)$.

2.4.2 Interdiction of Deterministic Plans

In the context of deterministic (PDDL-based) planning, let G be the set of goal literals, and R_l be the reward to the attacker for achieving a goal literal $l \in G$. Let the corresponding reward for the defender be R_l^D (for example, it can be the negative of the attacker's rewards).

The plan interdiction problem is then given by the following:

$$\max_{M \in \mathscr{M}} \sum_{l \in G} s_l(\pi(M)) R_l^D - c(M), \tag{2.6}$$

where s_l is a binary indicator of whether the goal literal l is achieved by the attacker and $c(M) = \sum_{m \in M} C_m$ is the cost of mitigations. This bi-level optimization problem can be solved using a combination of mixed-integer linear programming and constraint generation as demonstrated in Letchford and Vorobeychik et al. [19]. The constraints represent possible attack plans, and are iteratively added in constraint generation by computing approximately optimal plans using state of the art heuristic planning software, such as SGPlan [42].

2.5 Computational Protein Design and Drug Design

In this section, we transition to the domain of immunology and antibody design. Additionally, we outline the previous literature that is relevant to our research.

Vaccination is the administration of antigenic material (a vaccine) to stimulate an individual's immune system to develop adaptive immunity to a pathogen. Vaccination is historically one of the most important medical interventions for the prevention of infectious disease. Previously, vaccines were typically made of rather crude mixtures of inactivated or attenuated causative agents. However, over the last 10 to 20 years, several important technological and computational advances have enabled major progress in the discovery and design of highly effective vaccines. Three key breakthrough approaches have potentiated structural and computational vaccine design. Firstly, genomic sciences have enabled the rapid computational identification of potential vaccine antigens. Secondly, major advances in structural biology, experimental epitope (part of the antigen that is recognized by the immune system) mapping, and computational epitope prediction have yielded major insights into the determinants defining effectiveness of vaccines, enabling their optimization.

Finally and most recently, computational approaches have been used to convert this wealth of structural and immunological information into the design of improved vaccines. The main objective of a rational vaccine strategy is to design novel vaccines that are capable of inducing long-term protective immunity. Computational tools have played increasingly important roles in rational vaccine design in recent years. Some computational techniques developed for protein structure prediction and antibody analysis are discussed in [79].

2.5.1 Antibody Design

An antibody is a large, Y-shaped protein that is used by the immune system to identify and neutralize pathogens such as bacteria and viruses. The antibody recognizes a unique molecule of the harmful agent, called an antigen, via its variable region. Each tip of the ”Y” of an antibody contains a paratope (analogous to a lock) that is specific for one particular epitope (similarly analogous to a key) on an antigen, allowing these two structures to bind together with precision. Using this binding mechanism, an antibody can tag a microbe or an infected cell for attack by other parts of the immune system, or can neutralize its target directly (for example, by blocking a part of a microbe that is essential for its invasion and survival).

It is not surprising that over the years, there have been extensive efforts geared toward designing antibodies and libraries thereof. A number of experimental techniques have been developed and successfully applied to design antibodies that bind to desired antigens or to improve the binding characteristics of an existing antibody. Computational design has been used successfully by protein engineers for many years to alter the physicochemical properties of proteins [80, 81]. In the simplest case, protein design involves optimizing the amino acid sequence of a protein to accommodate a desired 3-D conformation. This approach has been extended to related tasks such as protein-protein interface design, de novo design of protein binding molecules, design of self-assembling protein nano-cages, etc. [82, 83, 84, 85]. Our growing understanding of sequence and structure relationships in antibodies, and advances in computational protein modeling has enabled progress towards computational methods that can assist in re-designing antibodies for higher binding affinity or other desired modifications [86, 87, 88]. Further, efficient experimental platforms exist to test predicted antibody structure or designed antibody function, thereby leading to an iterative feedback loop between computation and experiment. Recently, an important proof-of-principle experiment for computer-aided epitope-focused vaccine design was reported [89]. In Sevy et al. [90], the authors discuss computational prediction of antibody structure and design of function. Although much focus has been directed at engineering antibodies with desired properties, recent work has targeted the opposite side of the problem: engineering an antigen that can elicit a desired antibody in an effective and reproducible manner. This comes with the ultimate goal of the rational design of antigens to be used in vaccination that can elicit the antibodies.

We note that substantial work in antibody design is towards predicting structure. Antibodies are basically protein sequences. Protein sequences have four different levels of structure. The primary structure is a sequence of a chain of amino acids. In the secondary structure, the amino acids are linked by hydrogen bonds. In the tertiary strcture, attractions are present between alpha helices and pleated sheets. Quarternary structure consists of more than one amino acid chain. We focus on primary sequence representation in our work. However, determining the amino acids in the primary sequence is only the first step of antibody design. Comparative modeling of protein, refers to constructing an atomic-resolution model of the *target* protein from its amino acid sequence and an experimental three-dimensional structure of a related homologous protein (the *template*). It is also worthwhile to briefly mention *dead-end elimination* algorithms [91] that have been developed and applied mainly to the problems of predicting and designing the structures of proteins. In general, the dead-end elimination algorithm minimizes a function over a discrete set of independent variables. It identifies 'dead ends', i.e., combinations of variables that are not necessary to define a global minimum, such that the combination can be replaced by a better or equally good combination. These criteria are applied repeatedly until convergence such that no more eliminations can be performed. This algorithm is guaranteed to find an optimal (global) solution and significantly outperforms several alternatives derived from genetic algorithms and Monte Carlo methods, although it can be relatively slow to converge.

2.5.1.1 Multi-Specificity Design

Advancements in computational protein design have enabled the design of better antibodies. The above protein design approaches involve the straightforward application of design methodologies to a single, static protein conformation. However, there is a need to extend protein design to apply to several conformations simultaneously. These approaches, referred to as multistate design (MSD), can be used to modulate protein specificity, model protein flexibility, and engineer proteins to undergo conformational changes [92, 93, 94, 95, 96, 97, 98]. Multistate design [99] considers the impact that a sequence has on multiple structures (states) simultaneously to rule one sequence more favorable (fit) for a particular purpose than another sequence. This can be explained as follows. Some protein design tasks cannot be modeled by the traditional single state design strategy of finding a sequence that is optimal for a single fixed backbone. Such cases require multistate design, where a single sequence is threaded onto multiple backbones (states) and evaluated for its strengths and weaknesses on each backbone. For example, to design a protein that can switch between two specific conformations, it is necessary to find a sequence that is compatible with both backbone conformations. Multistate design is the most appropriate approach in such cases.

While existing experimental and computational antibody design methods have made key contributions, there is still a need for a general computational method that can rapidly design libraries of antibodies to bind to rapidly mutating virus sequences. Several methods have been developed to enable computationally expensive multistate design [21, 99]. However, these methods all suffer from large energetic barriers that limit sampling in sequence space, resulting in sub-optimal designs [21]. Recently, algorithms have been proposed for multi-specificity design, which extend general protein design by creating a sequence that has low energy with multiple binding partners [21]. Our research aims to address these significant limitations in sampling by developing global solution approaches for broadly binding antibody (and equivalently, protein sequence) design.

2.5.2 Game-Theoretic Models of Vaccination Decisions

Our work bears superficial similarity to game theoretic models of vaccination decisions [100, 101, 102]. 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. [103] 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 (Archetti et al. [104]), 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.

2.5.3 Combinatorial Drug Design

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 [105]. 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

[106]. 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 [107]. Richter et al. [108] 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.

In Ferugson et al. [109], computational models have been used to develop an approach to translate available viral sequence data into quantitative landscapes of viral fitness as a function of the amino acid sequences of its constituent proteins. Predictions emerging from these computationally defined landscapes for the proteins of HIV are positively tested against new in vitro fitness measurements and were consistent with previously defined in vitro measurements and clinical observations. These landscapes chart the peaks and valleys of viral fitness as protein sequences change and inform the design of vaccines and therapies that can target regions of the virus most vulnerable to selection pressure.

2.5.4 Learning Protein-Protein Interactions from Data

A core technical challenge in our research is to learn a prediction model of binding and stability energy scores, from data generated with ROSETTA. In studying the strength and specificity of interaction between members of two protein families, key questions center on which pairs of possible partners actually interact, how well they interact, and why they interact while others do not. The advent of large-scale experimental studies of interactions between members of a target family and a diverse set of possible interaction partners offers the opportunity to address these questions. Sequence-based graphical models of protein families have been used to capture amino acid interactions in order to predict protein structure [110, 111, 112] and function [113, 114] and to design new proteins [115, 116]. Sequence-based models of interacting protein families for binary prediction of interaction have been discussed in Thomas et al. [117]. The focus is on co-evolution of residues in interacting protein families. The cross-coupled residues are first identified by metrics like correlation and mutual information between amino acid types at a pair of positions. These can be used as predictors of interaction on the assumption that correlated mutations occur in case of interacting proteins. A number of approaches have considered the identification of binding sites given neutralization information, using sequence and/or structure information [118, 119, 120]. In Wang et al. [121], a computational method based on probabilistic relational model attempts to address this task using high-throughput protein interaction data and a set of short sequence motifs. The model is learned using the expectation-maximization (EM) algorithm, with a branch-and-bound algorithm as an approximate inference for the E-step. Feldmann et al. [120] use kernel SVMs with string kernels to infer interactions between amino acids and predict neutralization. The bi-linear energy score prediction model in our research is inspired from Kamisetty et al. [122], in which a data-driven graphical model is used to explicitly represent the pairwise amino acid interactions.

Part I

Plan Interdiction in Markov Decision Processes (MDPs)

Chapter 3

MDP INTERDICTION

Stackelberg game approaches to security have received considerable attention in recent years, both in theoretical investigation and practical use [123, 124, 30]. Examples include physical security (such as air marshall assignment and coast guard patrols), sustainability (such as security measures to prevent poaching), and computer network security. Commonly in such models the defender assigns security resources to targets, and the attacker chooses which target(s) to attack. However, in many settings the attacker performs a series of activities to accomplish their malicious goal. For example, in physical security these may involve reconnaissance, the choice of equipment to bring, and the path to take to targets which is taken. A general way to capture such multi-stage attacks is planning [65, 66, 64, 125, 126]. The defensive actions in such a scenario can then be viewed as *plan interdiction* [19], whereby a defender deploys protective measures to compromise the ability of the attacker to successfully execute its plan. A major challenge in such approaches is high-resolution modeling of adversarial evasion of defensive measures. Letchford and Vorobeychik proposed modeling such evasion in a Stackelberg framework of plan interdiction [19], where the Stackelberg leader eliminates a subset of attack actions, and the adversary computes an optimal plan in the restricted action space. This approach was developed both in the context of deterministic (PDDL-based) planning, and planning with markov decision processes (MDPs).

As discussed in the related work section, interdiction of deterministic plans has the advantage that the problems are highly structured and therefore, scalable approaches can be developed. However, there is a major limitation as deterministic planning fails to capture uncertainty in the attacker's planning problem. To address this, the previous research in Letchford and Vorobeychik et al. [19] models the attacker's planning problem as an MDP.

In this chapter, we formally introduce the MDP interdiction problem in which the attacker solves a discounted reward MDP.

We emphasize that the discounted reward MDP is a natural model for security, since it captures the fact that attackers prefer to achieve their goals (positive rewards) earlier, and incur costs (negative rewards) later. Additionally, it captures an element of deterrence: the attack which takes too many steps has far more opportunities to fail in practice. Thus, minor (low-reward) goals may be preferred over major (high-reward) goals if they can be achieved much more quickly.

3.1 The MDP Interdiction Problem

3.1.1 Problem Definition

We model MDP interdiction as a Stackelberg (two-stage, one-shot) game with two players: defender and attacker. The defender, who is the Stackelberg leader, commits to a set of mitigations, and the attacker, who is the follower, computes an MDP policy which optimally responds to (e.g., evades) these mitigations.

Formally, the *MDP interdiction problem (MDPI)* is defined by a tuple $\{\mathscr{M}, C_m, R^D, R^A, \mathbf{X}, A, P, \gamma\}$, where $\mathscr{M}$ is the set of mitigation strategies available to the defender, R^D and R^A are the reward functions for the defender and the attacker respectively, C_m is the cost of a mitigation $m \in \mathscr{M}$ to the defender, and $\mathbf{X}, A, P, \gamma$ are the state space, action space, transition function, and discount factor of an infinite-horizon discounted MDPs which the attacker is solving in response to mitigations deployed by the defender. The semantics of a mitigation $m \in \mathscr{M}$ is that it removes (protects against) a subset of attack actions from the original attacker action space A.[1] For a given set of mitigations $M \subseteq \mathscr{M}$ deployed by the defender, we can define an attacker's resulting MDP, $\tau(M) = [\mathbf{X}, A(M), R^A, P, \gamma]$ over the restricted action space $A(M)$ which includes only the actions which are not removed by any

[1]This is quite general; for example, we can model mitigations which modify the initial state by including actions with no preconditions and effects which represent initial state, and allow interdiction of these actions.

mitigation $m \in M$. In the MDPI Stackelberg game, the defender first chooses $M \subseteq \mathscr{M}$, and the attacker subsequently chooses a policy π in the resulting restricted MDP $\tau(M)$. Since the attacker is effectively facing a decision problem, it will suffice to restrict attention to optimal attacker policies which are deterministic and stationary. Let $\Pi^*(M)$ be the set of optimal deterministic stationary policies of $\tau(M)$. Define $\mathscr{V}^A(\mathbf{x}, \pi)$ to be the attacker's value function for a policy π starting at state $\mathbf{x}$ in MDP $\tau(M)$, and let $\mathscr{V}^D(\mathbf{x}, \pi)$ be the defender's value function (i.e., using the defender's reward function R^D). Let $\mathbf{x}_0$ be the initial state of the MDP. We seek a *strong Stackelberg equilibrium (SSE)* of MDPI, in which the defender solves

$$\max_{M \subseteq \mathscr{M}} \mathscr{V}^D(\mathbf{x}_0, \pi^*(M)) - \sum_{m \in M} C_m,$$

where $\pi^*(M) \in \arg\max_{\pi \in \Pi^*(M)} \mathscr{V}^A(\mathbf{x}_0, \pi)$, and the attacker breaks ties in the defender's favor.

3.1.2 General Approach

Letchford and Vorobeychik proposed a general approach for MDP interdiction [19] based on a mixed-integer linear programming (MILP). If we define variables D_m which are 1 iff the defender chooses a mitigation m, the defender's objective becomes the following: $\max \sum_{\mathbf{x}} \sum_{a} \phi_a(\mathbf{x}) R^D(\mathbf{x}, a) - \sum_{m \in M} D_m C_m$, where $\phi_a(\mathbf{x})$ are the dual variables of the MDP linear program as before. The attacker's objective is then given by $\max \sum_{\mathbf{x}} \sum_{a} \phi_a(\mathbf{x}) R^A(\mathbf{x}, a)$. To account for the attacker's best response, a set of constraints was introduced for the defender so that a) the optimal attacker policy chosen by the MILP is feasible given the set of defender mitigations, and b) the attacker's utility corresponding to this computed policy is better than that of any other feasible policy.

3.1.3 Research Objectives

The central challenge with MDP interdiction is the exponential size of the state space in the number of state variables. Since the general approach relies on the exact representation of the state space, it does not scale beyond toy-size problem instances (less than 10 state variables and 2^{10} state space size in the MDP when we consider binary state variables). The problem is intractability in representation as well as in computing an optimal value function and an optimal policy. Moreover, as we have described earlier, Stackelberg plan interdiction translates to bi-level optimization problems. The defender's optimization forms the outer level and the attacker MDP solution is the inner level in the optimization. Therefore, the computational challenge is compounded when we consider the combinatorial search space of the defender. Finally, a major challenge in plan interdiction is that the defender is typically not knowledgeable about the attackers planning problem, such as the initial vulnerabilities and attackers access and capabilities. Such uncertainties can be captured using a Bayesian Stackelberg game framework. However, the resulting Bayesian Stackelberg game is infeasible for state of the art plan interdiction approaches for even small problem instances. In the following chapters, we introduce our novel interdiction models which leverage factored representation of MDPs combined with an array of algorithmic techniques to achieve improved scalability and can handle practical-size problem instances.

Chapter 4

SCALABLE FACTORED MDP INTERDICTION

4.1 Contributions

In this chapter, we aim to address the problem of MDP interdiction at scale by leveraging approximation techniques developed for factored MDPs. Scalability in factored MDPs has been achieved by two basic approaches: a) exploiting structure in the MDP transition model and reward function [127, 128, 45], and b) value function approximation [38, 45, 47, 129, 130, 131]. Factored MDPs [44] represent the complex state space using state variables and the transition model using a dynamic Bayesian network. This representation allows an exponential reduction in the representation size of structured MDPs. Moreover, efficient approximate solution algorithms have been proposed that exploit structure in factored MDPs.

Starting with the approximation methods for factored MDPs, we develop a mixed-integer linear programming approach for factored MDP interdiction. In doing so, we face two challenges: 1) effective basis representation, and 2) a super-exponential set of constraints corresponding to alternative evasion plans for the attacker. To address the first challenge, we propose using a Fourier (parity function) basis over a Boolean hypercube to represent the value function over a binary factor space. While there always exists an exact Fourier basis for functions over a Boolean space, the representation is exponential in size. We address this challenge by developing iterative basis generation methods. Addressing the second challenge of an intractably large constraint space, we develop a novel constraint generation algorithm using a combination of linear programming factored MDP solvers and novel heuristics for attack plan generation. We demonstrate the effectiveness of the proposed approaches on realistic examples from the international planning competition (IPC). In particular, we show that our approach offers dramatically improved scalability without significantly compromising solution quality.

4.2 Factored MDP Interdiction

4.2.1 A Mixed-Integer Linear Programming Formulation for Factored MDP Interdiction

We first exhibit the defender and attacker objectives using a factored representation of states. Given a set of basis functions H and the variable cluster set $\mathscr{B}_{FMDP}$ the defender's utility is given by

$$\sum_{a}\sum_{j=1}^{r}\sum_{\mathbf{w}_j^a\in\mathrm{Dom}[\mathbf{W}_j^a]}\mu_a(\mathbf{w}_j^a)R_j^{Da}(\mathbf{w}_j^a)-\sum_{m\in M}D_mC_m,$$

where the expected sum of rewards is represented by the μ_a variables, the factored version of the visitation frequencies with scope restricted to that of the local reward functions. The first term is to minimize the attacker's value of the initial state (we set $R^D=-R^A$ in our experiments) and the second term represents mitigation costs. The attacker's objective, in turn, is

$$\sum_{a}\sum_{j=1}^{r}\sum_{\mathbf{w}_j^a\in\mathrm{Dom}[\mathbf{W}_j^a]}\mu_a(\mathbf{w}_j^a)R_j^{Aa}(\mathbf{w}_j^a).$$

For each mitigation $m\in M$, let $A_{m,a}=1$ iff m removes action a. To ensure that the computed policy is feasible, we add the constraints 2.4a-2.4e in the approximate factored dual LP. Let $\delta_\pi=1$ if and only if the policy π is interdicted, i.e., there is a deployed mitigation m that removes at least one action from π. We denote the following MILP formulation for *MDPI* by *MDPI_MILP*:

$$\max \sum_a \sum_{j=1}^{r} \sum_{\mathbf{w}_j^a \in \mathrm{Dom}[\mathbf{W}_j^a]} \mu_a(\mathbf{w}_j^a) R_j^{Da}(\mathbf{w}_j^a) - \sum_m D_m C_m$$

s.t.:

$$\forall a, m, D_m A_{m,a} \leq D_a \leq \sum_{m'} D_{m'} A_{m',a} \tag{4.1a}$$

$$\forall a, \forall \mathbf{B} \in \mathscr{B}, \forall \mathbf{b} \in \mathrm{Dom}[\mathbf{B}],$$

$$\mu_a(\mathbf{b}) \leq Z(1 - D_a) \tag{4.1b}$$

$$\forall \pi, a \in \pi, D_a \leq \delta_\pi \leq \sum_{a' \in \pi} D_{a'} \tag{4.1c}$$

$$\forall \pi, \sum_a \sum_{j=1}^{r} \sum_{\mathbf{w}_j^a \in \mathrm{Dom}[\mathbf{W}_j^a]} \mu_a(\mathbf{w}_j^a) R_j^{Aa}(\mathbf{w}_j^a)$$

$$\geq \mathscr{V}^A(\mathbf{x}_0, \pi) - Z\delta_\pi \tag{4.1d}$$

constraints $2.4a - 2.4e$

where Z is a large number and $\mathscr{B} = \mathbf{Tr}(\mathscr{B}_{FMDP})$. (We use $\hat{\mathbf{Tr}}(\mathscr{B}) = \mathscr{B}$ so that $\mathscr{B} = \mathscr{B}_{FMDP}$). The constraints 4.1a compute a variable D_a such that $D_a = 1$ iff there is a mitigation m that interdicts action a. Constraint 4.1b ensures that if an action is interdicted, the corresponding visitation frequencies are 0. Constraints 4.1c compute δ_π. Constraints 4.1d represents the condition that the policy generated for the attacker is its best response to the defender's choce of mitigations.

If H, the set of basis functions considered is general enough to include the full value function space, the solution to this MILP yields the optimal interdiction decision for the defender. The key challenge, however, is (a) what basis function space we should consider, and (b) given that capturing arbitrary value functions in the basis space is likely intractable, how can we best approximate a value function basis in this space. Finally, the set of constraints captures all possible attack policies, thereby rendering the MILP too large to be tractable even with a compact set of bases. We address these challenges next, starting with the issue of iteratively generating constraints to avoid complete enumeration of the policy space (Section 4.3), and proceeding to address the basis selection problem thereafter (Section 4.4).

4.3 Constraint Generation for Factored MDP Interdiction

The MDP interdiction algorithm requires the addition of policies and the corresponding utilities as constraints (captured by Constraint 4.1d). To compute the attacker's best response, we solve the approximate primal LP 2.3 for a given basis set H (we deal with the basis selection problem in Section 4.4). We can then compute the attacker's policy as discussed in Section 2.2.2.

4.3.1 Constraint Generation with Basis Function Selection

We define the *master* problem *MDPI_MASTER($\hat{\mathscr{P}}$)* as a relaxed version of the MILP with the constraints 4.1a-4.1d corresponding to a subset $\hat{\mathscr{P}}$ of all possible policies. For now, suppose that we have a method for selecting a subset of "important" basis functions (Section 4.4).

The constraint generation procedure (Algorithm 1) works as follows. In any iteration, $\hat{\mathscr{P}}$ contains a small set of attack policies generated thus far. We solve the master problem with $\hat{\mathscr{P}}$ to obtain a set of mitigations $\hat{M} \subseteq \mathscr{M}$, along with a policy $\hat{\pi} \in \hat{\mathscr{P}}$ with a utility of $\hat{V} = \mathscr{V}^A(\mathbf{x}_0, \hat{\pi})$ which is the attacker's best decision from the feasible subset of policies in $\hat{\mathscr{P}}$. Now there are two possibilities: either $\hat{\pi}$ is the actual best response of the attacker, in the presence of the deployed mitigations, or the actual attacker best response is not in $\hat{\mathscr{P}}$. To confirm, we can compute the best response for the attacker by solving a factored MDP (LP 2.3), removing actions which are blocked by the mitigations $\hat{M}$. At this point, we also improve our basis function set, as described in Section 4.4 (GENERATEBASIS($A_{\hat{M}}$,H)). The resulting solution will either have a utility of $\hat{V}$ to the attacker, confirming $\hat{M}$ as the optimal set of mitigations, or will be a strict improvement on $\hat{V}$, in which case we add the resulting policy (computed as described in Section 2.2.2), and its utility, to the master program, and repeat.

Algorithm 1 Constraint Generation with Basis Selection

```
function ConstraintGeneration($\hat{\mathscr{P}}$,$H$)
    $V = \infty$
    $\hat{V} = 0$
    while $\hat{V} < V$ do
        $(\hat{M}, D_a, \hat{V})$ =MDPI_MASTER($\hat{\mathscr{P}}$,$H$)
        $A_{\hat{M}} = \emptyset$
        for $a \in A$ do
            if $D_a = 0$ then
                $A_{\hat{M}} = A_{\hat{M}} \cup a$
        $(\pi, V, \hat{H})$ = GenerateBasis($A_{\hat{M}}$,$H$)
        if $V > \hat{V}$ then
            $\hat{\mathscr{P}} = \hat{\mathscr{P}} \cup \pi$
            $H = \hat{H}$
```

This constraint generation procedure suffers from two important bottlenecks: the number of iterations can be large, and each iteration can be computationally costly. Next we improve upon the baseline procedure above by alleviating both of these issues.

4.3.1.1 Reducing the Number of Iterations of Constraint Generation

A natural way to begin the constraint generation process is with an empty subset of attack policies $\hat{\mathscr{P}}$. However, this results in a large number of iterations of the constraint generation procedure building up enough attack policies to prevent trivial mitigation solutions, which mitigate no actions, or a single action sufficient to make all policies in $\hat{\mathscr{P}}$ infeasible. To address this, we start by selecting a subset of (possibly all) attacker actions $\hat{a} \in A$. For each $\hat{a}$, we solve the attacker's best response with the single action $\hat{a}$, using the approximate primal LP 2.3. The corresponding attack policies have only a single action. We then define the initial subset $\hat{\mathscr{P}}$ using these sets of attacker policies and the corresponding utilities, allowing us to warm start the constraint iteration procedure and saving a considerable amount of computation time in the process.

4.3.1.2 Fast Constraint Generation

While warm starting considerably reduces the number of iterations of the constraint generation procedure, each iteration still involves a costly set of computational operations even to evaluate whether new policies need to be added. However, we observe that to make progress in constraint generation, we only need to find *some* policy which yields a better utility for the attacker than the optimal policy computed in the master program.

A major part of the overhead is the size of the basis set. To speed up computation, we propose to attempt generating an improved policy (ATTACKERPOLICY(A,H)) first using only a small subset of the basis function (e.g., H_1 with each basis using a single state variable, as discussed in Section 4.4). If the attacker's utility computed in the subproblem is greater than the best response computed by the master program, we add this policy to the set of constraints. Otherwise, we fall back on the full combined basis selection and factored MDP solution approach. This approach may need more iterations to converge, but each iteration will be much faster. Algorithm 2 is a formalization of these ideas.

Algorithm 2 Fast Constraint Generation

```
function ConstraintGeneration($\hat{\mathscr{P}}, H_1$)
    $V = \infty$
    $\hat{V} = 0$
    while $\hat{V} < V$ do
        $(\hat{M}, D_a, \hat{V})$=MDPI_MASTER($\hat{\mathscr{P}}, H_1$)
        $A_{\hat{M}} = \emptyset$
        for $a \in A$ do
            if $D_a = 0$ then
                $A_{\hat{M}} = A_{\hat{M}} \cup a$
        $(\pi, V) =$ AttackerPolicy($A_{\hat{M}}, H_1$)
        if $V > \hat{V}$ then
            $\hat{\mathscr{P}} = \hat{\mathscr{P}} \cup \pi$
        else
            $(\pi, V, \hat{H}) =$ GenerateBasis($A_{\hat{M}}, H_1$)
```

4.4 Basis Generation

In this section we address the issue of selecting a basis function space for linear value function approximation and, subsequently, the incremental generation of the "important" set of basis functions H.

4.4.1 Fourier Basis Functions on Boolean Feature Space

We start by making use of the assumption that all variables are Boolean. In this case, the Fourier (parity) basis for Boolean function is a natural basis choice: every function $f : \{0,1\}^n \to \mathbb{R}$ can be uniquely represented as $f(\mathbf{x}) = \sum_{S \subseteq \{1,\ldots,n\}} \hat{f}(S) h_S(\mathbf{x})$ [132], where h_S is a parity function over the subset S of the variables:

$$h_S(\mathbf{x}) = \prod_{i \in S} (-1)^{x_i} = \begin{cases} +1, & \text{if } \sum_{i \in S} x_i \bmod 2 = 0. \\ -1, & \text{if } \sum_{i \in S} x_i \bmod 2 = 1. \end{cases} \tag{4.2}$$

While the full Fourier representation of the value function is therefore linear, and exact, it has 2^n bases. Consequently, it is crucial to intelligently select a small subset which yields a sufficiently good approximation of the value function for the purposes of computing an approximately optimal set of mitigations. We do this by an iterative basis function selection process described below.

4.4.2 Iterative Basis Function Selection

The attacker solves the approximate LP (2.3) to compute the best response to the imposed mitigations. Observe that the basis functions correspond to variables in this LP. Column generation can be used to generate only those variables which have the potential to improve the objective function. Thus, basis functions can be iteratively generated while computing the attacker's policy. However, since the variables $\mathbf{w}$ corresponding to the basis functions are unconstrained, the concept of reduced cost is not well-defined. In this case, we compute the magnitude of the constraint violation in the dual LP instead.

Recall that the non-linear constraint in the LP (2.3) $\max_{\mathbf{x}}\{R^a(\mathbf{x}) + \sum_i w_i[\gamma g_i^a(\mathbf{x}) - h_i(\mathbf{x})]\} \leq 0$ is represented as a set of linear constraints using variable elimination [44]. Instead of enumerating the entire state space, one variable is eliminated at a time. There is one set of factored LP constraints for each action a. Let X_j be the variable being eliminated. If X_j appears in any set $\mathbf{C} \cup \Gamma_a(\mathbf{C})$, ($\mathbf{C} = \text{Scope}[h]$), and/or $\mathbf{W}^a$ (the scope of any local reward function) these set of state variables are "relevant" while eliminating X_j. Denote this set of relevant variables by $\mathbf{Z}_j^a$. Only these variables are enumerated while maximizing over X_j. For each enumeration, the linear constraint is of the form $u^{max} \geq u^R + \sum_i w_i u^{\gamma g_i - h_i}$, where u^{max} is the variable introduced after elimination, u^R is the relevant factored reward term and $u^{\gamma g_i - h_i}$ represents $\gamma g_i - h_i$ for a relevant h_i. After all state variables are eliminated, the remaining elimination-introduced variables have empty scope and the final maximization constraint is added. The number of constraints in this LP grows exponentially in the induced width of the *cost network*, the undirected graph defined over the variables $X_1, \ldots, X_n$, with an edge between X_l and X_m if they appear together in $\mathbf{Z}_j^a$. Given this construction, we describe our basis function selection approach as follows.

We begin with a subset of basis functions H^0 and solve the above factored LP. It is necessary to include $h_0 = \emptyset$ in H^0 to ensure feasibility of the LP. Next, we need to determine whether a new basis function will improve the current LP objective. We consider the dual LP

$$\begin{aligned} \max_{\lambda_k \geq 0} \quad & \sum_k u_k^R \lambda_k \\ \text{s.t.:} \quad & \sum_k u_k^{\gamma g_i - h_i} \lambda_k = \alpha_i, \forall i, \end{aligned} \tag{4.3}$$

where λ_k is the dual variable corresponding to a factored linear constraint k in the primal LP, and u_k^R and $u_k^{\gamma g_i - h_i}$ are the reward function and basis function terms respectively in constraint k. If a new basis h_l is added, it generates a new column in the primal LP, and thus, a new constraint in the dual LP. If the new constraint is not satisfied given the current

λ, the objective can be improved by adding this basis. More precisely, if the new constraint is violated given the current λ, the amount of violation $\beta = |\sum_k u_k^{\gamma g_l - h_l} \lambda_k - \alpha_l|$ can be used to decide whether to include the new basis. We compute the magnitude of constraint violation for a possible new basis and choose the basis which maximizes this violation. We add this basis to the primal LP and repeat. Finally, we return the updated set of basis functions. The corresponding LP objective is the attacker's utility, given a set of actions A. We outline this procedure formally in Algorithm 3.

Algorithm 3 Iterative Basis Function Selection

```
function GENERATEBASIS($A, H$)
    $\lambda, V'$ =ATTACKERPOLICY($A, H$)
    for $s \in \{1, \ldots, s_{max}\}$ do
        while $H_s \neq \emptyset$ do
            $\beta = 0$
            for $h_l \in H_s$ and $h_l \notin H$ do
                if $|\sum_k u_k^{\gamma g_l - h_l} \lambda_k - \alpha_l| > \beta$ then
                    $\beta = |\sum_k u_k^{\gamma g_l - h_l} \lambda_k - \alpha_l|$
                    $\hat{h}_l = h_l$
            $H = H \cup \hat{h}_l$
            $(\lambda, V)$ =ATTACKERPOLICY($A, H$)
            if $|V - V'| < \theta$ then return $V', H$
            $H_s = H_s \setminus \hat{h}_l$
            $V' = V$
```

In Algorithm 3, ATTACKERPOLICY(A, H) solves the LP (2.3) and H_s is the set of parity basis functions over s state variables. To maintain smaller cost networks, we consider all bases of a particular size before moving to the next size until $s = s_{max}$, for some $s_{max} \leq n$. Within a particular size, we consider those variable clusters that are also connected in the underlying DBN of the factored MDP (i.e., one variable is the parent node of the other variable). We observe that many dual variables λ_k will be 0 so that we can restrict all computations to the set of active constraints $\{k, \lambda_k > 0\}$. Finally, using the parity basis functions allows two simplifications. First, we consider the g^a variable corresponding to a basis h: $g^a(\mathbf{y}) = \sum_{\mathbf{c} \in \mathbf{C}} \prod_{i|X_i \in \mathbf{C}} P_a(\mathbf{c}[X_i]|\mathbf{y}) h(c)$, for each assignment $\mathbf{y} \in \Gamma_a(\mathbf{C})$, where $\mathbf{C}$ is the scope of h and the sum is over Dom[$\mathbf{C}$], the enumeration of variables in $\mathbf{C}$. In our case, using the parity basis, this sum of products can be reorganized as a product of sums: $g^a(\mathbf{y}) = \prod_{i|X_i \in \mathbf{C}} P(x_i = 0|\mathbf{y},a) - P(x_i = 1|\mathbf{y},a)$. These terms can be precomputed for each state variable allowing efficient computation. Second, we consider $\alpha = \sum_{\mathbf{x}} \alpha(\mathbf{x})h(\mathbf{x}) = \sum_{\mathbf{c}\in\mathbf{C}} \alpha(\mathbf{c})h(\mathbf{c})$, where $\alpha(\mathbf{c})$ is the marginal of α over $\mathrm{Dom}[\mathbf{C}]$. In the case of parity basis functions, $\alpha_l = 0, \forall l \neq 0$ and $\beta = |\sum_k u_k^{\gamma g_l - h_l} \lambda_k|$.

4.5 Greedy Interdiction

In this section, we propose a greedy heuristic for factored MDP interdiction which requires the generation of attacker policies in response to specific mitigations. Specifically, we start with a mitigation strategy by randomly choosing an action to block. The attacker then computes a policy with utility V using the restricted set of actions. Next, we evaluate actions in the available set of actions A_{av}, at random, choosing an action to block if it decreases the sum of the attacker utility and total mitigation cost. Here we assume that each mitigation blocks exactly one action. The algorithm proceeds until no action can be found to be blocked so as to improve the defender's utility. This greedy algorithm is outlined as Algorithm 4.

We speed up greedy interdiction similar to fast constraint generation in Section 4.3.1.2 by computing policies with a small subset of basis functions (e.g., H_1). At the very end, we make further additions to the set of basis functions to compute the attacker's policy in response to the greedily computed mitigation strategy to check whether the attacker can indeed improve on the approximate best response computed over the restricted space.

4.6 Experiments

4.6.1 Problem Domains

We evaluate our MDP interdiction algorithms on several instances of three problem domains from the international planning competition (IPC 2014): a) sysadmin b) academic

Algorithm 4 Greedy Factored MDP Interdiction

```
$A_{av} = A$
$A_m = \emptyset$
$A_n = A_{av}$
$\hat{V} = \infty$
while $A_n \neq \emptyset$ do
    $a =$ CHOOSERANDOM($A_{av}$)
    $V =$ ATTACKERPOLICY($A_{av} \setminus a, H$)
    if $V^A < \hat{V}$ then
        $A_m = A_m \cup a$
        $\hat{V} = V$
        $A_{av} = A_{av} \setminus a$
        $A_n = A_{av}$
        if $A_{av} = \emptyset$ then
            break
    else
        $A_n = A_n \setminus a$
return $V =$ GENERATEBASIS($A \setminus A_m, H$)
```

advising and c) wildfire. While these have little direct connection to security, they provide the most meaningful evaluation of our approaches in terms of effectiveness and scalability: prior security-related domains which consider multi-stage attacks use toy examples which would not provide a meaningful evaluation.

For all experiments, each defender mitigation $m \in \mathscr{M}$ blocks exactly one action a. We also let $R^A(\mathbf{x}, a) = -R^D(\mathbf{x}, a) - C_a$, where C_a is the cost of action a, which we set to 0 for the default (no-op) action and to 0.5 for all other actions.. We set the cost of imposing a mitigation $C_m = 1$ for all m. We use the discount factor of $\gamma = 0.9$. The experiments are run on a 2.4GHz hyperthreaded 8-core Ubuntu Linux machine with 16 GB RAM, with CPLEX version 12.51 used to solve MILP instances.

4.6.2 Comparison with Exact MDP Interdiction

First, we compare the performance of the constraint generation with basis selection algorithm to the state-of-the-art optimal solution in MDP interdiction proposed by Letchford and Vorobeychik [19]. We consider the sysadmin domain with $n = 2 - 10$ state variables

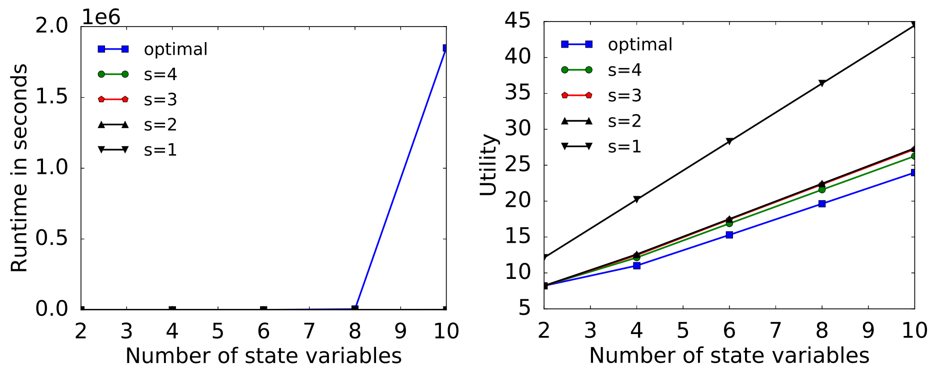


Figure 4.1: Comparison of exact and approximate MDP interdiction in terms of runtime (left) and attacker utility (right; lower is better for the defender).

(2^n states) and 10 actions. We evaluate our approach with $s = 1, 2, 3$ and 4, where s is the maximum number of state variables in the scope of any basis.

As expected, the runtime of the exact MDPI is dominated by our approach for sufficiently many state variables (Figure 4.1(a)); more significantly, the exact approach runs out of memory for larger problem sizes.

From Figure 4.1(b) we can see that while the utility of approximate interdiction improves significantly as s increases from 1 to 2, it already becomes close to optimal when $s = 2$, with little added value from increasing it further. The results are similar for other IPC domains. Consequently, our experiments below use $s = 2$.

4.6.3 Scalability

We evaluate the constraint generation approach on larger problem sizes on the sysadmin domain (up to 60 state variables and 60 actions). Even with constraint generation with only a subset of basis functions, our baseline algorithm (marked as "slow bilevel") scales poorly for $n > 30$. On the other hand, the use of fast constraint generation (Algorithm 2, marked as "fast bilevel"), significantly improves scalability (Figure 5.1 left). Indeed, the baseline (slow bilevel) becomes intractable for $n \geq 50$, whereas we can successfully solve these with the "fast" approach. Since we compute the utility of the final attacker policy using basis generation, the solution accuracy is not compromised (Figure 5.1 right).

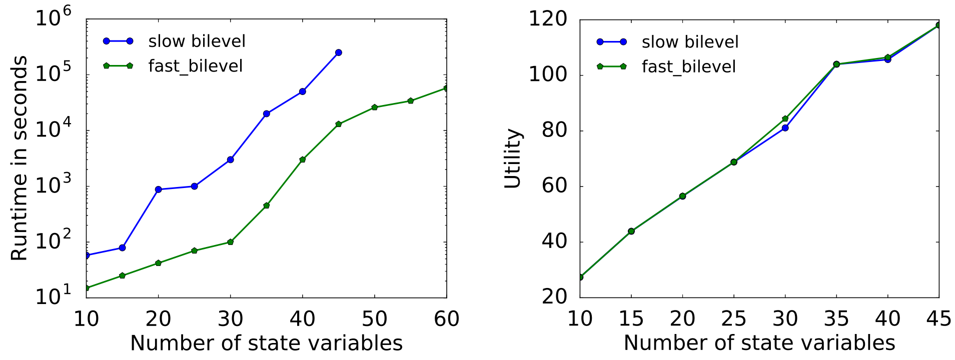


Figure 4.2: Comparison between baseline (slow) and fast interdiction on the sysadmin domain in terms of runtime (left) and utility (right).

In the second set of scalability experiments, we evaluate our approaches on 10 problem instances of the academic advising domain. The problem size increases with problem number from 10 to 30 courses (20 to 60 state variables and 10 to 30 actions). For each problem size, there are two instances, corresponding to different program requirements and course prerequisites. The first (odd numbered) problem instance is somewhat simpler (fewer prerequisites per course). The second (even numbered) instance is more complicated, with a larger number of prerequisites per course (larger number of connections in the underlying DBN). Problem 10 has the largest problem size with 30 courses, 11 program requirements, 3 prerequisites for most courses and 4 prerequisites for 8 courses. As demonstrated in Figure 5.2, we observe a similar trend as before: the fast constraint generation approach significantly outperforms baseline without compromising much solution quality. The baseline is intractable for problems 7 to 10 ($n \geq 50$).

In the third set of experiments, we evaluate on 6 problem instances of the wildfire domain. The grid size increases with problem number from $m = 3$ to 5 ($n = 2 \times m^2 = 18$ to 50 state variables, and 36 to 100 actions). For each grid size, there are two instances, corresponding to different neighbourhood configurations and targets (cells on the grid that need to be protected). The first (odd numbered) problem instance has fewer targets than the second (even numbered) instance. The results are shown in Figure 5.3. The baseline is

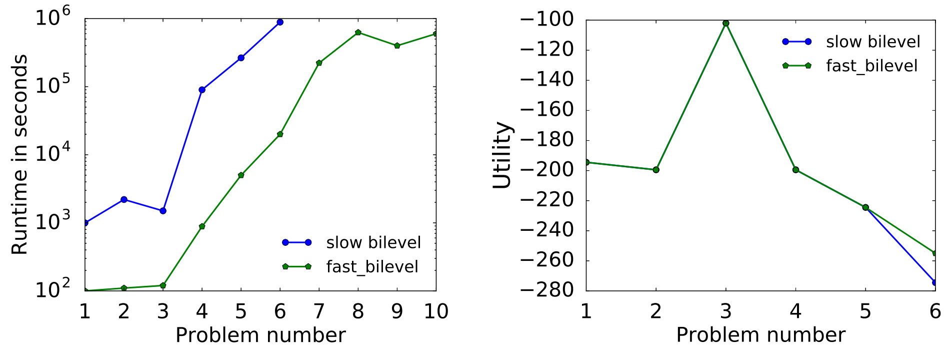


Figure 4.3: Comparison between baseline (slow) and fast interdiction on the academic advising domain in terms of runtime (left) and utility (right).

again intractable on problems 5 and 6 ($n = 50$) which can be solved by fast bilevel.

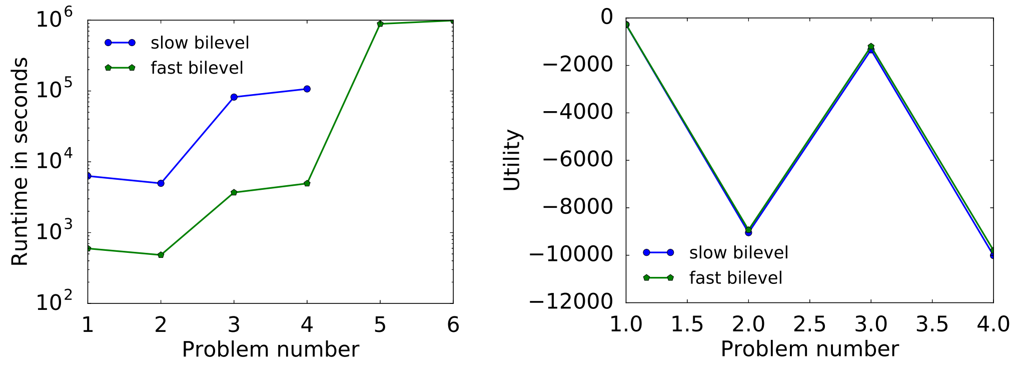


Figure 4.4: Comparison between baseline (slow) and fast interdiction on the wildfire domain in terms of runtime (left) and utility (right).

4.6.4 Effectiveness of Greedy Interdiction

Finally, we compare the greedy interdiction algorithm to fast constraint generation. As shown in Figures 4.5-4.7, the greedy algorithm is faster for larger problem sizes, saving up to an order of magnitude of computation time, without significantly compromising solution quality.

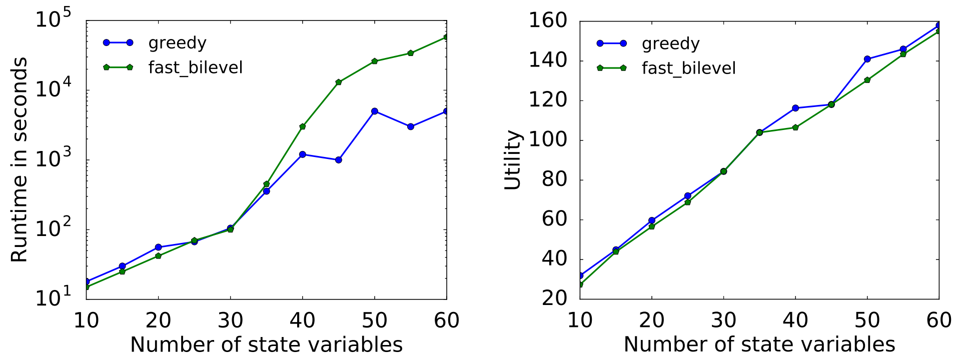


Figure 4.5: Comparison between fast interdiction and greedy in terms of runtime (left) and utility (right) on the sysadmin domain.

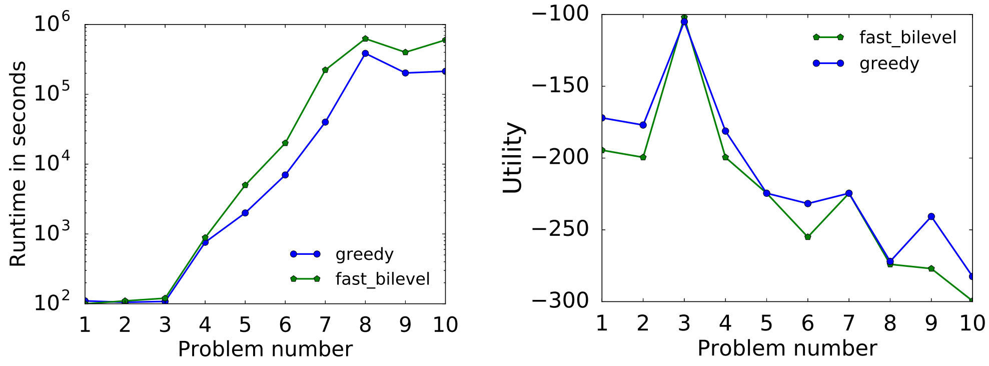


Figure 4.6: Comparison between fast interdiction and greedy in terms of runtime (left) and utility (right) on the academic advising domain.

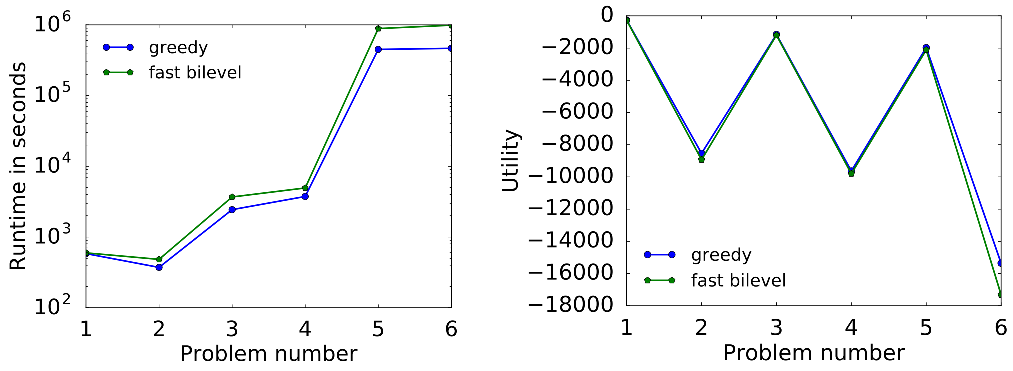


Figure 4.7: Comparison between fast interdiction and greedy in terms of runtime (left) and utility (right) on the wildfire domain.

4.7 Conclusions

We presented a MILP approach for factored MDP interdiction, using a parity basis for linear value function approximation over binary state variables. We offered an iterative basis generation approach to select the most effective set of basis functions, and presented several variations of constraint generation, combined with basis selection, to solve the MILP. We evaluated our approaches on several realistic problem instances and demonstrated significantly increased scalability while achieving near-optimal solutions. Finally, we proposed a greedy algorithm for MDP interdiction and showed that it can further improve scalability.

In this paper, we only model deterministic mitigation strategies. Related research on Stackelberg games for security often considers randomized defensive resource allocation, which in our case would translate to randomized mitigations that can yield considerably higher utility to the defender. Within our framework, such an extension is quite non-trivial, and remains an important question for future research.

Chapter 5

SCALABLE INITIAL STATE INTERDICTION IN FACTORED MDPS

5.1 Contributions

While our MDP interdiction algorithm in the previous chapter achieves superior scalability compared to prior art, it continues to face significant computational challenges and scalability limitations, particularly in capturing uncertainty about the attacker.

In this chapter, we propose a novel interdiction model in which the defender modifies the initial state of the attacker. This is quite general: for example, we can model prior interdiction approaches by adding action-specific preconditions as state variables. However, we show that this change enables significantly simpler and far more scalable interdiction techniques which rely on single-level integer linear programming, in contrast to difficult bi-level problems faced in prior art. We further improve scalability by using model-free reinforcement learning techniques with linear and non-linear action-value function approximators. In the former, we make use of a Fourier basis approximation for Boolean functions, develop methods for iteratively constructing a small subset of basis functions, and formulate an integer linear program for optimal interdiction. For the latter, we propose two iterative local search methods, the first optimizing one variable at a time, and the second using iterative linear approximations. Finally, we present a natural extension of our interdiction framework to Bayesian interdiction, in which the defender is uncertain about parameters of the attacker's planning problem.

We demonstrate the effectiveness and scalability of our proposed approaches compared to baseline alternatives on realistic examples from the international planning competition.

5.2 MDP State Interdiction

5.2.1 Problem Definition

We model MDP interdiction as a Stackelberg (two-stage, one-shot) game with two players: defender and attacker. The defender is the Stackelberg leader, choosing an optimal interdiction policy. In our model, and unlike prior work, the defender transforms a given *initial state* of the MDP, $\mathbf{x}_0$ (which represents an initial or default configuration) into a new *start state*, $\mathbf{x}'_0$ of the attacker's MDP. Note that this model is quite general. For example, we can capture removing an action by adding to it a binary precondition such that the action is not applicable when this precondition is false; removing this action is then equivalent to negating this precondition in $\mathbf{x}'_0$

The attacker is the follower, and computes an MDP policy beginning with the start state $\mathbf{x}'_0$ chosen by the defender. In many settings of interest, such as cybersecurity, the state variable domains are finite and relatively small (indeed, prior work has modeled it using deterministic planning [65]), and we can therefore represent the associated planning problems as MDPs with binary variables. Consequently, we henceforth assume that the state space is comprised of a collection of binary variables.

Formally, the *MDP state interdiction problem (MDPSI)* is defined by an MDP $\mathscr{M} = \{\mathbf{X}, A, R(\mathbf{x}, a), \gamma\}$, where $\mathbf{X}, A, R(\mathbf{x}, a), \gamma$ are the state space, action space, reward function, and the discount factor of an infinite-horizon discounted MDP which the attacker is solving, as well as the defender's reward function $R_D(\mathbf{x}, a)$ and interdiction costs $\rho(\mathbf{x}'_0, \mathbf{x}_0) = \sum_i \rho_i |x'_{i0} - x_{i0}|$ of modifying an initial state $\mathbf{x}_0$ into $\mathbf{x}'_0$, where ρ_i is the cost of modifying variable i. Note that this cost can also capture inability to modify specific state variables (the corresponding cost is then infinite).

We assume that the game is zero-sum (modulo interdiction costs), so that $R_D(\mathbf{x}, a) = -R(\mathbf{x}, a)$. Define $\mathscr{V}(\mathbf{x}_0, \pi)$ as the attacker's value function for a policy π starting at state $\mathbf{x}_0$ in MDP $\mathscr{M}$. Let Π be the set of deterministic stationary policies of the MDP. The defender then solves

$$\min_{\mathbf{x}_0' \in \mathbf{X}} \mathscr{V}(\mathbf{x}_0', \pi^*) + \rho(\mathbf{x}_0', \mathbf{x}_0),$$

where $\pi^* \in \arg\max_{\pi \in \Pi} \mathscr{V}(\mathbf{x}_0, \pi)$ is the optimal policy of $\mathscr{M}$. The key bottleneck is the exponential state space in the attacker MDP. We propose reasonable approximations for the attacker's value function and scalable algorithms to compute it using a) factored MDP solution approaches and b) reinforcement learning. The key observation is that the optimal policy is *independent* of the interdiction decision, since the solution to an MDP is a function of an arbitrary state; to put it differently, the attacker's optimal policy is *already* a function of the defender's choice of the start state, by virtue of it solving an MDP. This is a key to significantly cleaner optimization problems for MDP interdiction below compared to prior art, as well as associated scalability gains.

5.3 Integer Linear Program for Approximately Optimal Interdiction

As a first approach, we use a linear approximation of the attacker's value function, and leverage a linear programming approach for approximate planning in factored MDPs. In addition, we make use of a Fourier representation of functions over a Boolean hypercube as the basis. In particular, any function $f : \{0,1\}^m \to \mathbb{R}$ can be uniquely represented as $f(\mathbf{x}) = \sum_{j:S_j \subseteq \{1,\ldots,m\}} \hat{f}(S_j)\phi_j(\mathbf{x})$, where ϕ_j is a parity function over the subset S_j of variables [132]:

$$\phi_j(\mathbf{x}) = \prod_{i \in S_j} (-1)^{x_i} = \begin{cases} +1, & \text{if } \sum_{i \in S_j} x_i \bmod 2 = 0. \\ -1, & \text{if } \sum_{i \in S_j} x_i \bmod 2 = 1. \end{cases} \tag{5.1}$$

We define a factored value function over a set $\mathscr{B} = \{\phi_1, \ldots, \phi_{|\mathscr{B}|}\}$ of Fourier boolean basis functions: $\mathscr{V}(\mathbf{x}) = \sum_j^{|\mathscr{B}|} w_j \phi_j(\mathbf{x})$. The exact Fourier representation has 2^m bases where m is the number of state variables. However, prior work has developed an approach to select a small subset of these for a sufficiently good value function approximation [133].

Since the optimal policy, as well as the associated value function, is independent of interdiction strategy, we can pre-compute the approximate value function with an associated basis using techniques from prior art (e.g., [44, 133]). In the interdiction problem, the weights w_j in the value function are then fixed, and the goal is to optimize over the initial state $\mathbf{x}_0$ and, consequently, the associated basis function values $\phi_j(\mathbf{x}_0)$. We now develop an integer linear program (ILP) for computing the optimal choice of the initial state $\mathbf{x}_0$ given a value function $\mathscr{V}(\mathbf{x}) = \sum_j^{|\mathscr{B}|} w_j \phi_j(\mathbf{x})$ over a pre-computed Fourier basis $\mathscr{B}$.

Let $\mathbf{b}_j$ denote a binary vector of length m, indicating the variables included in basis ϕ_j. Then, for a given $\mathbf{x}_0$, $\sum\limits_i b_{ij} x_{i0} = \sum_{i \in S_j} x_{i0}$ in the Fourier representation (5.1), which is an integer. Then, for some positive integer h_j and a binary k_j, $\sum\limits_i b_{ij} x_{i0} = 2h_j + k_j$, and is odd if $k_j = 1$ and even otherwise. If this sum is even, the corresponding basis value $\phi_j = +1$, and $\phi_j = -1$ otherwise, which we can calculate by $\phi_j = 1 - 2k_j$.

We introduce binary variables c_{i1} and c_{i2} for each state variable x_i to compute the interdiction cost $\rho(\mathbf{x}'_0, \mathbf{x}_0)$. Putting everything together, we obtain the following ILP to compute the optimal interdiction:

$$\begin{aligned}
&\underset{\mathbf{x}'_0 \in \mathbf{X}, \phi, k, h, c}{\text{minimize}} \sum_{j=1}^{|\mathscr{B}|} w_j \phi_j + \sum_i \rho_i (c_{i1} + c_{i2}) \\
&\text{subject to } \sum_i b_{ij} x'_{i0} = 2h_j + k_j, \forall j && \text{(5.2a)} \\
&\phi_j = 1 - 2k_j, \forall j && \text{(5.2b)} \\
&c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i && \text{(5.2c)} \\
&x'_{i0}, k_j, c_{i1}, c_{i2} \in \{0,1\}, h_j \in \mathbb{Z}_+
\end{aligned}$$

Thus, the full interdiction approach involves two steps:

1. $[\mathscr{B}, \mathbf{w}]$ =approxMDP($\mathscr{M}$),
2. Solve ILP (5.2) given the approximate optimal value function $\mathscr{V}(\mathbf{x}) = \sum_j^{|\mathscr{B}|} w_j \phi_j(\mathbf{x})$.

The key bottleneck in this approach is Step 1, where the difficulty of solving the factored MDP grows exponentially in the number of interdependencies among state variables. An alternative approach is to use model-free reinforcement learning, which can scale significantly better when we use function approximation to represent the action-value function. However, this introduces new technical challenges, which we describe and address next.

5.4 Interdiction Using RL with Linear Action-Value Functions

In this section, we propose an alternative approach in which the attacker directly learns the action-value Q-function given states and actions, using a variation of the traditional Q-learning algorithm [40]. Just as we used a Fourier approximation of the value function earlier, we now use this basis to approximate the optimal Q-function:

$$Q(\mathbf{x}, a; \mathbf{w}) = \sum_{j=1}^{|\mathscr{B}|} w_j^a \phi_j(\mathbf{x}). \tag{5.3}$$

In order to perform interdiction, we now face two new technical challenges: first, since the Fourier basis is exponential, we need to adapt the learning algorithm to iteratively construct an effective small basis representation, and second, we need to adapt the interdiction approach to work with the Q-function, rather than the value function.

We begin with the adapted Q-learning algorithm that embeds an iterative Fourier basis construction, which proceeds as follows. The attacker starts at a random state and chooses an action a using an ε-greedy strategy. The ε parameter decays each iteration enabling more exploitation with time. The observation $\{\mathbf{x}^t, a^t, r^t, \mathbf{x}^{t+1}\}$ at each iteration is stored in a set $\mathscr{D}$. For computational speed and higher data efficiency, we use a batch gradient descent update over a randomly sampled subset $\hat{\mathscr{D}} \subset \mathscr{D}$ collected over past iterations (similar to experience replay in [53]).

5.4.1 Basis Generation

Unlike the baseline approach, addition of any new basis function is equally computationally expensive, irrespective of the interdependencies between variables in the particular basis function. We incorporate basis function selection in learning as follows. We start with an initial set $\mathscr{B} = \mathscr{B}_0$ of basis functions (e.g., all single-variable bases and the constant basis). During the learning iterations, we add a new basis function to the set $\mathscr{B}$ if it significantly reduces the squared error objective over the samples $s = (\mathbf{x}, a, r, \mathbf{x}') \in \hat{\mathscr{D}}$, $(Q'(\mathbf{x}, a; \mathbf{w}) - Q(\mathbf{x}, a; \mathbf{w}))^2$, compared to the current basis set, where $Q'(\mathbf{x}, a; \mathbf{w}) = r + \gamma \max_a Q(\mathbf{x}', a; \mathbf{w})$.

For any basis ϕ_j in the current basis set, the gradient descent weight update with learning rate η for the weight w_j^a of this basis (over a single observation) is:

$$w_j^a \leftarrow w_j^a + \eta (r + \gamma \max_a Q(\mathbf{x}', a; \mathbf{w}) - \sum_{j=1}^{|\mathscr{B}|} w_j^a \phi_j(\mathbf{x})) \phi_j(\mathbf{x}).$$

Now, consider a basis ϕ_n not present in the current approximation. In this case, the current weight $w_n^a = 0$ and the gradient update summed over the samples $s \in \hat{\mathscr{D}}$ is then

$$\mathscr{I} = \eta \sum_{s \in \hat{\mathscr{D}}} (Q'(\mathbf{x}, a; \mathbf{w}) - Q(\mathbf{x}, a; \mathbf{w})) \phi_n(\mathbf{x}).$$

Our goal is to find a basis $\phi_n(\mathbf{x})$ to add to $\mathscr{B}$ that maximizes $|\mathscr{I}|$, which measures the relative impact on the quality of the Q-function approximation. We then add this basis if $|\mathscr{I}|$ is above a predefined threshold. The following MILP computes the Boolean basis vector $\mathbf{b}$ corresponding to the new basis function with the largest marginal impact $|\mathscr{I}|$ on

Q-function approximation, given samples $s^j : (\mathbf{x}^j, a^j, r^j, \mathbf{x}^{j+1}) \in \hat{\mathscr{D}}$:

$$\begin{aligned}
&\underset{\mathbf{b},h,k,\phi,q,\delta}{\text{maximize}}\ \delta_1 + \delta_2 \\
&\text{subject to}\ \delta_1 - \delta_2 = \sum_{s_j \in \hat{\mathscr{D}}} (Q'(\mathbf{x}^j, a^j; \mathbf{w}) - Q(\mathbf{x}^j, a^j; \mathbf{w}))\phi_j && \text{(5.4a)} \\
&0 \le \delta_1 \le Lq && \text{(5.4b)} \\
&0 \le \delta_2 \le L(1-q) && \text{(5.4c)} \\
&\sum_i b_i x_i^j = 2h_j + k_j, \forall \mathbf{x}^j \in \hat{\mathscr{D}} && \text{(5.4d)} \\
&\phi_j = 1 - 2k_j, \forall \mathbf{x}^j \in \hat{\mathscr{D}} && \text{(5.4e)} \\
&\sum_i b_i \ge 1 && \text{(5.4f)} \\
&b_i, k_j, q \in \{0,1\}, h_j \in \mathbb{Z}_+, \delta_1, \delta_2 \ge 0.
\end{aligned}$$

δ_1 and δ_2 compute the linearized objective $|\mathscr{I}|$ (L is a large number). Constraints (5.4d) and (5.4e) compute the ϕ_j values as in ILP (5.2). Constraint (5.4f) ensures that the ILP does not select the constant basis function. To keep the basis set from becoming too large, we periodically monitor the weights and remove any basis functions with normalized weights below a predefined threshold.

5.4.2 Integer Linear Program for Interdiction

Given this approximation, we can now extend the interdiction LP 5.2 for the defender's optimal initial state $\mathbf{x}'_0$ to the following mixed integer program (recall that $\mathscr{V}(\mathbf{x}) = \max_a Q(\mathbf{x}, a; \mathbf{w})$):

$$
\begin{aligned}
\underset{\mathbf{x}'_0 \in \mathbf{X}, v, \phi, k, h, c}{\text{minimize}} \quad & v + \sum_i \rho_i (c_{i1} + c_{i2}) \\
\text{subject to} \quad & v \geq \sum_{j=1}^{|\mathscr{B}|} w_j^a \phi_j, \forall a && \text{(5.5a)} \\
& \sum_i b_{ij} x'_{i0} = 2h_j + k_j, \forall j && \text{(5.5b)} \\
& \phi_j = 1 - 2k_j, \forall j && \text{(5.5c)} \\
& c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i && \text{(5.5d)} \\
& x'_{i0}, k_j, c_{i1}, c_{i2} \in \{0,1\}, h_j \in \mathbb{Z}_+
\end{aligned}
$$

The full algorithm is outlined in Algorithm 5.

Algorithm 5 Interdiction using Linear Action-Value Function Learning

```
Initialize weights $\mathbf{w}$ to 0, randomly initialize state, $\varepsilon = \varepsilon_0$
**for** iterations $t$ in $1, \ldots, T$ **do**
    Select $\varepsilon$-greedy action $a^t$
    Store $s^t = (\mathbf{x}^t, a^t, r^t, \mathbf{x}^{t+1})$ in $\mathscr{D}$
    **for** each $s^j$ in a random sampled $\hat{\mathscr{D}} \subset \mathscr{D}$ **do**
        Features: $[\phi_1(\mathbf{x}^j), \ldots, \phi_{|\mathscr{B}|}(\mathbf{x}^j)]$
        Target: $r^j + \gamma \max_{a'} Q(\mathbf{x}^{j+1}, a'; \mathbf{w})$
    Gradient descent and $\mathbf{w}$ update on $\hat{\mathscr{D}}$, $\varepsilon = \varepsilon_0 e^{-t}$
    Every $\hat{t}$ iterations solve MILP (5.4), update basis set $\mathscr{B}$
Solve interdiction MILP (5.5) using basis set $\mathscr{B}$ and weights $\mathbf{w}$ **return** $\mathbf{x}'_0$
```

While this approach allows direct model-free learning and significantly more scalable interdiction (see the experiments section), the performance still relies on the subset of basis functions chosen for the approximation. We next extend the framework to allow for non-linear Q-function approximation.

5.5 Interdiction with Non-Linear Function Approximation

We now generalize the model-free interdiction framework to incorporate non-linear Q-function approximation, such as using neural network-based Q-functions. Let $Q(\mathbf{x}, a; \theta)$ denote the corresponding approximate Q-function, with parameters θ (e.g., corresponding to neural network weights).

For interdiction with a non-linear Q-function, we can no longer directly use the ILP approaches above. We instead propose two local search methods.

5.5.1 Interdiction Using Greedy Local Search

Our first approach is a greedy local search in the state space (Algorithm 6). The intuition behind the approach is to change one variable at a time, and accepting the change only if it improves the defender's utility. The process continues either for a fixed number of iterations, or until convergence.

Algorithm 6 Non-Linear Value Function Learning and Greedy Local Search

```
Start at a randomly chosen state $\mathbf{x}'_0$
Compute $U = \max_{a'} Q(\mathbf{x}'_0, a'; \theta) + \rho(\mathbf{x}'_0, \mathbf{x}_0)$
for iterations in $1, \ldots, T$ do
    Change one state variable at random to get $\bar{\mathbf{x}}$
    if $\max_{a'} Q(\bar{\mathbf{x}}, a'; \theta) + \rho(\bar{\mathbf{x}}, \mathbf{x}_0) < U$ then
        $U = \max_{a'} Q(\bar{\mathbf{x}}, a'; \theta) + \rho(\bar{\mathbf{x}}, \mathbf{x}_0)$
        $\mathbf{x}'_0 = \bar{\mathbf{x}}$
return $\mathbf{x}'_0$
```

5.5.2 Interdiction Using Local Linear Approximation

We observe that the above local search method can be slow for a large state space. To improve efficiency of state space exploration, we propose an alternative local search approach which iteratively linearizes the Q-function using a Taylor series approximation, and solves an ILP similar to that in previous sections using the linearized function. Specifically, we start the search with a random modification of the initial state, and obtain a linear approximation of the Q-function in the vicinity of this state. The attacker's objective is now a linear function of the state. We then formulate an ILP to compute a state that minimizes this objective. The search then continues by updating the linear approximation around this local solution.

To illustrate, we derive the algorithm in the context of a neural network with a single fully connected hidden layer, and a separate output for each action. The input , hidden and output layers have $m, |H|$ and $|A|$ units respectively. The hidden layer uses a rectified linear unit activation function. The output layer has a linear activation (since it predicts the action-value function which is basically unconstrained). Let θ^h and θ^o denote the weights associated with the hidden and the output layer respectively. With the above network architecture, the decision function is given by:

$$Q(\mathbf{x},a) = \sum_{j}^{|H|} \max\left[0, \sum_{i}^{m} \theta^h_{ij} x_i\right] \theta^o_{ja} \tag{5.6}$$

The first order Taylor series approximation of a multi-variable scalar-valued function is given by $f(\mathbf{x}+\delta\mathbf{x}) = f(\mathbf{x}) + \sum_i \frac{\partial f(\mathbf{x})}{\partial x_i}\delta x_i$. The linear approximation of the Q-function around $\mathbf{x}$ based on our neural network architecture

$$\begin{aligned} \hat{Q}(\mathbf{x}'_0, a) = Q(\mathbf{x}, a) + & \\ \sum_i \sum_j & \begin{cases} \theta^h_{ij}\theta^o_{ja}(x'_{i0} - x_i) & \text{if } \sum_i \theta^h_{ij} x_i \geq 0 \\ 0 & \text{if } \sum_i \theta^h_{ij} x_i < 0 \end{cases} \end{aligned} \tag{5.7}$$

Given this local linear approximation, we can compute the optimal interdiction using the following ILP:

$$\underset{\mathbf{x}'_0 \in \mathbf{X}, v, c}{\text{minimize}} \; v + \sum_i \rho_i (c_{i1} + c_{i2})$$

$$\text{subject to } v \geq \hat{Q}(\mathbf{x}'_0, a), \forall a \tag{5.8a}$$

$$c_{i1} - c_{i2} = x'_{i0} - x_{i0}, \forall i \tag{5.8b}$$

$$x'_{i0}, c_{i1}, c_{i2} \in \{0, 1\} \tag{5.8c}$$

The full search algorithm is outlined in Algorithm 7.

Algorithm 7 Interdiction with Local Linear Approximation

```
Start at a randomly chosen state $\mathbf{x}'_0$
for iterations in $1, \dots, T$ do
    Compute the linear approximation as in Equation (5.7)
    Solve the ILP 5.8 to update $\mathbf{x}'_0$
return $\mathbf{x}'_0$
```

5.5.3 Stabilizing the Q-Network

Finally, we describe two additional techniques that we used to stabilize the learning algorithm. First, at every step of training, the weights θ are updated. These constantly changing weights are used as targets in future iterations. The value estimations can thus, easily spiral out of control and destabilize learning. To alleviate this problem we use a second network to generate the target Q values and update its weights (to the primary network's weights) relatively less frequently [134]. Second, if the mean square error is relatively large for a sample, it can cause large changes to the network and destabilize it (the loss function is used in back-propagation for updating the weights). We use the Huber loss function to constrain this error [56].

5.6 Bayesian Interdiction Problem

Thus far, we had assumed that the interdiction game has *complete information*: both the defender and attacker know one another's utility functions and actions, as well as the initial state $\mathbf{x}_0$ that is being modified by the defender. Of course, in reality the defender is not privy to much of this information. We very generically model this uncertainty by partitioning the state variables X in the MDP into three groups: $X = (X^D, X^A, X^R)$ where X^D is the set of design variables that the defender can modify, X^A is the set of variables known only to the attacker (but not the defender), and X^R is the rest of the state variables (known to both players, but not modifiable by the defender). In particular, this uncertainty about the part of the initial state X^A may capture relevant access that the attacker possesses, or which actions are available to them (for example, by having these state variables model preconditions of relevant actions).

Let $\mathscr{X}^A$ be the set of all possible attacker *types* (i.e., initial attack states). The interdiction problem can be directly extended, with the defender solving the following minimization problem:

$$\min_{\mathbf{x}^D} \left[\mathbb{E}_{\mathbf{x}^A \in \mathscr{X}^A} \left(\mathscr{V}(\mathbf{x}^D, \mathbf{x}^A, \mathbf{x}^R)\right) + \rho(\mathbf{x}^D, \mathbf{x}_0)\right] \tag{5.9}$$

We can approximate this problem by replacing the expectation over a large set $\mathscr{X}^A$ with a sample average over a collection of samples of $\mathbf{x}^A$ according to the probability distribution over attacker types.

An important observation here is that the value function can be learned as a function of the *full* initial state, whether or not observed by the defender. Consequently, the proposed interdiction approaches discussed earlier can be extended directly to solve the above problem by introducing the variables and constraints specific to the sampled attacker types. In the case of neural networks, we can also extend the local search algorithms 6 and 7 by averaging over the attacker types at each search step.

5.7 Experiments

We evaluate our MDP interdiction algorithms on several instances of three problem domains from the international planning competition (IPC 2014): a) sysadmin b) academic advising and c) wildfire. While these examples have limited connection to security, they provide the most meaningful evaluation in terms of effectiveness and scalability. Previous work in security-related multi-stage attacks consider toy examples which would not provide appropriate evaluation. The most relevant prior work is [133], and offers a state-of-the-art solution to the problem. However, it considers action interdiction, rather than state interdiction. More significantly, we demonstrate that our baseline approach, which is closest to this work, has comparable running time on larger MDP instances (actually, tends to be faster). We use discount factor $\gamma = 0.9$ and set all interdiction costs $\rho_i = 1$. We denote the factored MDP interdiction as baseline (BI), and the linear and the non-linear interdiction approaches as LI, NLI1 and NLI2 respectively. We train the learning algorithms with $\varepsilon_0 = 1, \eta = 0.01$ and the RMSProp optimizer for the neural networks. The batch size $|\hat{\mathscr{D}}|$ increases from 40 to 400 with problem size. The experiments were run on a 2.4GHz hyperthreaded 8-core Ubuntu Linux machine with 16 GB RAM, with CPLEX version 12.51 for MILP instances and TensorFlow for learning algorithms [135].

5.7.1 MDP State Interdiction

First, we compare our interdiction approaches (optimized independently) on the sysadmin domain with $m = 10 - 60$ state variables and 11 to 61 actions. Each state variable corresponds to a machine and indicates whether it is working or has failed. We consider two possibilities for the initial state $\mathbf{x}_0$: a) all machines work and b) alternate machines work. In addition, we compare against the following cases, a) no interdiction (NI): when the initial state $\mathbf{x}_0$ is not modified, and b) random interdiction (RI): when the defender modifies the initial state randomly to $\mathbf{x}_0'$. The runtime and utility (defender's gains - interdiction costs) comparisons shown in (Figure 5.1) demonstrate that the proposed interdiction approaches significantly improve the defender's utility compared to the baselines (recall that lower is better). The defender utility is similar for BI,LI,NLI1,NLI2 in the experiments (for a given MDP domain). The utility differs significantly in case of NI and RI because these do not perform any optimization for interdiction and merely serve as baselines. In addition, we observe that the value function learning approaches scale much better than the baseline approach without compromising solution quality, and NLI2 tends to have the best scalability. This is primarily because these do not explicitly solve the MDP. In case of MDPs with higher order interdependencies in the transition model, scalability becomes a major bottleneck even with the approximate solution approaches with a limited number of basis functions.

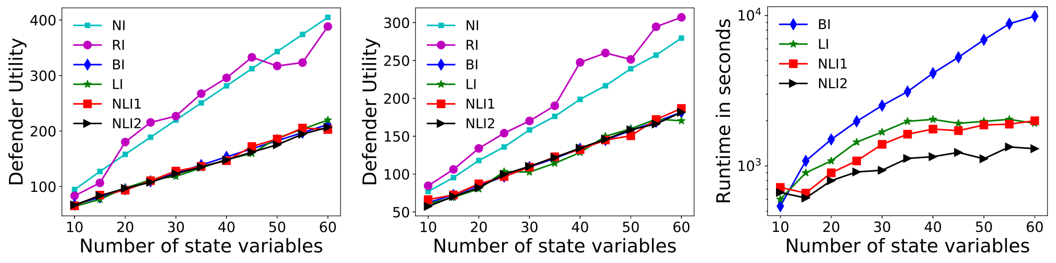


Figure 5.1: Comparison between the proposed interdiction approaches on the sysadmin domain in terms of utility (from two different starting states, left and center) and runtime (right).

Next, we evaluate our approaches on 10 problem instances of the academic advising domain. The problem size increases with problem number from 10 to 30 courses ($m = 20$ to 60 state variables, 10 to 30 actions). For each problem size, there are two instances, corresponding to different program requirements and course prerequisites. The first (odd numbered) problem instance is simpler (fewer prerequisites per course). The second (even numbered) instance is more complicated, with a larger number of prerequisites per course (larger number of connections in the underlying DBN). Problem 10 has the largest problem size with 30 courses, 11 program requirements, 3 prerequisites for most courses and 4 prerequisites for 8 courses. The two initial states correspond to selection of a) all courses and

b) alternate courses. As demonstrated in Figure 5.2, we observe a similar trend as before: effectiveness of optimized interdiction and superior scalability in case of the learning-based approaches.

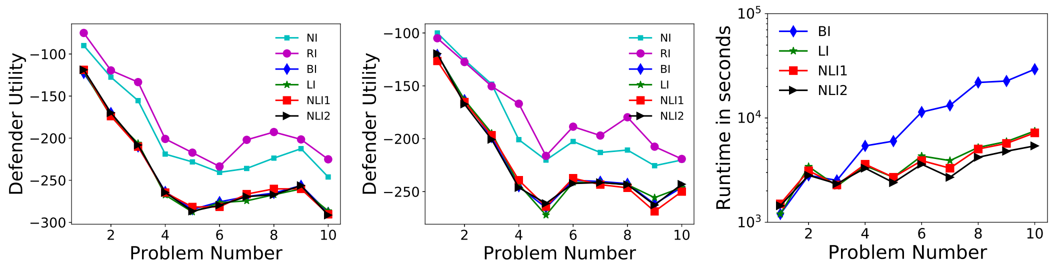


Figure 5.2: Comparison between the proposed interdiction approaches on the academic advising domain in terms of utility (different starting states, left and center) and runtime (right).

Finally, we evaluate on 6 problem instances of the wildfire domain. The problem is defined on a grid and the size increases with problem number from $n = 3$ to 5 ($m = 2 \times n^2 =$ 18 to 50 state variables, 36 to 100 actions). For each grid size, there are two instances, corresponding to different neighbourhood configurations and targets (cells on the grid that need to be protected). The first (odd numbered) problem instance has fewer targets than the second (even numbered) instance. In this case, we scale down the original rewards by a factor of 100 to ensure better convergence of the learning algorithms. The results are shown in Figure 5.3, and are broadly consistent with previous observations.

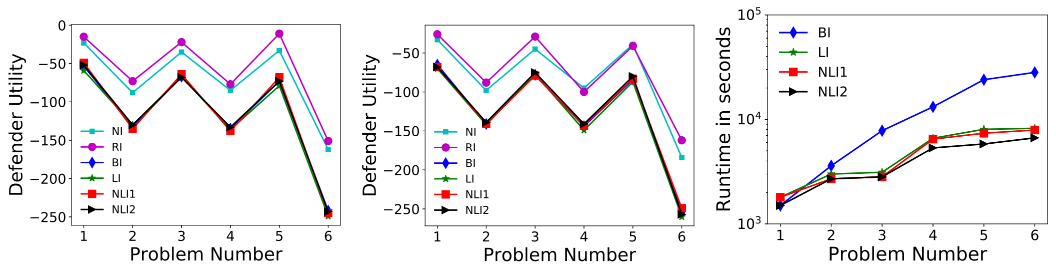


Figure 5.3: Comparison between the proposed interdiction approaches on the wildfire domain in terms of utility (different starting states, left and center) and runtime (right).

5.7.2 Bayesian Interdiction

Our final set of experiments deals with Bayesian interdiction. As a baseline, we consider interdiction of a worst-case attack. We obtain the initial state corresponding to this attacker by maximizing the approximate value function. The defender's problem is then given by $\min_{\mathbf{x}^D}[\max \mathscr{V}(\mathbf{x}^D, \mathbf{x}^A, \mathbf{x}^R) + \rho(\mathbf{x}^D, \mathbf{x}_0)]$.

In each of the domain examples considered, we divide the state variables as $X = (X^D, X^A, X^R)$, with 40%, 40% and 20% relative proportions. We randomly sample 500 assignments to the attacker's variables, with equal probability. In each case, we plot the difference in the defender's utility, between the baseline and the Bayesian cases. The results in the Figures 5.4, 5.5 and 5.6 exhibit a large *decrease in utility* (of the attacker), that is, a large benefit to the defender from considering Bayesian, rather than baseline, interdiction. While the runtime does increase somewhat, this increase is small compared to the time it takes to solve the MDP.

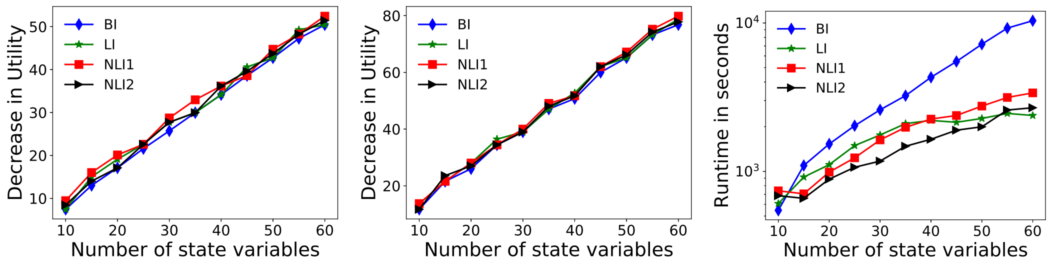


Figure 5.4: Improvement in the defender's utility using Bayesian interdiction in the sysadmin domain (different starting states, left and center) and interdiction runtime (right).

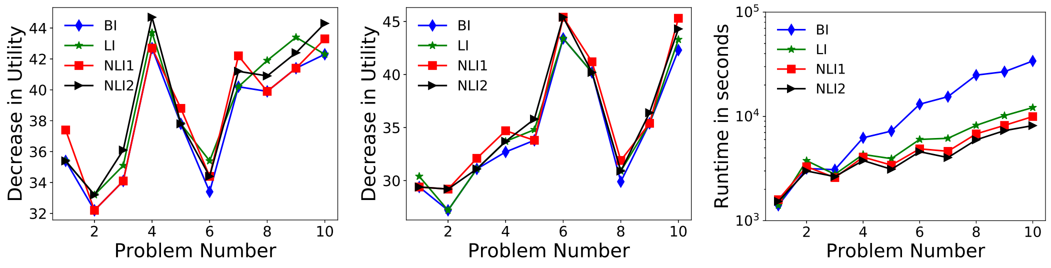


Figure 5.5: Improvement in the defender's utility using Bayesian interdiction in the academic advising domain (different starting states, left and center) and interdiction runtime (right).

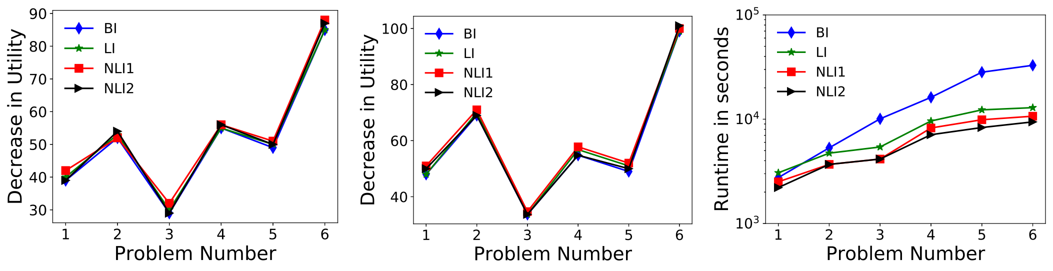


Figure 5.6: Improvement in the defender's utility using Bayesian interdiction in the wildfire domain (different starting states, left and center) and interdiction runtime (right).

5.8 Conclusions

We presented a novel interdiction model in which the defender constrains the initial state of the attacker. We proposed scalable interdiction techniques with single-level integer linear programming, compared to difficult bi-level problems discussed in previous work. We further improved scalability by using model-free reinforcement learning techniques with linear and non-linear action-value function approximators. We extended the interdiction framework to Bayesian interdiction. Finally, we evaluated the effectiveness of our proposed approaches on several realistic MDP problem instances.

Part II

Application of Plan Interdiction Games in Antibody Sequence Design

Chapter 6

THE ANTIBODY DESIGN PROBLEM

In this chapter, we transition to the immunology domain and recapitulate the vaccine design problem that we presented in the introduction of the dissertation. Infectious diseases pose a major threat to public health. In 2016, about 36.7 million people were living with HIV, and it resulted in 1 million deaths [136]. From the time AIDS was identified, it has caused an estimated 35 million deaths worldwide [137]. A recent Ebola outbreak in Africa killed thousands [138], and annual influenza outbreaks affect millions, with hundreds of thousands hospitalized, and thousands dying from the influenza or its side-effects [139].

Vaccination therapies are among the most important methods for combating infectious diseases. Vaccines are external substances that stimulate the immune system to produce antibodies that bind to the vaccine substance. As antibodies develop in response to a vaccine against a particular pathogen, they remain in the individual's bloodstream and rapidly neutralize and clear the pathogen if the individual is ever infected, thereby preventing illness. Traditional vaccine design involves laborious and costly lab work aimed at finding just the right substance which would successfully and reliably elicit antibodies binding the target pathogen. Recently, a promising approach has been taking shape in which vaccines are designed *computationally*, making use of modern computational protein modeling tools, such as ROSETTA [140]. One of the common approaches involves two steps: first, finding an antibody with desired neutralization characteristics, and second, finding a vaccine which binds tightly to the desired antibody, thereby eliciting the associated target immune response. We focus on the first step of computational antibody design.

The central goal in computational antibody design is to find an antibody protein sequence which neutralizes the target pathogen. In order for the antibody to neutralize a pathogen, it needs to *bind* to it; the specific position at which the two typically bind is called the *binding site*. When two proteins (such as an antibody and viral proteins) bind, they form a *complex*, which is a configuration minimizing the total energy of the two molecules. Binding is typically highly specific: a small change in the sequence can destabilize binding.

However, binding a single fixed antigen (portion of the pathogen which typically interacts with the antibody) is often insufficient: for example, viruses such as HIV and flu have many strains, and an antibody which neutralizes one will often fail to neutralize another. An area of active research in antibody design (computational and otherwise), therefore, is to develop and characterize *broadly binding antibodies*, that is, antibodies which effectively bind to (and, ideally, neutralize) many variants of the pathogen [17]. Nevertheless, as a pathogen evolves, it may well still escape neutralization; for example, HIV has an extremely high mutation rate [18].

6.1 Antibody Design as a Plan Interdiction Problem

We observe that the virus escape problem is a version of a planning problem in which the virus sequence seeks to optimize a series of mutations in order to escape binding to the antibody while maintaining important criteria for survival etc. This is analogous to an attack plan in case of plan interdiction. Given this scenario, the antibody's goal is to interdict the virus attack plan, or equivalently, the series of mutations until escape. This insight enables us to directly map the conceptual ideas in plan interdiction to the antibody design problem. Specifically, antibody design can be modeled as a Stackelberg game in which the antibody is the defender (leader) and the virus sequence is the attacker (follower). The virus observes the antibody sequence in action and computes a sequence of mutations to evade it.

6.2 Research Objectives

To summarize, an optimal antibody should have the following capabilities. First, it should achieve broad binding on a diverse set of virus sequences, i.e., the designed antibody should be able to bind to a diverse set of virus sequences. Second, it should be robust to virus escape mutations, i.e., the designed antibody should continue to bind as these virus sequences make mutations to escape. Finally, it should comply with energy stability considerations, i.e., the designed antibody should be stable in a complex (energy minimized configuration) with the virus. In the following chapters, we propose algorithms for antibody design which optimize the sequence space of the antibody primary sequence, in order to achieve the above objectives. However, we face significant conceptual and technical challenges.

The primary challenge is the enormous combinatorial search space on the antibody and target virus sequence space (the binding sites on the antibody and the virus side typically consist of 30 positions each and there are 20 possible amino acids resulting in a search space of $\geq 20^{60}$). The second challenge is determining whether an arbitrary antibody-virus pair bind. For this purpose, we make use of ROSETTA, a premier computational protein modeling tool [20]. However, ROSETTA 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). The third challenge is capturing the appropriate energy scores to reflect binding and stability. In addition to the ROSETTA antibody-virus binding energy score, we need to compute the overall energy of the antibody-virus complex to determine stability and viability of the designed antibodies. In the following chapters, we dive deeper into the research goals and develop efficient algorithms for broadly binding as well as robust antibody design.

Chapter 7

MACHINE LEARNING AND LINEAR OPTIMIZATION FOR BROADLY BINDING ANTIBODY DESIGN

7.1 Contributions

In this chapter, we develop efficient algorithms for broadly binding antibody design. Computational design has been used successfully by protein engineers for many years to alter the physicochemical properties of proteins [80, 81]. In the simplest case, protein design involves optimizing the amino acid sequence of a protein to accommodate a desired 3-D conformation. This approach has been extended to related tasks such as protein-protein interface design, de novo design of protein binding molecules, design of self-assembling protein nano-cages, etc. [82, 83, 84, 85]. Each of these examples involves the straightforward application of design methodologies to a single, static protein conformation. However, there is a need to extend protein design to apply to several conformations simultaneously. These approaches, referred to as multistate design (MSD), can be used to modulate protein specificity, model protein flexibility, and engineer proteins to undergo conformational changes [92, 93, 94, 95, 96, 97, 98]. Several methods have been developed to enable computationally expensive multistate design [21, 99]. However, these methods all suffer from large energetic barriers that limit sampling in sequence space, resulting in sub-optimal designs [21] . In addition, these methods are severely limited in scale by the size and number of states that can be included. To address these limitations, we develop a method that integrates structural modeling with integer linear programming to enable a fast global search through large ensembles of target states.

This work has been performed in collaboration with Alexander Sevy, Center for Structural Biology, Vanderbilt University, who contributed in terms of data generation, comparison to MSD and ROSETTA evaluations.

7.2 Experimental Workflow

Our design algorithm, which we call BROAD (BReadth Optimization for Antibody Design) incorporates ROSETTA-based structural modeling with integer linear programming to more easily traverse boundaries in the energy function (Figure 7.1). The experimental workflow involves generating a large training set of randomly mutated proteins, fitting a linear model (described below) to predict binding, and using integer linear programming to find an optimal antibody sequence balancing stability and binding with respect to a collection of target virus epitopes. We applied this method to the problem of designing broadly binding anti-HIV antibodies. We modeled anti-HIV antibody VRC23 [141] against a set of 180 diverse viral proteins, creating antibody variants that were mutated randomly in the paratope region. The viral panel used was derived from Chuang G-Y, et al [118]. Based on known binding patterns of VRC23 we calculated the predicted binding energy that corresponds to observable binding, and searched antibody space using integer linear programming to optimize stability of the unbound antibody while achieving predicted 100% binding breadth to the 180 target viral proteins. We then used a non-linear Support Vector Machine classifier, trained on the entire dataset produced by ROSETTA, to identify top sequences. Finally, we entered the top scoring sequences back into ROSETTA structural modeling to measure the predicted breadth of antibody variants.

7.3 Sequence-based Linear Classification and Regression Models to predict Binding and Stability

Our end goal is to design broadly binding and stable antibodies by searching the sequence space, i.e., to optimize the amino acids at each binding position of the antibody. The key challenge for this approach is that an exhaustive search in the combinatorial sequence space is intractable. To address this issue, we first propose to learn sequence-based linear classification and regression models to predict binding and stability from data. Build-

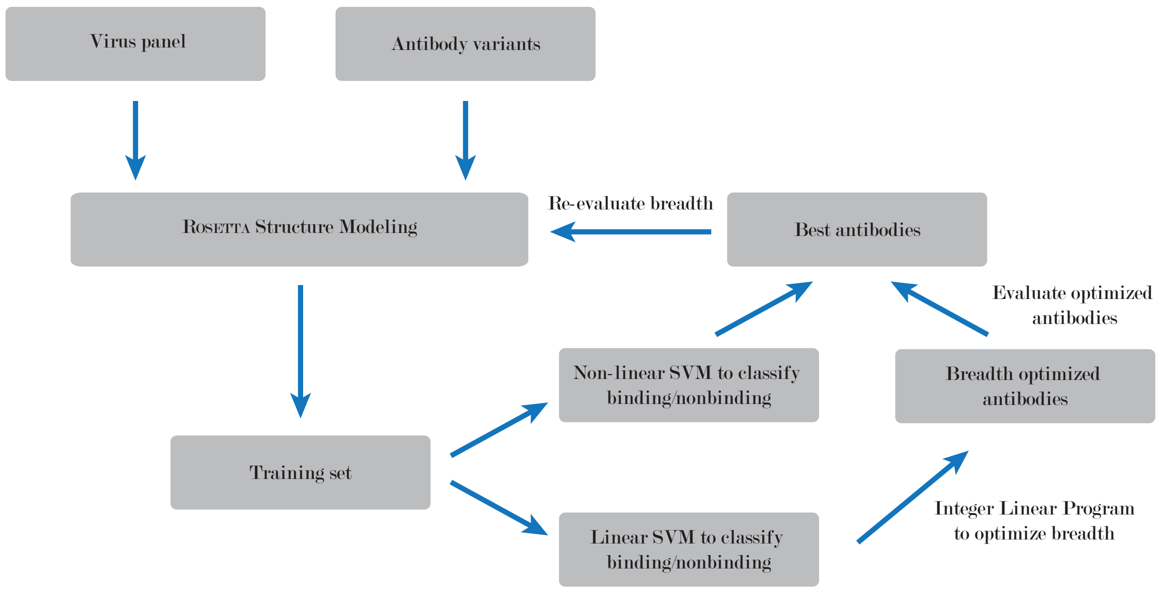

Virus panel
Antibody variants
Rosetta Structure Modeling
Re-evaluate breadth
Best antibodies
Evaluate optimized antibodies
Non-linear SVM to classify binding/nonbinding
Breadth optimized antibodies
Training set
Integer Linear Program to optimize breadth
Linear SVM to classify binding/nonbinding

Figure 7.1: Experimental workflow of the BROAD design method. The method uses ROSETTA structural modeling to generate a large set of mutated antibodies, support vector machines (SVM) to predict ROSETTA energy from amino acid sequence, and integer linear programming to optimize breadth of binding across a set of viral proteins.

ing on these models, we formulate an integer program to accomplish global search in the antibody sequence space. To generate our training set, we determined three contiguous stretches on the antibody that are in contact with the viral protein. These positions were determined to be residues 46-62, spanning FR2-CDR2-FR3; residues 71-74 in FR3; and residues 98-100b in CDR3 (Figure 7.2). We then created randomly mutated antibody variants, modeled their binding poses using ROSETTA, and used this data to train a binding classifier to predict ROSETTA score and binding energy from amino acid composition.

The binding classifier is based on the assumption that the amino acids at the binding positions of the antibody interact with those on the binding positions of the virus. In particular, this model assumes that binding between an antibody and a viral protein is determined by two factors: a) the individual amino acids in each binding position of the antibody and the virus respectively and b) the effects of the pairwise amino acid interactions between the antibody and the virus respectively. To capture these, we construct a sequence-based binary feature vector from the input antibody and virus pair, which explicitly represents the

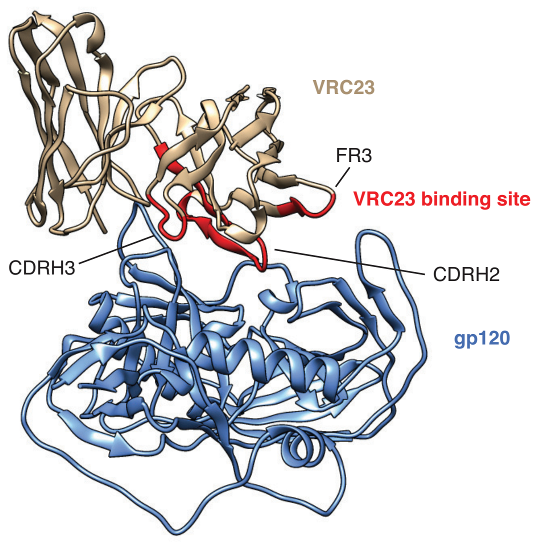
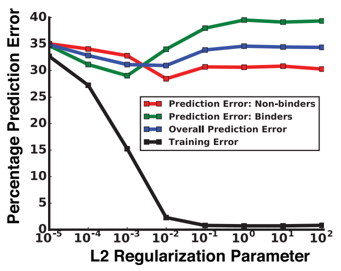
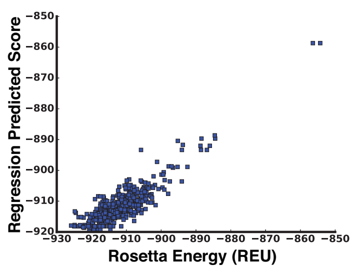


Figure 7.2: Binding site of VRC23 shown in context of the antibody-antigen complex. The binding site encompasses FR2, CDR2, FR3 and CDR3 regions of the antibody heavy chain.

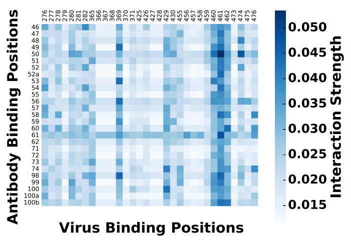
individual and pairwise amino acid contributions. Let the input antibody-virus pair represented as vectors of amino acids, be denoted by (a, v). Let $b(a, v)$ denote the ROSETTA predicted binding energy for (a, v) and let $F(a, v)$ denote the binary binding decision. We chose a threshold θ such that $F(a, v) = +1$ if $b(a, v) \leq \theta$ (i.e., a and v bind) and $F(a, v) = 1$ otherwise. For evaluation of our approach, we choose the value of θ based on experimental neutralization data. This data is available as the experimental neutralization IC50 (in units of μg/ml) of VRC23 with the 180 virus sequences in the panel [118]. Lower values represent better neutralization potency and values that have > 50 concentration represent a virus that is not neutralized by VRC23. Accordingly, VRC23 has a neutralization breadth of 63.5% on this panel. We set $\theta = -28.5$ such that the VRC23 breadth of binding computed on the ROSETTA generated data (sequences and the corresponding ROSETTA binding scores) is consistent with the above experimental neutralization data. We learn the classifier $F(a, v)$ as a linear Support Vector Machine (SVM) [142] using the binary feature set comprised of actual antibody and virus sequences along the corresponding binding sites, as well as all pairwise interactions of antibody and virus amino acids. The SVM classifier uses the ROSETTA binding energy as the ground truth, and allows more efficient sampling by approximating the ROSETTA score function by sequence alone. To optimize the L2 regularization parameter of the SVM, we performed 10-fold cross-validation on the full dataset, using 80% of the data for training and 20% for testing. Smaller parameter values enforce higher regularization and higher values lead to overfitting. The average prediction accuracy is shown in Figure 7.3 A for different values of the L2 regularization parameter. We also plot the prediction error on the two classes: binders $(+1)$ and non-binders (-1). The prediction accuracy is 67% on the test set using the optimized parameter (a random predictor would be at 50%). We observe that even if the prediction accuracy is relatively low, it provides reasonable signal within the subsequent breadth optimization step (discussed in the results section). Since the final decision is determined by solving the breadth optimizing integer linear program, our approach does not rely on a highly accurate classification model. In previous research [143], a similar model was introduced to predict δG values for interaction between PDZ domains and peptide ligands. The result was a 0.69 correlation coefficient in 10-fold cross validation. This model can also be interpreted to identify the important binding position pairs that contribute significantly to the final prediction. We plot this interaction strength for each pairwise interaction in Figure 7.3 C (please refer to the methods section for details). Next, we learned a linear regression model to predict the thermodynamic stability, using only the antibody amino acids as features. The prediction of thermodynamic stability is necessary to ensure that our designed antibodies can be expressed stably. To simplify the approach, we predicted the stability of the antibody-virus complex as a function of the antibody sequence only (note that we do not make this assumption during evaluation). Specifically, we constructed a binary feature vector restricted to amino acids in the antibody binding positions. Let $s(a,v)$ denote the ROSETTA stability for the pair (a,v). We learn a linear model $C(a)$ to predict $s(a,v)$ for an antibody a (i.e., independent of the virus). To measure the accuracy of prediction, we computed the correlation coefficient between the true scores and the predicted scores. Interestingly, our assumption that stability scores are only weakly dependent on the virus protein sequence is borne out: we found a correlation of 0.85 between the predicted and actual stability energy score on the test set (Figure 7.3 B).



A



B



C

Figure 7.3: Training results for the linear classification: (a) 10-fold cross validation results. (b) Correlation between predicted score and ROSETTA energy score in linear regression. (c) Interaction strength of each pairwise interaction between antibody and virus binding positions.

7.4 Algorithm

Given the classification and regression model learned from data, we formulate an integer linear program (ILP) to optimize the amino acids in the antibody sequence space to achieve both breadth and stability. The variables are the amino acids in the antibody binding positions. The objective function optimizes the predicted stability score (i.e., minimizes $C(a)$). The constraints represent the condition that the designed antibody should bind to all the viruses in the panel, using binding predictions from $F(a,v)$. We found that this problem was always feasible: there always existed some antibody sequence that could bind to all viral proteins based on our learned binding model. More generally, we can impose a minimal binding breadth criterion. This algorithm is outlined inFigure 7.4.

Armed with these tools, we used the following protocol to generate a collection of candidate antibodies to be evaluated using ROSETTA. First, we took a random subsample of the full training data corresponding to 100 out of the 180 virus sequences. Using only this subsample, we trained the binding and stability models, F(a, v) and C(a) respectively. We then solved the ILP described above to compute a stable, broadly-binding antibody sequence,

```
function SolveILP
    Input: linear binding and stability models
    Output: optimized antibody sequence
    Variables: amino acids at the antibody binding sites
    Objective: maximize stability
    Constraints: the antibody should bind to each virus sequence in the
training set
end function
```

Figure 7.4: Pseudocode describing the Integer Linear Program.

considering only the 100 out of 180 selected virus sequences (that is, we only constrain the ILP to bind to these 100 virus proteins, rather than the full set of 180). We repeated this procedure 50 times, to obtain 50 candidate antibody sequences. To validate these optimized antibody candidates, we predicted binding and stability scores using a model trained on all the data. In case of stability prediction, we used a linear model as described above (since the model is reasonably accurate). For binding prediction however, we trained a non-linear (radial basis function kernel) SVM for improved prediction accuracy. Each of the 50 candidate antibodies were scored using these models trained on all data, in terms of predicted binding breadth and stability, and 10 best candidates were then chosen for Rosetta evaluation using the full panel of 180 virus proteins. This procedure is outlined in Figure 7.5.

```
Generate Data: Rosetta(virus panel,antibody variants)
Learn Models: binding $\Phi$ and stability $\Psi$ on all data
Choose 50 random subsamples of 100 viruses
for each random subsample of 100 viruses do
    Learn linear binding and stability models
    SolveILP
    Evaluate breadth (against full panel) using $\Phi$ and $\Psi$
end for
Choose top optimized antibody candidates
Evaluate: Rosetta structure modeling on full panel
```

Figure 7.5: Pseudocode describing the BROAD algorithm for design of broadly binding antibodies.

7.5 Results

7.5.1 Redesign of VRC23 Improves Predicted Breadth

After generating redesigned antibody sequences with predicted increases in breadth, we threaded these sequences onto the VRC23-gp120 complexes and subjected them to structural modeling to measure the change in predicted breadth. We refined the complexes using the ROSETTA relax protocol. To test the accuracy of the ROSETTA relaxed models, we compared the relaxed models to solved structures of gp120 viral variants and computed the root mean squared deviation (RMSD) over Cα atoms on gp120. We observed that the relax protocol recapitulates the gp120 conformations with an average RMSD of 2.2 $\AA$, whereas the pairwise RMSD between gp120 conformations, representing the intrinsic flexibility of these molecules, is 1.8 $\AA$ (Table 7.1).

gp120 for indicated HIV strain	PDB ID	RMSD between relaxed model and crystal structure
Q23-17	4j6r	1.4
YU2-DG	3tgq	2.1
Du172-17	5te7	2.8
RHPA-7	5t33	2.2
X2088-c9	5te4	2.5
ZM109-4	3tih	1.9
JRCSF-JB	4r2g	2.4
HXB2-DG	1g9m	2.5
Q842-d12	4xmp	2.4
Average	2.2	

Table 7.1: ROSETTA relaxed models used in BROAD optimization were compared to solved structures of gp120 viral variants and the root mean squared deviation (RMSD) was computed over Cα atoms on gp120. The relax protocol recapitulates the gp120 conformations with an average RMSD of 2.2 $\AA$

Considering that we substituted only residues at the binding site of the gp120 variants, and not the entire gp120 sequence, we consider that the variant gp120 conformations are recapitulated with sufficient accuracy for this experiment. As a control, we generated sequences using structure-based multistate design with the RECON method [21]. The RE-

CON method uses ROSETTA design combined with coordination between differing states to generate an antibody sequence with increased affinity for all target states. Using RECON to redesign antibody-antigen complexes has been benchmarked and been shown to generate germline-like, broadly binding antibodies [21]. We compared the 10 sequences created by BROAD to 10 sequences generated by RECON multistate design to compare the change in breadth to alternate approaches. We found that the BROAD method resulted in a significant increase in predicted breadth over the RECON multistate design method (Figure 7.6 A). The BROAD-designed antibodies were able to achieve predicted breadth ranging from 86.1-100% of viruses, whereas multistate designed antibodies reached a predicted breadth of 62.8 - 85.6% of viruses. Notably, both methods were able to increase predicted breadth from the starting value of 53.3% for wild-type VRC23. This finding suggests that the wild-type VRC23 sequence is sub-optimal for breadth, which is supported by the observation that other known broadly neutralizing antibodies bind in a similar mode to VRC23 but with breadths exceeding 85% [144, 145, 146, 147]. In addition, we observed that the BROAD method samples sequence space that is not sampled in multistate design (Figure 7.6 B). We hypothesize that the BROAD method is able to cross energetic barriers that restrict sampling in traditional structure-based design methods, and is thereby able to generate antibodies with greater predicted breadth and lower energy. To support this hypothesis we analyzed the difference in score and binding energy for antibodies designed by BROAD and multistate design over the panel of viral proteins (Figure 7.7). BROAD was consistently able to generate lower energy antibody-antigen complexes, with a marked decrease in binding energy. This finding supports the hypothesis that BROAD is able to search sequences that are unavailable to multistate design, and that these new sequences have favorable score and binding energy.

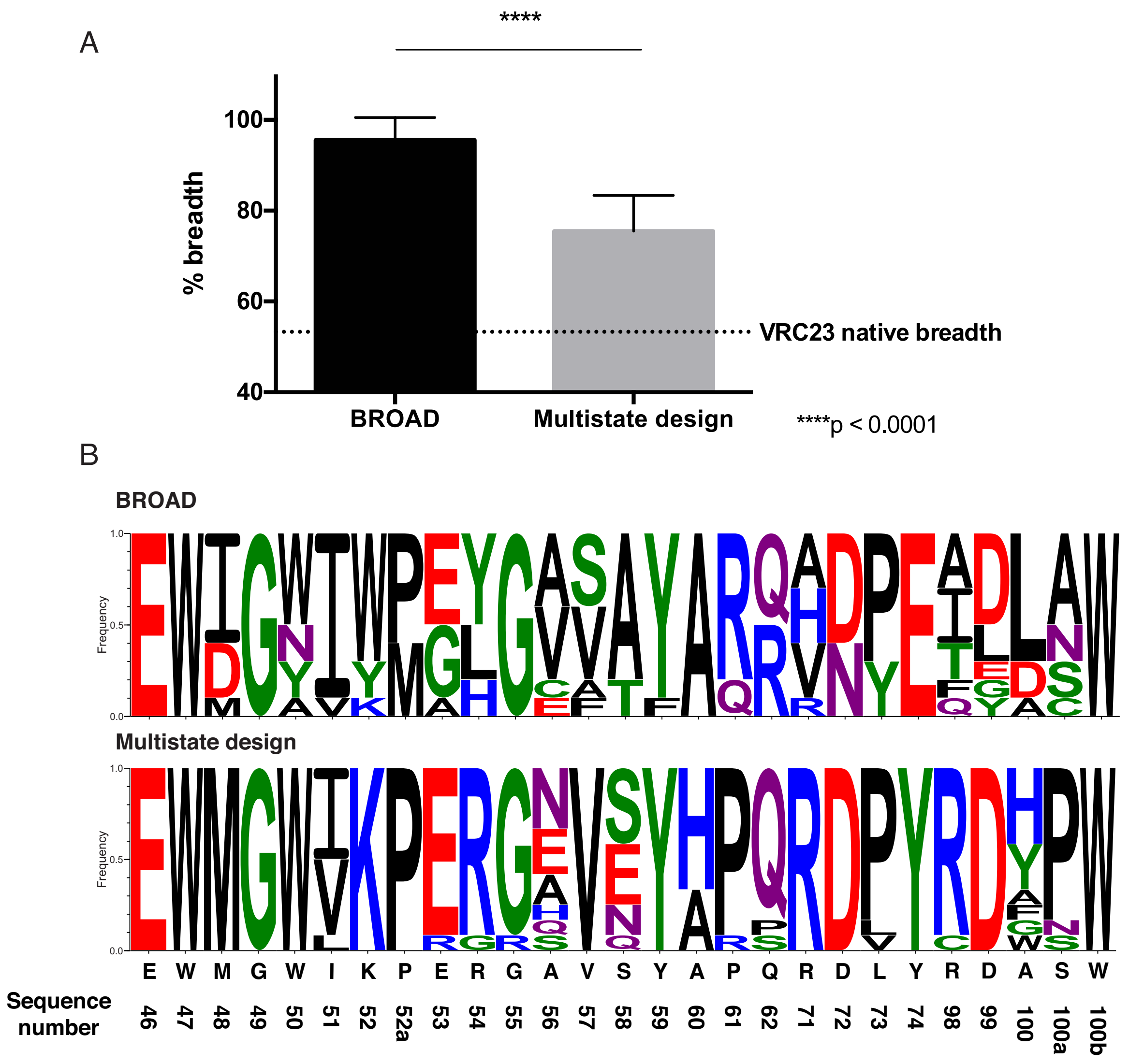


Figure 7.6: Redesign of VRC23 using integer linear programming increases predicted breadth over HIV viral strains. A. Predicted breadth of 10 redesigned antibodies generated either by BROAD or multistate design. Bars show mean and standard deviation of 10 sequences. Dotted line shows the predicted breadth of the native VRC23 antibody. B. Sequence logos of designed antibodies generated by BROAD or multistate design. Amino acids are colored based on chemical properties. The native VRC23 sequence is shown below.

7.5.2 Designed Residues Recapitulate Known Binding Motifs

A frequent problem in computational protein design is false positives, that is, sequences that are predicted to be favorable according to the score function, but are unable to recapitulate that activity in vitro. The ROSETTA score function uses many approximations

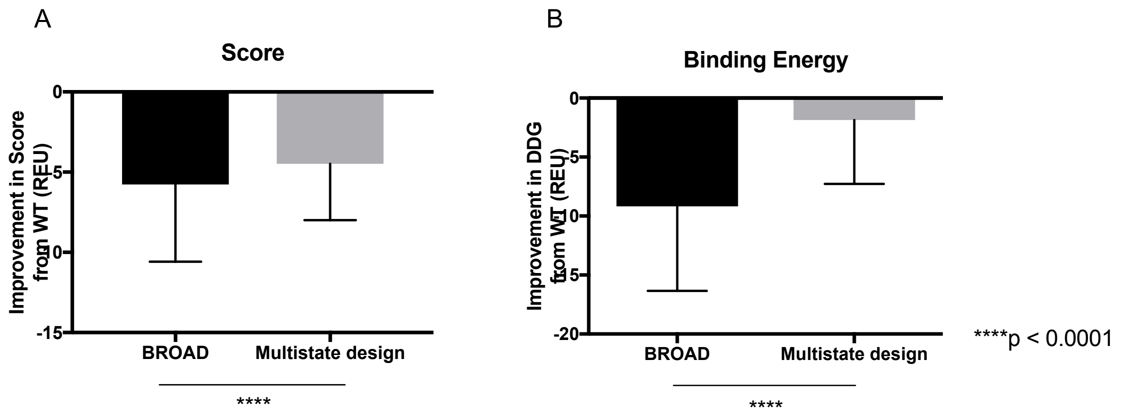


Figure 7.7: Score comparison of redesigned antibodies. The ROSETTA score (A) and binding energy (DDG) (B) are shown for ten redesigned antibodies made either by BROAD or multistate design, paired with 180 viruses. Bar plots shown mean and standard deviation. Shown on the Y axis is difference between score/DDG between the redesigned antibody and wild-type.

of energetic terms to enable faster simulations, and these approximations can introduce inaccuracies [148, 149]. To reduce the possibility that the redesigned VRC23 variants are scored favorably due to inaccuracies in the score function, we compared the designed residues introduced by BROAD to structural motifs of known broadly neutralizing antibodies (Figure 7.8). In several cases, the residues introduced by BROAD mimicked a known interaction of an existing antibody. For example, position 61 was mutated from proline in VRC23 to arginine (Figure 7.8, top left). The broadly neutralizing antibody VRC01 has an arginine that occupies similar space to the designed arginine [20]. This phenomenon can be observed for several different broadly neutralizing antibodies, such as VRC-CH31, 3BNC117, and NIH45-46, all of which target the CD4 binding site, but at slightly different orientations [144, 145, 146, 150]. We observed several examples of this type of recapitulation. Mutation Q62R on VRC23 placed an arginine residue to fill space that is occupied by a tyrosine on VRC-CH31 (Figure 7.8, top right)this mutation fills a void at the interface to improve antibody-antigen packing. Mutation L73Y places an aromatic group overlapping with the position of a tyrosine in antibody 3BNC117, which also improves packing with the antigen (Figure 7.8, bottom left). Lastly, the D102E mutant on the CDRH3 places a carboxylic acid group in the same position as a glutamic acid on NIH45-46, improving electrostatic interactions with the antigen (Figure 7.8, bottom right). This observation is remarkable due to the fact that the antibody loops occupy different space, but redesigned residues are able to mimic the interactions of the broadly neutralizing antibody side chains. In addition, it is worthwhile to note that out of these four mutants that recapitulate known broad motifs, three were unobserved in the sequences sampled by multistate design (Figure 7.6 B). As an additional comparison, we identified 1,041 sibling sequences of known broadly neutralizing antibody VRC01, that were isolated in a previous study [151]. These siblings presumably represent the sequence space accessible to VRC01, and are a good test case to compare how well our design algorithms are capturing natural sequence variation in a broad HIV antibody. Since these sequences have CDRH3 loops of different lengths we were not able to include the portion of the binding site corresponding to the CDRH3 loop. However we compared the rest of the binding site to the sequences seen in the VRC01 lineage (Figure 7.9). We observe that at several positions, BROAD samples sequences that are present in the VRC01 lineage but absent from MSD-sampled sequences (Figure 7.9, blue boxes). For example, at the third position in the binding site isoleucine is sampled at a high frequency in BROAD and VRC01 lineage sequences, but is never sampled by MSD (Figure 7.9). We highlight a total of five positions where BROAD outperforms MSD in sampling sequences that are seen in the VRC01 lineage. To quantify the sequence similarity we computed a sum of squared difference between the two matrices and normalized the values to 100% [21, 152]. According to this metric the sequences sampled by BROAD are 79.5% similar to those from the VRC01 lineage, whereas those sampled by MSD are only 76.3% similar. We conclude that BROAD more accurately recapitulates motifs known in broadly neutralizing antibodies.

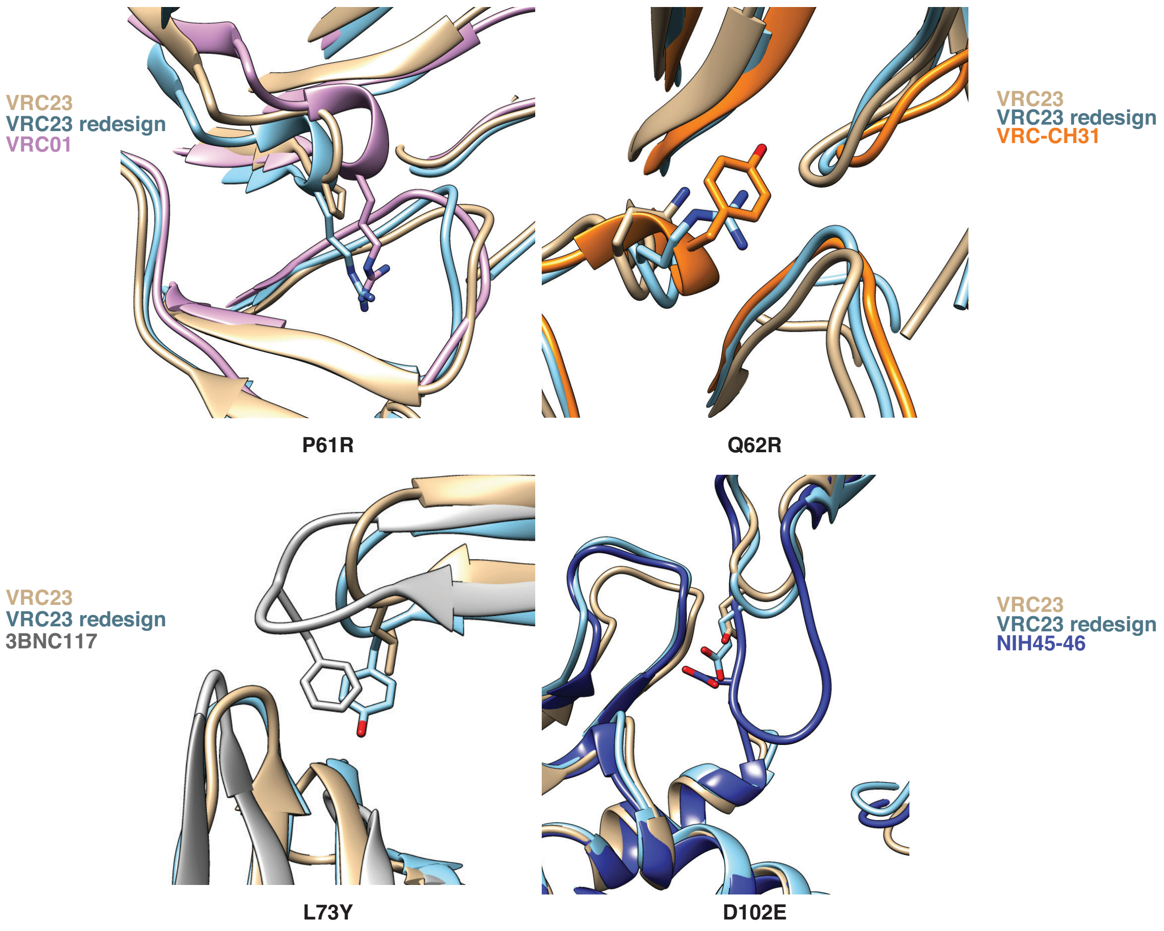


Figure 7.8: BROAD design recapitulates structural motifs of known broadly neutralizing antibodies. Residues that were mutated from the native VRC23 sequence were compared to known antibodies. Proteins shown are VRC23 (PDB ID: 4j6r); VRC01 (3ngb); VRC-CH31 (4lsp); 3BNC117 (4jpv); and NIH45-46 (3u7y).

7.6 Discussions

7.6.1 Summary of Results

In this paper, we describe the development of a new protein design method that we call BROAD. This method uses structural modeling with ROSETTA combined with integer linear programming optimization techniques to rapidly search through sequence space for broadly binding antibodies. We validated this method by computationally optimizing the amino acid sequence of the broadly neutralizing anti-HIV antibody VRC23. After modeling VRC23 variants in silico we were able to generate VRC23 variants with a

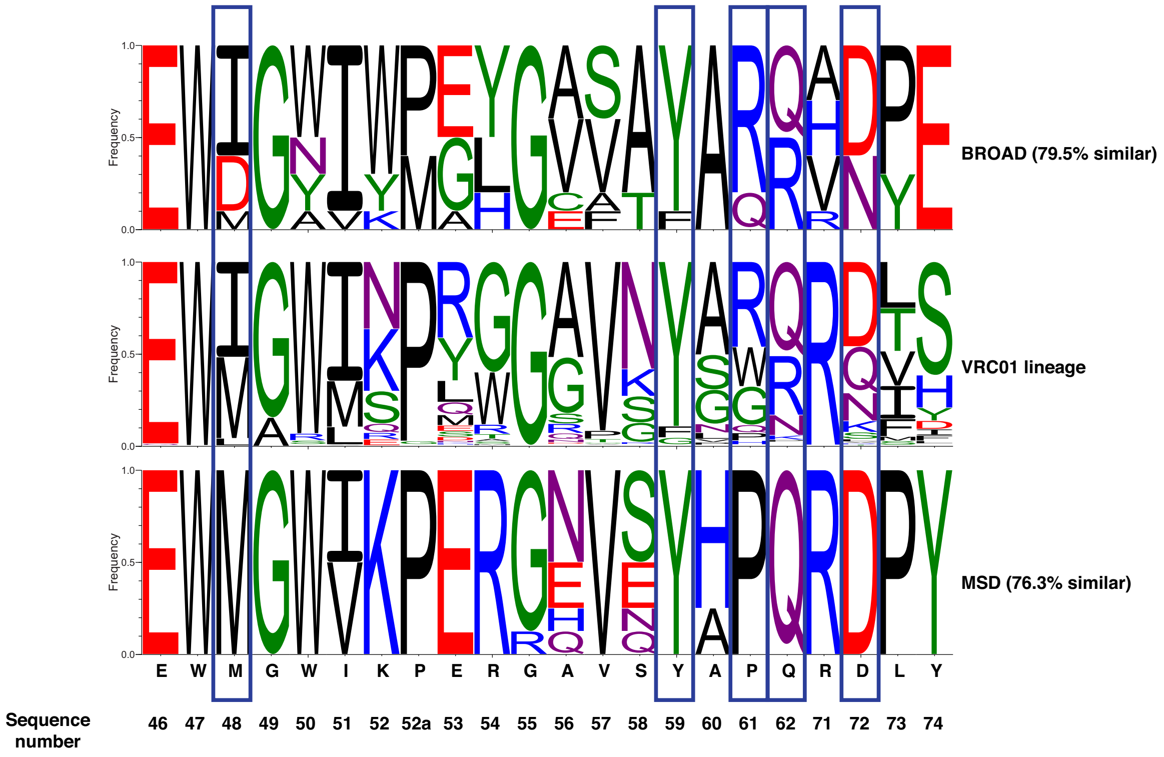


Figure 7.9: Sequences from BROAD design recapitulate sequences observed in the lineage of broadly neutralizing antibody VRC01. For BROAD and MSD sequences a percentage similarity to the VRC01 lineage was computed (similarity shown in parenthesis). Blue boxes highlight positions where BROAD samples an amino acid that is present in the VRC01 lineage but was not sampled by MSD. The VRC23 native sequence is shown below.

predicted breadth of 100% over the simulated viral panel, compared to a predicted 53% breadth for the wild type antibody. This outcome represents a substantial step forward in protein design, and our methodologies can be used to address a wide variety of protein design problems in which traditional structural models are insufficient. Although we did not test antibody variants in vitro in this study, we predict that the computationally designed variants will have greater breadth against the HIV viral panel. However, we note several caveats with respect to experimental validation of these antibodies. Since this experiment was designed as a computational proof of principle, we modeled only the amino acids at the antibody binding interface of gp120, and not the entire gp120 sequence. This led to gp120 models with 2 $\AA$ accuracy (Table 7.1), which we consider sufficient for validating our design principles but not necessarily for experimental validation. Future directions in this work include optimizing protocols for gp120 homology modeling to reduce this discrepancy and enable experimental validation.

7.6.2 Backbone Optimization in Protein Design

A distinct advantage of the BROAD method is the ability to truly incorporate backbone movement into protein design. Many protein design methods have been developed that incorporate backbone ensembles to some degree [96, 21, 153, 154]. However, this work typically involves either pre-generating large backbone ensembles, many of which may be redundant, or introducing backbone movement iteratively after steps of sequence design. In our approach, since we are relaxing the backbone of all mutants before fitting the sequence-based predictor, we were able to design sequences that may be slightly sub-optimal on the starting backbone coordinates, but can be highly favorable when a slight backbone relaxation is applied. This approach allows us to search sequence space that is not accessible to other methods, which are highly constrained to the initial backbone coordinates. We observed that the BROAD-generated sequences are not sampled by ROSETTA design using the RECON method, and indeed are more favorable according to the ROSETTA energy function. Therefore, we conclude that we are searching a ‘blind spot’ in the sequence space that is missed by traditional design.

7.6.3 Application to HIV Immunology

This approach to research could be of great utility to the field of HIV immunology. A longstanding goal of the field is discovering broadly neutralizing antibodies as the basis of a rational structure-based vaccine strategy [155, 156]. Much work has gone into redesigning existing antibodies to increase their breadth and potency [82, 145]. However, HIV is known for its variability, and with this variability comes a difficulty in generating a single antibody with potent neutralization against all possible variants. The BROAD method addresses this problem by enabling rapid redesign of known antibodies against viral panels of arbitrary size. This technology can be used in the future as part of the antibody discovery and characterization process, by rapidly searching sequence space for variants for greater breadth. In addition, protein design also has been used on the reverse side of the vaccination problem, namely, to design a vaccine with high affinity for antibodies of interest [157, 158, 159]. We can foresee the application of the BROAD method to this problem as well, by optimizing immunogens for recognition of germline precursors of known broadly neutralizing antibodies.

7.7 Materials and Methods

7.7.1 Structural Modeling

The VRC23-gp120 complex used for modeling was from the Protein DataBank (PDB ID: 4j6r). The structure was downloaded from the PDB (www.rcsb.org) and processed manually to remove water and non-protein residues. The CH1 and CL1 domains of the antibody structure were removed from the structure manually, and the structure was renumbered starting from residue 1. To select binding sites on the antibody and virus, we applied a distance cutoff of 4 $\AA$ from the opposing protein chain, where any residue with a heavy atom within 4 $\AA$ of a heavy atom on the opposing protein was considered to be at the binding site. Distance calculations were done using PyMol visualization software [160]. We expanded this binding site to several neighboring residues to include contiguous stretches of at least four residues to constitute a binding site. A total of 27 residues on the antibody were included in the binding site. We similarly determined a viral binding site to use for structural modeling. This site included 5 contiguous stretches that were determined to be in contact with VRC23 (32 positions total). These positions were 276282; 365371; 425430; 455462; and 473476 (HXB2 numbering). To model gp120 variants, we performed a multiple sequence alignment using ClustalW [161] of the variant sequences with the gp120 in the crystal structure (Q23.17), and substituted the corresponding amino acids at the binding site using ROSETTA side chain optimization [148].

7.7.2 Training Set

To generate a training set of structural models, we made random antibody substitutions in the previously defined binding site. Each antibody variant had five randomly selected amino acid mutations. Viral variants were taken from a set of 180 known HIV gp120 sequences [118]. We chose random combinations of antibody variants and viruses, as well as the native antibody sequence with all 180 viruses, for a total of 2200 antibody-virus pairs to serve as the training set. All antibody-virus pairs were subjected to an energy minimization via the ROSETTA relax protocol, which involves iterative rounds of side chain repacking and backbone minimization with an increasing repulsive force [162]. 50 models of each antibody-virus pair were generated by ROSETTA relax, and the lowest scoring model was used for further evaluation. The talaris2013 score function was used for all ROSETTA simulations.

7.7.3 Linear Classification and Regression

Our data-driven sequence-based model to learn amino acid contributions to binding and stability is similar to the graphical model approach proposed in [122]. Let N_a and N_v denote the number of binding positions on the antibody and the virus respectively. Let $\mathbf{A} = \{A_1, A_2, \ldots, A_{N_a}\}$ be a set of discrete random variables representing the amino acids in the binding positions of the antibody. Each A_i takes values in the set of $M = 20$ amino acids. Similarly, let $\mathbf{V} = \{V_1, V_2, \ldots, V_{N_v}\}$ represent the variables for the virus binding positions. The inputs for binding prediction are the antibody and virus sequences $\mathbf{a} = \{a_1, a_2, \ldots, a_{N_a}\}$ and $\mathbf{v} = \{v_1, v_2, \ldots, v_{N_v}\}$ where a_i and v_j are the amino acid values for the random variables A_i and V_j. Amino acid contributions to binding can be modeled as a bipartite graph in which nodes for $\mathbf{A}$ and $\mathbf{V}$ represent the amino acids and the edges $\Omega \subseteq \mathbf{A} \times \mathbf{V}$ represent the pairwise amino acid interactions. Each node a_i and v_j has associated weight vectors $\mathbf{x}_i$ and $\mathbf{y}_j \in \mathbb{R}^M$. The edge (i,j) between nodes a_i and v_j has an associated weight matrix $Q_{ij} \in \mathbb{R}^{M\times M}$ to represent the position specific contribution to binding for each amino acid pair. Consequently, given $\mathbf{a}$ and $\mathbf{v}$, the binding score varies as the sum of individual amino acids and pairwise interaction effects. Given this setting, $\mathbf{a}$ and $\mathbf{v}$ are predicted to bind, i.e., $\Phi(\mathbf{a},\mathbf{v}) = +1$ $(b(\mathbf{a},\mathbf{v}) \leq \theta)$, if

$$\sum_{i=1}^{N_a}\sum_{j=1}^{M} x_{ij}a_{ij} + \sum_{i=1}^{N_v}\sum_{j=1}^{M} y_{ij}v_{ij} + \sum_{k=1}^{N_a}\sum_{l=1}^{N_v}\sum_{u=1}^{M}\sum_{m=1}^{M} a_{ku}q_{kl}^{um}v_{lm} + c \leq 0 \tag{7.1}$$

where c is the intercept term and a_{ij} and v_{ij} are binary indicator variables that take the value 1 if amino acid j is present at position i ($\sum_j a_{ij} = 1, \sum_j v_{ij} = 1 \forall i$). The q_{kl}^{um} term represents $Q_{kl}(u,m)$. These weights can be efficiently learned using a linear SVM classifier. Formally, a SVM constructs a hyperplane in a high-dimensional space. A hyperplane that has the largest distance to the nearest training data point of a particular class intuitively achieves the best separation between the classes. The feature vector $\mathbf{f}$ consists of $N_a \times M$ binary antibody features, $N_v \times M$ binary virus features and $N_a \times N_v \times M \times M$ binary pairwise interaction features corresponding to $\mathbf{x}, \mathbf{y}$ and Q respectively. Given a set of d training instance-label pairs $(\mathbf{f}_i, l_i)$, $i = 1,\ldots,d$, $l_i = \{+1,-1\}$, a l2-regularized linear SVM generates a weight vector $\mathbf{w}$ by solving the following unconstrained optimization problem for the max-margin hyperplane: $\min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + \lambda \sum_{i=1}^{d}(\max(1 - l_i\mathbf{w}^T\mathbf{f}_i, 0))^2$, where $\lambda > 0$ is the regularization parameter. Smaller λ values enforce higher regularization. The second term is the squared hinge loss function. The decision function is given by $\text{sign}(\mathbf{w}^T\mathbf{f})$. We used the LIBLINEAR SVM implementation [163] to learn the classifier. Finally, the weights $\mathbf{x}, \mathbf{y}$ and Q are retrieved from the combined weight vector $\mathbf{w}$.

The linear regression model $\Psi(\mathbf{a})$ predicts the stability scores as a function of the antibody sequence features.

$$\Psi(\mathbf{a}) = \sum_{i=1}^{N_a}\sum_{j=1}^{M} x_{ij}^{s}a_{ij} + c^s \tag{7.2}$$

where $\mathbf{x}^s \in \mathbb{R}^M$ is the weight vector is regression and c^s is the intercept. Given a set of d training instance-score pairs $(\mathbf{a}_i, s_i)$ ($s_i = s(\mathbf{a}_i, \mathbf{v}_i)$, so there are multiple scores for the same antibody feature vector), $i = 1, \ldots, d$, the regression objective with l1 regularization is given by: $\min_{\mathbf{x}^s} \frac{1}{2d}(||(\mathbf{x}^s)^T \mathbf{a}_i + c^s - s_i||_2)^2 + \alpha||\mathbf{x}^s||_1$, where the first term is the least squares penalty, α is the regularization parameter and $||\mathbf{x}^s||_1$ is the l1-norm of the weight vector. We used the Lasso implementation in scikit-learn [164] to learn this model. In this case of a large feature space, l1 regularization promotes sparse corfficients and is a natural choice for feature selection. The feature set in case of the classifier is much larger. However, while training the classifier, l1-regularization assigns the majority of the weights to 0 and most of the signal from pairwise interactions is lost. Therefore we used l2-regularization to learn the classifier.

7.7.4 Breadth Maximization Integer Program

We leverage the weights in the binding and stability prediction models $\Phi(\mathbf{a}, \mathbf{v})$ and $\Psi(\mathbf{a})$ to formulate an ILP for optimization in the antibody sequence space. The objective is to maximize stability (minimize stabilty score). The constraints enforce the condition that the designed antibody should bind to each virus sequence in the training set. Finally, we add the constraint that the binary variables at each antibody binding position should sum to 1, i.e., each position takes only one amino acid. The ILP is given by the following:

$$\text{minimize} \sum_{k=1}^{N_a} \sum_{u=1}^{M} (x_{ku}^{s}) a_{ku}$$

subject to

$$\sum_{k=1}^{N_a} \sum_{u=1}^{M} \Big(\sum_{l=1}^{N_v} \sum_{m=1}^{M} q_{um}^{kl} v_{lm}^{n} + x_{ku} \Big) a_{ku}$$
$$+ \sum_{i=1}^{N_v} \sum_{j=1}^{M} y_{ij} v_{ij}^{n} + c \leq 0 - \varepsilon$$
$$\forall n \in 1, \dots, t$$
$$\sum_{u=1}^{M} a_{ku} = 1, \forall k, a_{ku} \in \{0,1\}$$

where $\varepsilon = 0.0001$. We used CPLEX version 12.51 to solve the above ILP on a 2.4GHz hyperthreaded 8-core Ubuntu Linux machine with 16 GB RAM.

7.7.5 RECON Multistate Design

VRC23 was placed in complex with all 180 viruses and designed via RECON multistate design to increase predicted breadth across the panel. Models of viral variants were created as previously described, by substituting amino acids at the binding site. All VRC23-gp120 pairs were refined by ROSETTA relax with constraints to the starting coordinates to prevent the backbone from making substantial movements. Constraints were placed on all Cα atoms with a standard deviation of 0.5 $\AA$. All residues at the binding site of VRC23 were included in design, for a total of 27 residues. The RECON protocol was run in parallel over 180 processors (manuscript describing parallelization in preparation), with four rounds of design and a ramping convergence constraint [21]. The binding sites on both the antibody and gp120 chain was subjected to backrub movements between rounds of design to increase sequence diversity [165]. A total of 100 designs were generated. Sequences generated by both BROAD and RECON methods were visualized using the WebLogo tool https://weblogo.berkeley.edu/logo.cgi.

7.7.6 Sequence Validation

To compare sequences generated by BROAD optimization and RECON multistate design, we threaded the optimized antibody sequences over the unprocessed VRC23-gp120 complexes, and subjected these complexes to ROSETTA relax to determine the score and binding energy of optimized antibodies vs. wild-type. 50 models were generated for each complex, and the lowest scoring model was used for evaluation. To compare native and optimized VRC23 sequences, we compared the total energy of the VRC23-gp120 complex as well as the binding energy (DDG), defined below:

$$DDG = E_{\text{complex}} - (E_{\text{Ab}} + E_{\text{Ag}})$$

where E_{Ab} and E_{Ag} are the energies of the antibody and antigen alone, respectively. Structures of modeled VRC23-gp120 complexes were visualized using Chimera software [83].

7.7.7 Comparison to VRC01 Lineage Sequences

VRC01 lineage sequences were derived from a previous study [151]. The 1,041 curated heavy chain sequences we used in this analysis are available in GenBank with accession numbers KP840719KP841751. To compare sequence profiles we used a modified Sandelin-Wasserman similarity score, as described in [166, 152] . Briefly, this score was calculated by computing the sum of squared difference for each amino acid frequency at each position, which was then subtracted from two and normalized to yield a percent similarity for each position and summed over all designed positions to give an overall similarity score.

Chapter 8

ANTIBODY DESIGN AS A STACKELBERG GAME

8.1 Contributions

In this chapter, 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.

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 [20]. 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.

8.2 Antibody Design as 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 [167]. 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:

$$\min_{v \in V} \|v^0 - v\|_0 \tag{8.1a}$$

$$\text{s.t.}: O(a,v) \geq \theta, \tag{8.1b}$$

where V is the space of virus sequences under consideration, and θ 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 (8.1) 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 (8.1). Let $v(a)$ be the solution to this problem—naturally, a function of the antibody choice a. The designer's decision problem is then

$$\max_{a \in A} \|v^0 - v(a)\|_0, \tag{8.2}$$

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

[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.

problem composing (8.2) with (8.1):

$$\max_{a \in A} \min_{v \in V} \|v^0 - v\|_0 \tag{8.3a}$$

$$\text{s.t.} : O(a,v) \geq \theta, \tag{8.3b}$$

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 [26, 27, 28]; 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 [28].

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 8, where a^0 and v^0 are the native antibody-virus pair. Algorithm 8 takes as a black box our ability

Algorithm 8 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 $a^*$
```

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 [168], 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.

8.3 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

[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.

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.

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.

8.4 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.

8.4.1 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 9).

[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.

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 θ. The escape cost is computed simply as the number of greedy iterations, e.

Algorithm 9 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
        $v \leftarrow \arg\max_{w \in V : \|v-w\|_0 = 1} \mathscr{O}(a, w)$
        $e \leftarrow e + 1$
    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 8.1 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.

8.4.2 Speeding Up Search through Learning

Figure 8.1 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

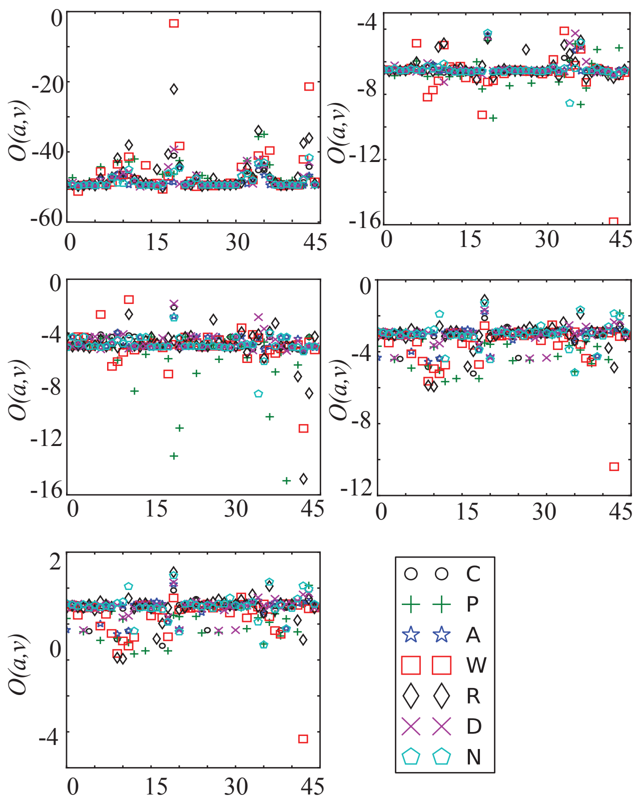


Figure 8.1: Example greedy search to compute escape cost. Horizontal axes correspond to point mutations in each virus sequence position, relative to the sequence from previous iteration, and vertical axis is the corresponding binding score. C,P,A,W,R,D,N correspond to the 7 amino acid classes that are candidate mutations.

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 10) 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 10 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 $e$
```

8.5 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.

8.5.1 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 10.

Simulated annealing: A relatively widely used stochastic local search method is *simulated annealing* [168]. Our variation of simulated annealing (Algorithm 11) 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 11 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($a', v^0$)
```

8.5.2 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 β 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.

8.6 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 8.2 (left).

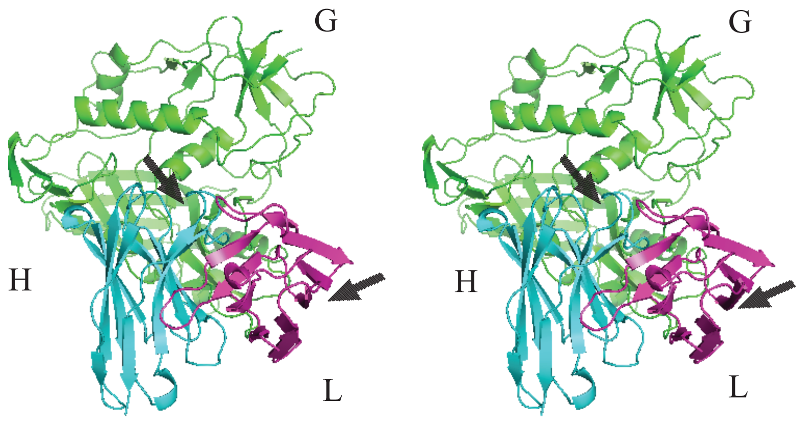


Figure 8.2: The native antibody, H and L, with the native virus, G (left) and antibody with escape cost=7 (right). The arrows point at some significant differences.

8.6.1 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 use

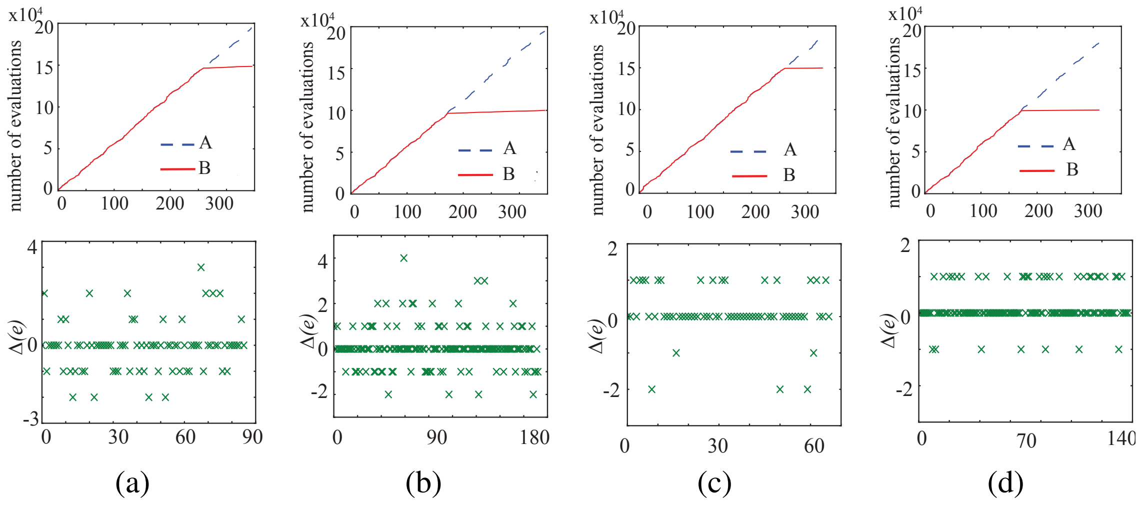


Figure 8.3: Comparison between baseline (A) and classifier-based greedy (B) algorithms for computing virus escape in terms of the number of evaluations (top) and computed escape time (bottom). (a) $\theta = 0$, 75% of data for training; (b) $\theta = 0$, 50% of data for training; (c) $\theta = -15$, 75% of data for training; (d) $\theta = -15$, 50% of data for training. Horizontal axes denote antibodies.

the *combined* running time expended both in collecting the training data and the evaluation. We used the LIBLINEAR SVM implementation [169], using l_2 loss and l_1 regularization. The ratio of $+1$ to -1 instances in the training data is $\sim$0.005 for both classifiers. The average accuracy for the classifier Ω 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 Ψ, 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 8.3. 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 8.3 (bottom) show a scatterplot of the escape time difference (Δe) 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.

8.6.2 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 [170] to fit Poisson regression parameters, using l_1 regularization. We find that the average 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 8.4 . 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 8.5 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 8.6). It is noteworthy that we found many antibodies which are much more robust to escape than the native when $\theta = 0$.

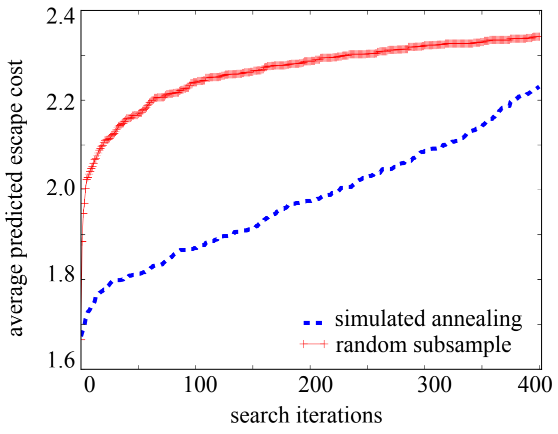


Figure 8.4: Antibody design algorithms comparison ($\theta = 0$).

8.6.3 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 8.2 (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 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

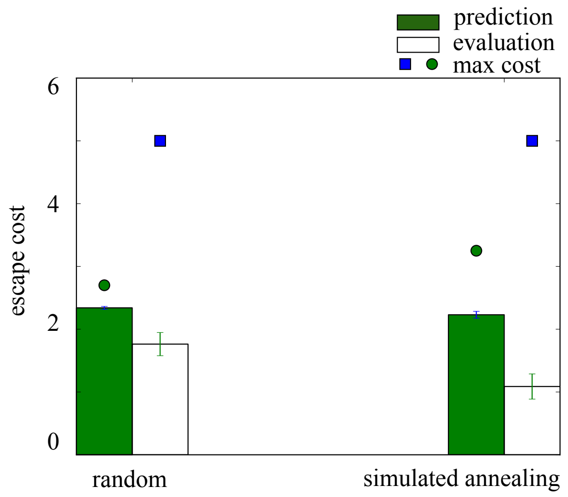


Figure 8.5: Antibody design algorithms comparison after 400 iterations averaged over 80 search sequences ($\theta = 0$).

significant difference in the ultimate breadth of binding, emphasizing the importance of a computational micro-level design approach.

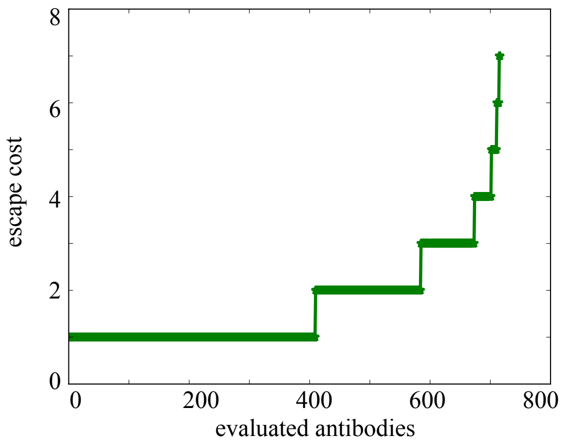


Figure 8.6: Evaluated antibodies for $\theta = 0$, ranked by escape cost. The native antibody escape cost is 1.

8.7 Discussions

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

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 [17]. Addressing this issue requires both extensive "wet-lab" evaluation, and, ultimately, clinical evaluation, both clearly outside the scope of this work.

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).

Chapter 9

LEVERAGING PROBLEM STRUCTURE FOR GLOBALLY OPTIMAL SOLUTION IN THE ANTIBODY DESIGN GAME

9.1 Contributions

In this chapter, we investigate deeper into the computational aspects of our radically different approach for antibody design in the context of rapidly mutating viruses: using a game theoretic (Stackelberg game) model for the interaction between the antibody and the virus. In this game, the antibody designer chooses an antibody sequence, while the virus aims to maximally destabilize binding to the resulting antibody, subject to a constraint on the number of mutations (this constraint captures the fact that such a mutation has to be sufficiently likely). This game can be formulated as a bi-level optimization problem; unfortunately, such a formulation is quite intractable. We address tractability in three steps: first, we learn a linear approximation of the antibody-virus binding score as a function of its sequence (including all pairwise interactions at the binding site); second, we formulate the optimal virus escape problem as an integer linear program; and third, after relaxing the integrality constraint in the virus escape program and taking its dual, we formulate the antibody design bi-level problem as a mixed-integer linear program. Our experimental results demonstrate that our approach is extremely effective against two recent prior approaches for HIV antibody design.

9.2 A Game Theoretic Model of Antibody Design

We define an antibody or virus primary sequence as a sequence (vector) of amino acids as in previous work [171]. Let $\mathbf{c}$ denote the native virus (the initial virus strain before mutations) and $(\mathbf{a}, \mathbf{v})$ be arbitrary antibody and virus sequences respectively. Let $\mathscr{B}(\mathbf{a}, \mathbf{v})$ and $\mathscr{S}(\mathbf{a}, \mathbf{v})$ denote the binding energy and the thermodynamic stability scores of the antibody-virus complex. A combination of these is used as the overall *energy score* (often known as the z-score) of the complex, which is what we actually work with, and denote by $\mathscr{Z}(\mathbf{a},\mathbf{v})$. Also, lower (more negative) scores indicate stronger binding and stability of the antibody-virus complex.

The virus sequence attempts to escape binding to the antibody by making a series of mutations. We can represent the number of mutations in a mutated virus sequence $\mathbf{v}$ from the native $\mathbf{c}$ as $\|\mathbf{v}-\mathbf{c}\|_0$, where the l_0 norm computes the number of sequence positions in $\mathbf{v}$ that are different from $\mathbf{c}$. Given an antibody $\mathbf{a}$, we model the virus as making up to α mutations with the goal of maximizing its binding energy score so as to destabilize binding. This model is motivated by natural selection: viral proteins which tightly bind to an antibody will be cleared by the immune system, leaving those which do not, and the remaining viral variants, mutating from a native sequence, will thereby increase in relative prevalence.

In general, there are many potential virus variants that can infect an individual. To capture this, we consider T virus sequences of different types t in a virus panel, each starting from a native sequence $\mathbf{c}^t$ and making mutations to escape binding to $\mathbf{a}$.

We can formally represent the optimization problem being solved by a collection of viruses as follows:

$$\begin{aligned} &\underset{\mathbf{v}^t\in\mathscr{V}}{\text{maximize}} \sum_{t=1}^{T} \mathscr{Z}(\mathbf{a},\mathbf{v}^t) \\ &\text{subject to } \|\mathbf{v}^t-\mathbf{c}^t\|_0 = \alpha, \forall t. \end{aligned} \tag{9.1a}$$

where $\mathscr{V}$ is the space of virus sequences under consideration. The optimization problem (9.1) can be viewed as the combined *best response* of the virus panel to a fixed antibody $\mathbf{a}$.

The space of feasible virus sequences $\mathscr{V}$ can be all possible combinations of amino acids in corresponding positions. However, in practice many such combinations are infeasible in nature, for example, because some mutations in specific positions destabilize the viral protein, or affect function. These considerations are too complex to capture cleanly. As a proxy, we constrain feasible mutations in each position to those which have been observed in nature (in that position) sufficiently often. More precisely, we only consider a mutation in a position i to an amino acid j if $p_{ij} \geq \theta$, where p_{ij} is the empirical frequency of the associated position-specific mutation, and θ an exogenously specified threshold ($\theta = 0$ is a natural choice, and one we use in the experiments; at this threshold, we only disallow mutations that have *never* been observed in nature).

In addition to only allowing mutations which are not too rare in nature, we impose another natural restriction on $\mathscr{V}$. Specifically, first-order effects in regard to its antibody binding properties are determined by the sequence that is a part of the native virus *binding site* (i.e., positions on the native virus sequence which are in contact with the native antibody in the original binding complex). Therefore, we only consider the problem of virus escape in terms of binding site mutations. This also allows us to significantly reduce the dimensionality of the problem in practice.

Now we consider the problem of designing an antibody, $\mathbf{a}$, that is robust to virus escape, as we have now formally defined using the optimization problem (9.1). The antibody designer's decision problem is then to choose an antibody which minimizes the energy scores (strengthens binding and stability) *with respect to the virus panel* $\{1,\ldots,T\}$, accounting for potential mutations of each virus in response. This gives rise to the following bi-level optimization problem for antibody design:

$$\min_{\mathbf{a}\in\mathscr{A}} \max_{\mathbf{v}^t\in\mathscr{V}} \sum_{t=1}^{T} \mathscr{L}(\mathbf{a},\mathbf{v}^t)$$

$$\text{subject to } \|\mathbf{v}^t - \mathbf{c}^t\|_0 = \alpha, \forall t \tag{9.2a}$$

where $\mathscr{A}$ is the antibody design space, which we restrict to the native binding site for the same reasons as for the virus. Observe that the antibody-virus interaction in our model can be viewed as a Stackelberg game in which the designer (antibody) is the leader, and each 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 minimizes the energy score, a quantity which is maximized by each virus t.

9.3 Solution Approach

9.3.1 A Bi-Linear Representation of Energy Scores

The optimization problem (9.2) is intractable in general, even when simulated using the ROSETTA software. In particular, computing such a function using ROSETTA even for a given pair of sequences requires many runs of stochastic local search, and takes on the order of minutes or hours. We make progress by approximating the complex black-box ROSETTA energy function $\mathscr{Z}(\mathbf{a},\mathbf{v})$ by a bi-linear function of the antibody and virus sequences, similar to the approach proposed by Kamisetty et al. [143]. The model is based on an assumption that the binding and stability of an antibody-virus complex is primarily determined by two factors: a) the individual amino acids in each binding position of the antibody and the virus respectively, and b) the effects of the pairwise amino acid interactions between the antibody and the virus. We now describe this model in detail.

We represent an antibody sequence $\mathbf{a}$ as a binary position by amino-acid matrix, with $a_{ij} = 1$ iff amino acid j appears in position i, and $a_{ij} = 0$ otherwise. Thus, $\sum_j a_{ij} = 1$, since exactly 1 amino acid can be in a given position. Similarly, virus protein sequence is represented as a binary matrix v_{ij} which is 1 iff amino acid j is in position i. Let N_a and N_v denote the number of binding positions on the antibody and the virus respectively, and let $M = 20$ denote the number of amino acids.

Amino acid contributions to the energy score can be modeled as a bipartite graph in which nodes represent the amino acids and the edges represent the pairwise amino acid interactions. Each antibody position node i has an associated weight vector $\mathbf{x}_i \in \mathbb{R}^M$. Similarly, each virus position node j has an associated weight vector $\mathbf{y}_j \in \mathbb{R}^M$. The edge (i, j)

between antibody position node i and virus position node j has an associated weight matrix $Q_{ij} \in \mathbb{R}^{M\times M}$ to represent the position specific contribution to the energy score for each amino acid pair. Consequently, given **a** and **v**, the energy score varies as the sum of individual amino acids and pairwise interaction effects. Given this setting, the z-score for a given pair **a** and **v** is defined as:

$$\mathscr{Z}(\mathbf{a},\mathbf{v}) = \sum_{i=1}^{N_a}\sum_{j=1}^{M} x_{ij}a_{ij} + \sum_{i=1}^{N_v}\sum_{j=1}^{M} y_{ij}v_{ij} + \sum_{k=1}^{N_a}\sum_{l=1}^{N_v}\sum_{u=1}^{M}\sum_{m=1}^{M} a_{ku}q_{kl}^{um}v_{lm} + I \tag{9.3}$$

where I is the intercept term and q_{kl}^{um} represents $Q_{kl}(u,m)$.

Our bi-linear model thus has four sets of parameters: $\mathbf{x}_i$, $\mathbf{y}_j$, and Q_{ij} for all pairs of antibody and virus positions, i and j, respectively, and the intercept I. We learn these parameters by generating a dataset of ROSETTA energy function values for a number of pairs of antibody and virus sequences (as detailed in the experiments).

Armed with the bi-linear model described in this section, we can convert the hard bilevel optimization problem into a significantly more tractable mixed-integer linear program through a combination of convex relaxation and duality, as we describe next.

9.3.2 Integer Linear Program for Virus Escape

Our first step is to formulate the virus optimal escape problem as an integer linear program.

We start by observing that the number of mutations α can be computed using a dot product with the sequence representation described above. Specifically, $\mathbf{v}^t \cdot \mathbf{v}^t = N_v$ and $\mathbf{v}^t \cdot \mathbf{c}^t = N_v - \alpha$. Moreover, $\mathscr{Z}(\mathbf{a},\mathbf{v})$ is now a linear function with the above sequence representation. These observations allow us to formulate the virus escape optimization in Equation 9.1 as an integer linear program (ILP). Since in this problem the antibody **a** is fixed, we can group the model in Equation 9.3 in terms of the variables **v** as $\sum_{i=1}^{N}\sum_{j=1}^{M} x_{ij}a_{ij} +$

$\sum_{i=1}^{N}\sum_{j=1}^{M}\left(y_{ij}+\sum_{k=1}^{N}\sum_{u=1}^{M}a_{ku}q_{uj}^{ki}\right)v_{ij}+I$. Thus, the virus escape ILP for a particular native virus indexed by t (from a collection of T of these) can be formulated as follows:

$$\begin{aligned}
\underset{\mathbf{v}^t\in\mathscr{V}}{\text{maximize}} \quad & \sum_{t=1}^{T}\sum_{i=1}^{N_v}\sum_{j=1}^{M}\left(y_{ij}+\sum_{k=1}^{N_a}\sum_{u=1}^{M}a_{ku}q_{uj}^{ki}\right)v_{ij}^t+T\sum_{i=1}^{N_a}\sum_{j=1}^{M}x_{ij}a_{ij} \\
\text{subject to} \quad & \sum_{j=1}^{M}v_{ij}^t=1,\forall i,t && \text{(9.4a)}\\
& N_v-\sum_{i=1}^{N_v}\sum_{j=1}^{M}v_{ij}^t c_{ij}^t=\alpha,\forall t && \text{(9.4b)}\\
& v_{ij}^t\le L(p_{ij}-\theta),\forall i,j,t && \text{(9.4c)}\\
& v_{ij}^t\in\{0,1\},\forall i,j,t
\end{aligned}$$

where constraint 9.4a enforces that the binary variables v_{ij}^t at each antibody binding position should sum to 1, i.e., each position admits one amino acid. The term $\sum_{i=1}^{N_v}\sum_{j=1}^{M}v_{ij}^t c_{ij}^t$ in constraint 9.4b computes the dot product $\mathbf{v}^t\cdot\mathbf{c}^t$. The constraint 9.4c encodes the constraint that we only allow mutations at positions to amino acids which have been observed at a frequency $p_{ij}\ge\theta$ as a linear constraint; here, L is a large number.

9.3.3 Mixed Integer Linear Program for Antibody Design

While we can represent the optimization problem faced by the virus *given a fixed antibody* using a linear integer program, our underlying problem of antibody design is still a bi-level problem. Such bi-level problems (with integer variables, as in our case) are, in general, extremely challenging to solve.

At the high level, we propose to leverage the linear structure of the problem to solve it. First, we relax the integrality constraint of the inner (virus escape) problem. This yields a linear program, the dual of which we embed into the outer integer linear program. By relaxation, combined with strong duality of linear programming, the resulting mixed-integer linear program minimizes an *upper bound* on the z-score objective with respect to optimal virus escape.

We start with the ILP 9.4 computing the optimal virus escape, and relax the integrality constraint; that is, we relax the binary v_{ij}^t variables to be continuous and add the constraints $0 \leq v_{ij}^t \leq 1$. Next, we show that the resulting relaxed LP has integral optimal solutions.

The standard form LP $\{\max w^T s : As = b, s \geq 0\}$ with integral right-handside vector b has an integral optimal solution if its constraint matrix A is *totally unimodular* [172], a notion we now define.

Definition 9.3.1 (Total Unimodularity)**.** *A matrix A is totally unimodular (TUM) if the determinant of each square submatrix of A is in* $\{0, 1, -1\}$*. In particular, each entry of A is in* $\{0, 1, -1\}$.

Theorem 9.3.1 (Sufficient Condition)**.** *A matrix A is TUM if it only has at most two non-zero entries 1 or -1 in every column, and for all columns with two non-zero coefficients, the column sum is 0.*

We next use this sufficient condition for TUM to prove that our LP relaxation yields optimal solutions to the original ILP. This result allows us to work with the relaxed LP for the virus escape problem.

Proposition 1. *The LP relaxation of the virus escape ILP 9.4 has integer optimal solutions.*

Proof. We first prove that the constraint matrix in the LP relaxation is TUM. Consider the LP relaxation with the constraints 9.4a and 9.4b and the non-negative variables. The additional constraints $0 \leq v_{ij}^t \leq 1$ are already enforced using 9.4a and the fact that all variables are non-negative. The corresponding constraint matrix has at most two non-zero elements in any given column corresponding to the variables v_{ij}^t. The first non-zero element +1 from the relevant constraint 9.4a and the second non-zero element -1 from 9.4b. Therefore, using theorem 9.3.1, the constraint matrix is TUM. Since the right hand side vector has integer elements, this LP relaxation has optimal integer solutions. This conclusion continues to hold after adding the constraints 9.4c since these only additionally restrict the variables to be zero under specific conditions. $\square$

We observe that the primal relaxed LP is feasible and bounded, and, therefore, the dual is also feasible and bounded, and (by strong duality) has the same solution as the primal. Let the associated (non-negative) dual variables be denoted by ψ_{ij}^t for each of the constraints $v_{ij}^t \leq 1$, and let ϕ_i^t (unrestricted), π^t (unrestricted) and ρ_{ij}^t (non-negative) denote the dual variables corresponding to constraints 9.4a, 9.4b, and 9.4c. Note that all dual variables are continuous. The dual LP is the given by the following (**a** is fixed here as in the primal LP):

$$\underset{\phi,\psi,\rho,\pi}{\text{minimize}} \sum_{t=1}^{T} \left[\sum_{i=1}^{N_v} \phi_i^t - (N_v - \alpha)\pi^t + \sum_{i=1}^{N_a}\sum_{j=1}^{M} L(p_{ij} - \theta)\rho_{ij}^t + \sum_{i=1}^{N_v}\sum_{j=1}^{M} \psi_{ij}^t \right] + T\sum_{i=1}^{N_a}\sum_{j=1}^{M} x_{ij}a_{ij}$$

$$\text{subject to } \phi_i^t - \pi^t c_{ij}^t + \rho_{ij}^t + \psi_{ij}^t \geq \Big(y_{ij} + \sum_{k=1}^{N}\sum_{u=1}^{M} a_{ku} q_{uj}^{ki} \Big), \forall i,j,t \tag{9.5a}$$

$$\psi, \rho \geq 0, \pi, \phi \text{ unrestricted variables}$$

Next, we integrate this dual LP into the antibody optimization problem in Equation 9.2 to formulate the following mixed integer linear program (MILP):

$$\underset{\mathbf{a}\in\mathscr{A},\phi,\psi,\rho,\pi}{\text{minimize}} \sum_{t=1}^{T} \left[\sum_{i=1}^{N_v} \phi_i^t - (N_v - \alpha)\pi^t + \sum_{i=1}^{N_a}\sum_{j=1}^{M} L(p_{ij} - \theta)\rho_{ij}^t + \sum_{i=1}^{N_v}\sum_{j=1}^{M} \psi_{ij}^t \right] + T\sum_{i=1}^{N_a}\sum_{j=1}^{M} x_{ij}a_{ij}$$

$$\text{subject to } \phi_i^t - \pi^t c_{ij}^t + \rho_{ij}^t + \psi_{ij}^t - \sum_{k=1}^{N}\sum_{u=1}^{M} a_{ku} q_{uj}^{ki} \geq y_{ij}, \forall i,j,t \tag{9.6a}$$

$$\sum_{u=1}^{M} a_{ku} = 1, \forall u \tag{9.6b}$$

$$a_{ij}^t \in \{0,1\}, \forall i,j,t$$

$$\psi, \rho \geq 0, \pi, \phi \text{ unrestricted variables}$$

The variables now include the binary antibody variables a_{ij}, and the constraints ensure that these sum to 1 at each antibody binding position, i.e., each position admits one amino acid. An important observation we can make is that while originally we had bi-linear terms involving antibody and virus decision variables, these are decoupled after taking the dual, resulting in solely linear terms.

9.4 Experiments

The data comprises the anti-HIV antibody VRC23 [141] (the native antibody) against a set of 180 diverse HIV gp120 virus sequences (derived from Chuang et al. [118]). To generate our training data, we make random antibody and virus substitutions in the binding sites of VRC23 and the set of 180 virus sequences ($N_a = 27$ and $N_v = 32$). Each antibody/virus variant has five randomly selected amino acid mutations. All antibody-virus pairs are subjected to an energy minimization via the ROSETTA relax protocol (iterative rounds of side chain repacking and backbone minimization [173], talaris2013 score function). We generate 50 models of each antibody-virus pair and choose the lowest scoring model in each case. We then construct the dataset for our experiments with a total of 7360 such random antibody-virus combinations (including VRC23 and the 180 virus sequences). We compute mutation frequencies (p_{ij} in our terminology) from an exhaustive database of over 66,000 HIV-1 sequences (from the Los Alamos HIV sequence database http://www.hiv.lanl.gov/). We set the threshold θ to be 0, excluding only mutations which are *never* observed in nature.

9.4.1 Bi-linear Z-score Model

The feature vector $\mathbf{f}$ consists of $N_a \times M$ binary antibody features, $N_v \times M$ binary virus features and $N_a \times N_v \times M \times M$ binary pairwise interaction features corresponding to $\mathbf{x}, \mathbf{y}$ and Q respectively. We use sparse matrices to represent this feature space and use the Lasso implementation in scikit-learn [174] with l_1 (sparse) regularization. To measure the accuracy of predictions, we compute the correlation coefficient between the ROSETTA computed z-scores and the scores predicted by regression. We perform a 10-fold cross validation experiment with 80% of the data for training and 20% for testing. Based on this parameter tuning, we choose regularization parameter $\lambda = 0.01$ with an average correlation of 0.85 between the predicted and the ROSETTA computed z-scores.

We denote our proposed antibody design approach as STRONG: STackelberg game theoretic model for RObust aNtibody desiGn and compare against the two prior approaches, a) BROAD [166] and b) the game theoretic approach proposed in [171] (henceforth denoted as AAMAS2015).

9.4.2 Comparison against BROAD

BROAD [166] is a state of the art algorithm for antibody design against a *fixed* panel of HIV virus variants that involves generating a large training set of binding and stability scores using ROSETTA, fitting linear models to predict binding and stability, and solving an ILP to compute an optimal broadly binding antibody sequence.

We perform the comparison following the experimental workflow in BROAD. We construct 50 random subsamples of the full training data corresponding to $T = 100$ out of the 180 virus sequences We train binding and stability prediction models on this data and compute the BROAD antibody sequence by solving an ILP with the T virus sequences in the training subsample. Next, for each training subsample we learn the bi-linear model in Equation 9.3 and save the coefficients. Then, we solve the MILP 9.6 to compute the corresponding STRONG antibody for a given α. Given this antibody, we solve ILP 9.4 to compute T escaping virus sequences corresponding to each of the T training sequences (native). We use CPLEX version 12.51 to solve the (mixed) integer linear programs. Finally, we train a z-score model on the full dataset ($T = 180$). We evaluate the BROAD and the STRONG antibody sequences in terms of the predicted z-score against a) the full 180 virus panel and b) the 100 escaping virus sequences in case of each training subsample. This procedure is outlined in Algorithm 12. As we show in Figure 9.1 STRONG is significantly better in minimizing the z-score objective as compared to BROAD.

Algorithm 12 Generating and evaluating STRONG antibody candidates

```
for each random training set subsample corresponding to $T = 100$ virus sequences do
    Training Data: $\mathscr{B}(\mathbf{a},\mathbf{v}), \mathscr{S}(\mathbf{a},\mathbf{v}), \mathscr{X}(\mathbf{a},\mathbf{v})$ corresponding to the $T$ training sequences
    Learning: bi-linear model for z-score
    Optimization: STRONG antibody $\leftarrow$ MILP 9.6 solution, escaping set $\leftarrow$ ILP 9.4 solution
    Evaluation: predicted z-score using model trained on the full dataset, and ROSETTA modeling
```

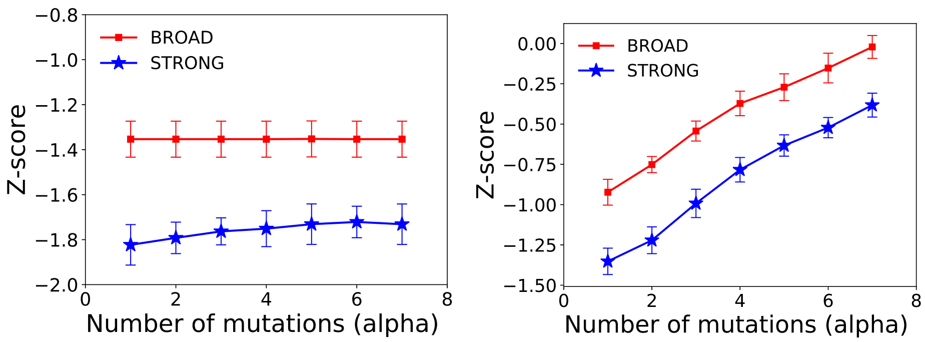


Figure 9.1: Comparison between STRONG and BROAD in terms of the z-score objective (lower is better): on the full 180 virus panel (left) and the 180 escaping virus set (right).

Finally, we evaluate in terms of the breadth of binding (fraction of viruses in the evaluation panel to which the designed antibody binds) generated using ROSETTA structure modeling. We choose 50 random subsamples of training sets with $T = 30$ virus sequences. Based on binding and stability models trained on the full dataset, we generate the top 10 BROAD candidates. Next, we generate the STRONG antibody for a randomly chosen top BROAD candidate using $\alpha = 5$. We perform ROSETTA structure modeling on these antibody candidates (one BROAD and one STRONG candidate) and the escaping set of 30 virus sequences. For comparison, we also include the native antibody VRC23. We present the ROSETTA computed breadth in each case in Table 9.1. STRONG significantly outperforms BROAD against the escaping virus panel while it continues to be effective against the training panel.

Virus Sequences for Evaluation	VRC23	BROAD	STRONG
180 HIV panel	53.3	100	96.1
30 Escaping virus sequences	43.3	86.7	93.3
30 Training virus sequences	56.7	100	100

Table 9.1: ROSETTA structure modeling results: breadth of binding (%).

9.4.3 Comparison against AAMAS2015

The game-theoretic antibody design approach in [171] uses machine learning guided stochastic local search to compute optimal antibodies. Following the biased random approach in the above research, we generate a set of 350 antibody sequences starting with VRC23. We compute the corresponding average escape costs (number of mutations to escape) with greedy local search starting from the 180 virus panel. We train a binary antibody-virus binding prediction model using an rbf kernel SVM on the full dataset and use this model in the greedy search to evaluate binding. Next, we learn a linear regression model to predict this average escape cost as a function of the antibody sequence. Using these models, we perform 50 independent sequences of local search (400 iterations, random with native bias [171]) to compute 50 optimal antibody candidates.

For comparison, we generate STRONG antibodies corresponding to the above 50 antibodies, with α set to the nearest integer escape cost in MILP 9.6. Using the z-score model trained on the full dataset, we evaluate these antibodies in Figure 9.2, on the full 180 panel and the escaping set in each case (from ILP 9.4). Our proposed approach is significantly better in minimizing the objective (z-score). We also plot the comparison as a function of the local search iterations and observe a similar trend in Figure 9.3. Note that the z-scores increase with iterations since the average escape cost increases as well.

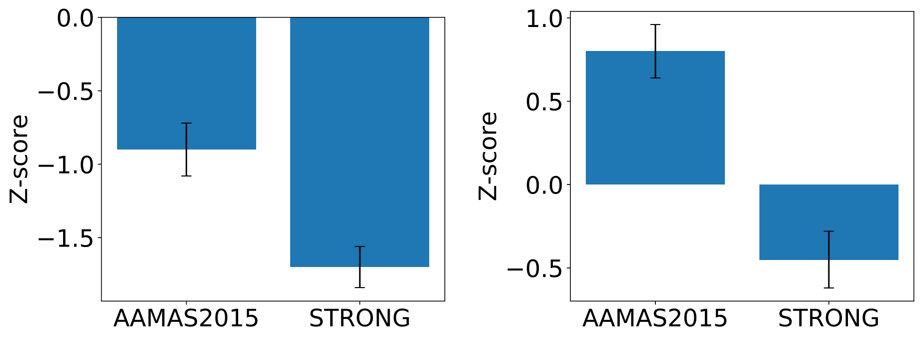


Figure 9.2: Comparison between STRONG and AAMAS2015 in terms of the z-score objective (lower is better): on the full 180 virus panel (left) and the 180 escaping virus set (right).

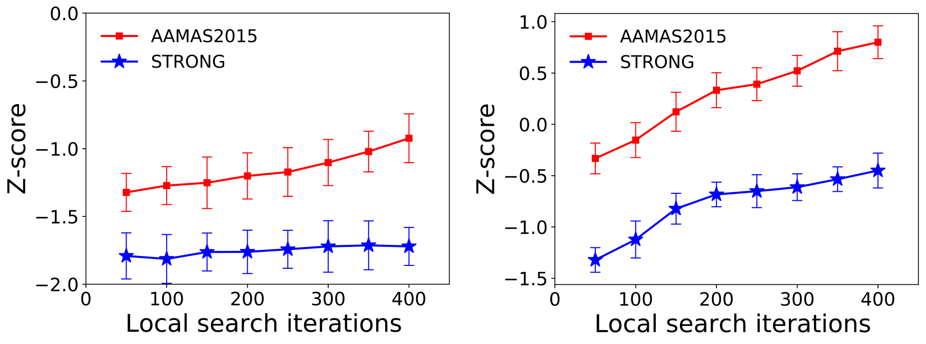


Figure 9.3: Comparison between STRONG and AAMAS2015 in terms of the z-score objective against search iterations (lower is better): on the full 180 virus panel (left) and the 180 escaping virus set (right).

9.5 Conclusions

We proposed an efficient approach for computational antibody design using a Stackelberg game model for the interaction between the antibody and the virus. We formulated the game as a bi-level optimization problem, and proposed a method for solving it by leveraging a bi-linear model predicting binding stability as a function of antibody and virus sequence, combined with integer programming. We show, in particular, that we can compute optimal virus escape using an integer linear program the LP relaxed version of which has integer solutions. Consequently, taking the dual of the associated relaxed LP we obtain an optimization program which can be directly embedded in the optimal antibody design problem, so that the antibody design problem can be solved using mixed-integer linear programming. Our experiments show that our approach significantly outperforms both the prior game theoretic alternative, and a state-of-the-art broadly binding antibody design algorithm.

Chapter 10

CONCLUSIONS

10.1 Summary of Contributions

Adversarial planning and decision problems arise in several real-world situations and these involve strategic interactions between multiple agents. In computing optimal decisions, one needs to consider the corresponding combinatorial strategy space. Therefore, a major challenge is to compute reasonably optimal solutions and to simultaneously ensure scalable decision making algorithms. In this dissertation, we achieve the following two major goals. First, we present plan interdiction games, specifically, Stackelberg game theoretic interaction to model large-scale adversarial decision problems. Second, we develop scalable algorithms to solve the corresponding bi-level optimization problems in the combinatorial strategy space.

In the plan interdiction framework, the defender deploys specific mitigation strategies to interdict the attacker's plans. The attacker then responds to the deployed mitigations by computing an optimal plan in that environment. In previous work, a Stackelberg game model formulation of plan interdiction has been considered, inspired by cybersecurity applications. In this model, the defender chooses a subset of actions to block (remove), and the attacker constructs an optimal plan in response. This previous research is in the context of deterministic (PDDL-based) planning and also, planning with uncertainty by modeling the attacker as a markov decision process (MDP). However, there are several challenges in representing and solving the MDP interdiction game, especially because the MDP state space is exponential in size of the state variables. In this dissertation, we present novel game-theoretic models for MDP interdiction and develop several efficient algorithmic approaches to alleviate the scalability limitations. Concurrently, we conjecture that the decision making framework in interdiction games is a natural fit for real-world applications beyond cybersecurity. In particular, we focus on a novel application area in immunology and vaccine design. A major problem faced by drug and vaccine designers is that the disease (virus) evolves and mutates (extremely rapidly in some cases e.g., HIV) and therefore escapes any prescribed treatment. The primary function of a vaccine is to generate antibodies in the immune system, the protective protein sequences that fight against and destroy any foreign infections (virus sequences). The goal of vaccine design, therefore, is to design or discover antibodies that are robust against specific viruses. A robust antibody should be effective against specific virus sequences and it should continue to be effective as these sequences mutate to escape. We consider the problem of robust antibody design as a variation of plan interdiction, with an antibody sequence as the defender and an escaping virus sequence as the attacker. Specifically, we model the interaction between the antibody and the virus as a Stackelberg game. Our work, to our knowledge, is the first game-theoretic model of molecular-level interaction between infectious disease treatment (antibodies or vaccines) and the disease (virus).

This dissertation consists of two parts. In the first part, we present game-theoretic models for MDP interdiction and develop efficient algorithms for computing optimal decisions for the defender. In each case, we demonstrate superior scalability through extensive experiments on realistic MDP problem domains from the international planning competition (IPC). In our proposed algorithms, it is important to note the trade-off between scalability and utility (solution quality). In general, if we select the entire set of basis functions, we will compute an exactly optimal solution. However, in our experiments, we demonstrate that it is possible to achieve dramatic speed up without compromising solution quality.

In the second part, we focus on robust antibody design as an interdiction problem. Specifically, the optimal antibody sequence (the defender) interdicts the escape plan of the virus sequence (the attacker). Before we consider a game-theoretic model, we focus on the general problem of broadly binding antibody design, i.e., we attempt to optimize the antibody sequence against a fixed panel of virus sequences. Our algorithm achieves 100% breadth of binding on a standard panel of 180 HIV sequences, as validated by extensive experiments using the state of the art protein modeling software ROSETTA. Next, we proceed to present a Stackelberg game-theoretic model for antibody and virus interactions. We develop several algorithmic approaches to solve the difficult bi-level optimization problems corresponding to the game. The specific contributions in this dissertation are summarized in the following sections.

10.2 MDP Interdiction

10.2.1 Factored Representation and Scalable Bi-Level Optimization

The primary challenge in MDP interdiction is that the state space is exponential in the number of state variables. We address this problem by leveraging approximation techniques for factored MDPs. We present a mixed integer linear program formulation of a Stackelberg game model of factored MDP interdiction. We represent the value function as a linear combination of a set of basis functions. For effective basis representation, we make use of a Fourier basis over a Boolean hypercube to represent the value function over the binary state variables. Since the exact Fourier basis representation is still exponential, we generate basis function iteratively while we compute the attacker's optimal policy by approximate factored MDP solution approaches. In doing so, we are able to efficiently represent the exponential state space for value function computations, using an optimized set of Fourier basis functions. However, in solving the MDP interdiction game, we face a second challenge of a super-exponential set of constraints corresponding to alternative evasion plans of the attacker. To tackle this problem, we develop a novel constraint generation algorithm using a combination of linear programming factored MDP solvers and heuristics for attack plan generation.

Our algorithms achieve dramatically improved scalability on realistic MDP problem domains from the IPC and scale up to more than 2^{60} state space sizes compared to the exact interdiction baseline which does not scale beyond toy problem sizes (about 2^8). We also demonstrate empirically that our algorithms achieve near-optimal interdiction decisions in each MDP domain in our experiments.

10.2.2 MDP Initial State Interdiction: Single-Level Optimization

Although our proposed MDP interdiction algorithms are reasonably scalable, some significant scalability limitations continue to prevail. For example, if we want to capture uncertainty about the attacker, we would need to compute an optimal policy for each attacker type, and this becomes intractable quite easily. We propose a novel interdiction model in which, unlike prior work, the defender modifies the initial state of the attacker. The attacker then starts planning from this modified initial state to compute an optimal policy. This is also very general because the previous interdiction approaches can be modeled by adding action-specific preconditions as state variables. We demonstrate that this model achieves much simpler and more scalable interdiction algorithms. Specifically, the attacker's optimal policy computation is independent of the defender's interdiction decision (modification of initial state to a new start state). This results in a single-level optimization problem corresponding to interdiction as compared to the more difficult bi-level optimization problems faced in previous work. We present a baseline initial state interdiction algorithm with factored MDP solution approaches with Fourier basis and formulate the interdiction optimization as an integer linear program. However, we observe that the difficulty of solving the factored MDP grows exponentially in the number of interdependencies among state variables in the underlying Bayesian network. Consequently, we propose model-free reinforcement learning to further improve scalability by directly learning the value function from observations.

10.2.3 Improving Scalability with Reinforcement Learning

We begin with linear action-value Q-function approximation using the Fourier basis set and adapt the traditional Q-learning algorithm to embed an a basis function selection step in between gradient descent iterations. Specifically, we formulate a mixed integer linear program that computes the most important basis function to add to an existing set, based on the largest marginal impact on the Q-function approximation. We then extend the interdiction integer program to work with the Q-function and present an algorithm for state interdiction with linear Q-learning. Extensive experiments demonstrate that this approach achieves dramatically improved scalability. However, the performace still depends on the subset of basis functions chosen for the approximation. To address this limitation, we generalize the reinforcement learning framework to incorporate non-linear Q-function approximation, e.g., neural networks-based Q-function. Since the integer linear program that we developed for interdiction will no longer work with the non-linear approximation, we propose two local search approaches, a) greedy local search by changing one state variable at a time and b) a local linear approximation algorithm which linearizes the Q-function locally and solves an integer linear program in that region. Extensive experiments demonstrate that the non-linear Q-learning approach with local linear approximation achieves the best scalability while ensuring reasonable solution accuracy. We reason that the Q-learning based approaches scale better primarily because these do not need to explicitly solve the MDP and learn the value function directly from observations.

10.2.4 Bayesian Interdiction

So far, in our models, we assumed that the interdiction game has complete information. However, this is often not the case in real-world situations. We model this uncertainty about the attacker (e.g., its capabilities and actions) in a Bayesian game framework to incorporate several possible attacker types (in terms of the initial attack state such that the defender does not have access to the full initial state). We observe that the value function learning is in terms of the full initial state, whether or not observed by the defender. Therefore, all our interdiction algorithms can be extended to the Bayesian game framework by introducing additional variables and constraints specific to an attack type (and averaging over attacker types in case of non-linear Q-learning). Extensive experiments demonstrate a large benefit to the defender from considering Bayesian interdiction compared to the baseline interdiction of a worst-case attack. While the runtime increases in Bayesian interdiction, it is small relative to the actual time it takes to solve the MDP. To summarize, we present a scalable extension of MDP interdiction to include uncertainty about the attacker in a Bayesian game framework.

10.3 Robust Antibody Design as an Interdiction Game

10.3.1 Broadly Binding Antibody: Single-Level Optimization

We begin by proposing an algorithm for broadly binding antibody design against a fixed panel of virus sequences. We quantify antibody-virus binding and stability using scores generated by ROSETTA. However, ROSETTA evaluations are extremely expensive computationally (about an hour for a single antibody-virus pair scoring evaluation). First, we learn a bi-linear model of pairwise amino acid interactions from data generated using ROSETTA binding and stability scores. We then proceed to formulate the antibody design optimization problem as an integer linear program that builds on the coefficients of the amino acid interactions in the bi-linear model. Our algorithm achieves global optimization in the antibody sequence space against a standard HIV virus panel and exhibits 100% breadth of binding as predicted through ROSETTA evaluations. Additionally, it significantly outperforms the state of the art computational protein design algorithms. We further demonstrate that sequences recovered by this method recover known binding motifs of broadly neutralizing anti-HIV antibodies. Finally, our approach is general and can be extended easily to other protein systems. Although our modeled antibodies were not tested in vitro, we predict that these variants would have greatly increased breadth compared to the wild-type antibody. To summarize, the combination of methods from optimization and protein structure modeling allows us to surpass protein design limitations that have been seen up to this point. We predict that if we test these optimal antibodies against the HIV panel they will have greater neutralization breadth compared to existing antibodies.

10.3.2 Game Theoretic Robust Antibody Optimization

Next, we consider the goal in robust antibody design, i.e., the designed antibody should continue to bind and deactivate the virus sequences against escape mutations. We formulate antibody design as a Stackelberg game between the antibody sequence / vaccine designer (leader) and the virus (the follower) that responds by attempting to escape through the minimum number of mutations.

The key challenge is the enormous combinatorial search space corresponding to the antibody and virus sequence binding sites. We propose and evaluate several stochastic local search approaches to optimize the antibody sequence, starting from a native antibody. Our search algorithms incorporate the computationally expensive ROSETTA evaluations. To speed up the search, we use classification learning to predict antibody-virus binding and restrict actual ROSETTA evaluations to the cases in which there is escape prediction. While this helps in the inner loop (virus escape search) of the bi-level optimization, the outer loop (antibody optimization) is still computationally expensive. To address this, we learn the escape costs (minimum number of mutations to escape) using a Poisson regression, as a function of the antibody sequence. This makes the outer loop tractable since the inner-loop evaluations are restricted to only those antibody candidates that are predicted to be robust. Overall, we propose scalable algorithms for antibody design with stochastic local search approaches incorporating classifiers to predict virus escape and Poisson regression to predict escape costs. Our algorithms achieve improved scalability without significantly compromising solution quality. Finally, we exhibit an antibody that is far more robust compared to the native antibody. Specifically, we report an optimized antibody which requires a minimum of 7 mutations for the virus to escape binding to it. The native antibody, on the other hand, fails to bind the virus after a single strategic escape mutation.

10.3.3 Global Solution to the Bi-Level Optimization

While our game-theoretic antibody design framework achieves superior performance, it relies on local search and hence, does not compute a global solution. Moreover, it does not consider the stability score of the antibody-virus complex, where stability of complex is a critical factor in deciding the viability of a designed antibody sequnece. To achieve these objectives, we formulate the bi-level optimization problem (corresponding to the antibody design game) in terms of the combined binding and stability score (denoted as z-score). We leverage the bi-linear pairwise amino acid interaction model to learn the z-scores generated using ROSETTA. Building on this bi-linear model, we formulate the virus escape decision problem as an integer linear program. Typically, it is absolutely not trivial to compute an optimal solution to such bi-level problems with integer variables. Using linear program relaxation and duality, we formulate the antibody design bi-level problem as a compact mixed integer linear program. We further demonstrate with a proof that our compact formulation computes the optimal integer solution despite the relaxation and duality transformations. Moreover, we can efficiently solve our compact optimization problem using state of the art solvers, e.g., CPLEX. The compact formulation also allows us to incorporate a panel of viruses into the decision problem. Therefore, our algorithm performs efficient global optimization in the antibody space, compared to state of the art local search approaches, which may only achieve limited exploration of the search space. Extensive experiments demonstrate that our algorithm outperforms our previous algorithms in terms of robustness to escape mutations evaluated as predicted z-scores.

10.4 Future Work

In this section, we outline some of the limitations of our algorithmic approaches and provide directions for future work.

10.4.1 Randomized Strategy Commitment

In our algorithms, we consider problems in which the defender deploys deterministic mitigation strategies. However, recent advances in Stackelberg games for security involve randomized strategies in case of the defender and demonstrate that such randomization leads to even higher utilities. Considering mixed strategies poses several algorithmic challenges in the interdiction framework and is definitely an interesting direction for future work.

10.4.2 Partial Observability

MDPs provide an efficient modeling framework to capture the attacker's behavior in plan interdiction. In our algorithms, we assume full observability of the MDP state space. However, in many real-world situations, the actual state may not be observed directly. Instead, a probability distribution is maintained over the set of possible states, based on a set of observations and the observation probabilities. This partially observable MDP (POMDP) framework is general enough to model a variety of real-world sequential decision processes. The solution to a POMDP computes the optimal action for each possible belief over the set of states. Generalizing our MDP interdiction games to incorporate partial observability is another interesting direction for future research.

10.4.3 Multiple Defenders

An interesting future direction is to develop algorithms to incorporate multiple defenders in the decision making process. Such a scenario is common in several real-world situations, e.g., power grids, transporation networks and even physical security forces that often need to coordinate decision making to collectively defend against the attacker(s). In decentralized network interdiction games, multiple agents with differing objectives focus on interdicting parts of a shared network. This modeling framework is also relevant in drug design when multiple antibodies / treatment therapies / drugs act in combination against an external infection.

10.4.4 Repeated Games

The Stackelberg game model is undoubtedly a natural fit that capture the interactions between the defender and attacker in physical security as well as cybersecurity and immunology as we have discussed in this dissertation. At the same time, in certain real-world situations, the players in the game interact repeatedly over time and a repeated game model may be more accurate. This applies to the plan interdiction problem as well as specific applications in drug design. In plan interdiction, an important research direction is that of dynamic interdiction, where the objectives of the players can change over time, and also, each player observes and responds to the other player's strategy. A relevant modeling framework is that of stochastic games which generalize both MDPs and repeated games [78] and factored representations have also been studied [175]. Along similar lines, a critically important future research direction in immunology is to model antibody and virus co-evolution in a repeated game framework.

10.4.5 Challenges in the Antibody Design Application

We now summarize some of the limitations of our approaches and future directions in the antibody design application.

Appropriate Cost Function for Virus Escape

In our virus escape model, we use single point mutations (protein sequence edit distance) as a measure of viral escape difficulty. A more biologically meaningful measure is the number of neucleotide / mRNA mutations involved. For example, a single amino acid change is achieved through a specific sequence of neucleotide mutations.

Learning Rosetta Scores

A major challenge is to learn the ROSETTA binding and stability scores with sufficient accuracy from data. While our (linear and non-linear) learning algorithms do achieve reasonable accuracy, there is plenty of room for improvement. A promising future direction would be to leverage recent advances in deep neural networks to accurately learn the ROSETTA energy scores. This will address the issue of computationally expensive ROSETTA evaluations. Moreover, bi-level optimization algorithms that build on more accurate energy prediction models can potentially achieve better quality solutions (although our antibody design algorithms are extremely efficient despite the moderately accurate predictions using the bi-linear model).

Negative Design

The optimization framework in our models can be adapted to solve the problem of negative design which is very important in computational protein design. At a very high-level, in negative design, a protein sequence needs to maintain binding to one set of protein sequences and simultaneouly needs to escape binding from a separate set of protein sequences.

Challenges in Extending the Antibody design Algorithms to Wet Lab Experiments

We conjecture that if we experimentally test our designed optimal antibodies against the HIV panel those will have greater neutralization breadth compared to existing antibodies. However, at present, there are several challenges before we can proceed to wet lab experiments. First, insertions and deletions in the antibody protein sequence space make experimental modeling very difficult. Homology modeling of viral proteins is also a critical problem. To evaluate our designed antibodies, we need to use a set of viral proteins that are already crystallized. Finally, modeling only a small binding site is also an issue. Ideally, the algorithms model the entire antibody-virus interface.

Finally, it is important to emphasize that the proposed research has enormous potential to bring about remarkable changes in the way vaccines are designed against rapidly mutating pathogens e.g., HIV and influenza. Furthermore, the computational techniques in our research can be directly translated to the more general problems of robust drug design and treatment regimens.

BIBLIOGRAPHY

[1] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, (1):41–50, 2003.

[2] Daniel A Menascé et al. The insider threat security architecture: A framework for an integrated, inseparable, and uninterrupted self-protection mechanism. In *2009 International Conference on Computational Science and Engineering*, pages 244–251. IEEE, 2009.

[3] Qian Chen, Sherif Abdelwahed, and Abdelkarim Erradi. A model-based approach to self-protection in computing system. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 16. ACM, 2013.

[4] Firas B Alomari and Daniel A Menascé. Self-protecting and self-optimizing database systems: Implementation and experimental evaluation. In *Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference*, page 18. ACM, 2013.

[5] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordóñez. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces*, 40(4):267–290, 2010.

[6] Daniel Bilar, George Cybenko, and John Murphy. Adversarial dynamics: the conficker case study. In *Moving Target Defense II*, pages 41–71. Springer, 2013.

[7] Daniel Lowd and Christopher Meek. Good word attacks on statistical spam filters. In *CEAS*, volume 2005, 2005.

[8] Prahlad Fogla and Wenke Lee. Evading network anomaly detection systems: formal

reasoning and practical techniques. In *Proceedings of the 13th ACM conference on Computer and communications security*, pages 59–68. ACM, 2006.

[9] Long Cheng, Fang Liu, and Danfeng Yao. Enterprise data breach: causes, challenges, prevention, and future directions. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 7(5):e1211, 2017.

[10] Lin Chen and Jean Leneutre. A game theoretical framework on intrusion detection in heterogeneous networks. *IEEE Transactions on Information Forensics and Security*, 4(2):165–178, 2009.

[11] Tansu Alpcan and Tamer Başar. *Network security: A decision and game-theoretic approach*. Cambridge University Press, 2010.

[12] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. In *IJCAI*, pages 5494–5501, 2018.

[13] James Pita, Manish Jain, Janusz Marecki, Fernando Ordóñez, Christopher Portway, Milind Tambe, Craig Western, Praveen Paruchuri, and Sarit Kraus. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems: industrial track*, pages 125–132. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[14] Carol Koetke. One size doesn't fit all. *TECHNOS*, 8(2):20–26, 1999.

[15] Fei Fang, Thanh Hong Nguyen, Rob Pickles, Wai Y Lam, Gopalasamy R Clements, Bo An, Amandeep Singh, Milind Tambe, Andrew Lemieux, et al. Deploying paws: Field optimization of the protection assistant for wildlife security. In *AAAI*, pages 3966–3973, 2016.

[16] UNAIDS. Hiv fact sheet, 2013.

[17] Jinghe Huang, Byong H Kang, Marie Pancera, Jeong Hyun Lee, Tommy Tong, Yu Feng, Hiromi Imamichi, Ivelin S Georgiev, Gwo-Yu Chuang, Aliaksandr Druz, et al. Broad and potent hiv-1 neutralization by a human antibody that binds the gp41–gp120 interface. *Nature*, 515(7525):138, 2014.

[18] José M Cuevas, Ron Geller, Raquel Garijo, José López-Aldeguer, and Rafael Sanjuán. Extremely high mutation rate of hiv-1 in vivo. *PLoS biology*, 13(9):e1002251, 2015.

[19] Joshua Letchford and Yevgeniy Vorobeychik. Optimal interdiction of attack plans. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 199–206. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[20] Jeffrey J Gray, Stewart Moughon, Chu Wang, Ora Schueler-Furman, Brian Kuhlman, Carol A Rohl, and David Baker. Protein–protein docking with simultaneous optimization of rigid-body displacement and side-chain conformations. *Journal of molecular biology*, 331(1):281–299, 2003.

[21] Alexander M Sevy, Tim M Jacobs, James E Crowe Jr, and Jens Meiler. Design of protein multi-specificity using an independent sequence search reduces the barrier to low energy sequences. *PLOS Comput Biol*, 11(7):e1004300, 2015.

[22] Drew Fudenberg and Jean Tirole. Game theory mit press. *Cambridge, MA*, page 86, 1991.

[23] Martin J Osborne and Ariel Rubinstein. *A course in game theory*. MIT press, 1994.

[24] Heinrich Von Stackelberg. *Market structure and equilibrium*. Springer Science & Business Media, 2010.

[25] Zhengyu Yin, Dmytro Korzhyk, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: Interchangeability, equivalence, and uniqueness. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 1139–1146. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[26] Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit 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.

[27] Manish Jain, James Pita, Milind Tambe, Fernando Ordóñez, Praveen Paruchuri, and Sarit Kraus. Bayesian stackelberg games and their application for security at los angeles international airport. *SIGecom Exch.*, 7:10:1–10:3, June 2008.

[28] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[29] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90. ACM, 2006.

[30] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[31] Christopher Kiekintveld, Janusz Marecki, and Milind Tambe. Approximation methods for infinite bayesian stackelberg games: Modeling distributional payoff uncertainty. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1005–1012. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[32] Jason Tsai, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, and Shyamsunder Rathi. Iris-a tool for strategic security allocation in transportation networks. 2009.

[33] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 13–20. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

[34] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *IJCAI*, pages 2589–2595, 2015.

[35] Chao Zhang, Arunesh Sinha, and Milind Tambe. Keeping pace with criminals: Designing patrol allocation against adaptive opportunistic criminals. In *Proceedings of the 2015 international conference on Autonomous agents and multiagent systems*, pages 1351–1359. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[36] Zhengyu Yin, Albert Xin Jiang, Matthew Paul Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *IAAI*, 2012.

[37] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[38] Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1, pages 560–564. IEEE, 1995.

[39] Christian Fritz and Sheila A McIlraith. Computing robust plans in continuous domains. In *ICAPS*, 2009.

[40] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[41] Stuart J Russell and Peter Norvig. *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016.

[42] Yixin Chen, Benjamin W Wah, and Chihwei Hsu. Temporal planning using subgoal partitioning and resolution in sgplan. *Journal of Artificial Intelligence Research*, 26:323–369, 2006.

[43] Martin L Puterman. Markov decision processes: Discrete stochastic dynamic programming. 1994.

[44] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored mdps. *Journal of Artificial Intelligence Research*, 19:399–468, 2003.

[45] Daphne Koller and Ronald Parr. Computing factored value functions for policies in structured mdps. In *IJCAI*, volume 99, pages 1332–1339, 1999.

[46] Carlos Ernesto Guestrin. *Planning under uncertainty in complex structured environments*. PhD thesis, Stanford University, 2003.

[47] Daphne Koller and Ronald Parr. Policy iteration for factored mdps. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 326–334. Morgan Kaufmann Publishers Inc., 2000.

[48] Csaba Szepesvári. Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103, 2010.

[49] Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

[50] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

[51] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[52] JN Tsitsiklis and B Van Roy. An analysis of temporal-difference learning with function approximationtechnical. Technical report, Report LIDS-P-2322). Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1996.

[53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[54] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.

[55] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[56] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[57] Javier Salmeron, Kevin Wood, and Ross Baldick. Worst-case interdiction analysis of large-scale electric power grids. *IEEE Transactions on power systems*, 24(1):96–104, 2009.

[58] Alan W McMasters and Thomas M Mustin. Optimal interdiction of a supply network. *Naval Research Logistics Quarterly*, 17(3):261–268, 1970.

[59] PM Ghare, Douglas C Montgomery, and WC Turner. Optimal interdiction policy for a flow network. *Naval Research Logistics Quarterly*, 18(1):37–45, 1971.

[60] Laura P Swiler, Cynthia Phillips, David Ellis, and Stefan Chakerian. Computer-attack graph generation tool. In *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX'01. Proceedings*, volume 2, pages 307–321. IEEE, 2001.

[61] Oleg Sheyner, Joshua Haines, Somesh Jha, Richard Lippmann, and Jeannette M Wing. Automated generation and analysis of attack graphs. In *null*, page 273. IEEE, 2002.

[62] Dan Zerkle and Karl N Levitt. Netkuang-a multi-host configuration vulnerability checker. In *USENIX Security Symposium*, 1996.

[63] Ronald W Ritchey and Paul Ammann. Using model checking to analyze network vulnerabilities. In *Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on*, pages 156–165. IEEE, 2000.

[64] Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 217–224. ACM, 2002.

[65] Mark Boddy, Johnathan Gohde, Tom Haigh, and Steven Harp. Course of action generation for cyber security using classical planning. In *International Conference on Automated Planning and Scheduling*, pages 12–21, 2005.

[66] Jorge Lucangeli Obes, Carlos Sarraute, and Gerardo Richarte. Attack planning in the real world. *arXiv preprint arXiv:1306.4044*, 2013.

[67] Nayot Poolsappasit, Rinku Dewri, and Indrajit Ray. Dynamic security risk management using bayesian attack graphs. *IEEE Transactions on Dependable and Secure Computing*, 9(1):61–74, 2012.

[68] Stefano Bistarelli, Marco Dall'Aglio, and Pamela Peretti. Strategic games on defense trees. In *International Workshop on Formal Aspects in Security and Trust*, pages 1–15. Springer, 2006.

[69] Saman A Zonouz, Himanshu Khurana, William H Sanders, and Timothy M Yardley. Rre: A game-theoretic intrusion response and recovery engine. *IEEE Transactions on Parallel and Distributed Systems*, 25(2):395–406, 2014.

[70] Cynthia A Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785. ACM, 1993.

[71] Gerald G Brown, W Matthew Carlyle, Robert C Harney, Eric M Skroch, and R Kevin Wood. Interdicting a nuclear-weapons project. *Operations Research*, 57(4):866–877, 2009.

[72] James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen, and Erin Steigerwald. Guards: game theoretic security allocation on a national scale. In *The 10th Interna-*

tional Conference on Autonomous Agents and Multiagent Systems-Volume 1, pages 37–44. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[73] Yevgeniy Vorobeychik and Satinder P Singh. Computing stackelberg equilibria in discounted stochastic games. In *AAAI*, 2012.

[74] Albert Xin Jiang, Zhengyu Yin, Chao Zhang, Milind Tambe, and Sarit Kraus. Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 207–214. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[75] Nicola Basilico, Nicola Gatti, and Francesco Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184:78–123, 2012.

[76] Steven Willmott, Julian Richardson, Alan Bundy, and John Levine. An adversarial planning approach to go. In *International Conference on Computers and Games*, pages 93–112. Springer, 1998.

[77] Raghuram Ramanujan, Ashish Sabharwal, and Bart Selman. On adversarial search spaces and sampling-based planning. In *ICAPS*, volume 10, pages 242–245, 2010.

[78] Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

[79] Linling He and Jiang Zhu. Computational tools for epitope vaccine design and evaluation. *Current opinion in virology*, 11:103–112, 2015.

[80] Brian Kuhlman, Gautam Dantas, Gregory C Ireton, Gabriele Varani, Barry L Stod-

dard, and David Baker. Design of a novel globular protein fold with atomic-level accuracy. *science*, 302(5649):1364–1368, 2003.

[81] Bassil I Dahiyat and Stephen L Mayo. De novo protein design: fully automated sequence selection. *Science*, 278(5335):82–87, 1997.

[82] Jordan R Willis, Gopal Sapparapu, Sasha Murrell, Jean-Philippe Julien, Vidisha Singh, Hannah G King, Yan Xia, Jennifer A Pickens, Celia C LaBranche, James C Slaughter, et al. Redesigned hiv antibodies exhibit enhanced neutralizing potency and breadth. *The Journal of clinical investigation*, 125(6):2523–2531, 2015.

[83] Sarel J Fleishman, Timothy A Whitehead, Damian C Ekiert, Cyrille Dreyfus, Jacob E Corn, Eva-Maria Strauch, Ian A Wilson, and David Baker. Computational design of proteins targeting the conserved stem region of influenza hemagglutinin. *Science*, 332(6031):816–821, 2011.

[84] Eva-Maria Strauch, Steffen M Bernard, David La, Alan J Bohn, Peter S Lee, Caitlin E Anderson, Travis Nieusma, Carly A Holstein, Natalie K Garcia, Kathryn A Hooper, et al. Computational design of trimeric influenza-neutralizing proteins targeting the hemagglutinin receptor binding site. *Nature biotechnology*, 35(7):667, 2017.

[85] Neil P King, William Sheffler, Michael R Sawaya, Breanna S Vollmar, John P Sumida, Ingemar André, Tamir Gonen, Todd O Yeates, and David Baker. Computational design of self-assembling protein nanomaterials with atomic level accuracy. *Science*, 336(6085):1171–1174, 2012.

[86] M Rosenberg and A Goldblum. Computational protein design: a novel path to future protein drugs. *Current pharmaceutical design*, 12(31):3973–3997, 2006.

[87] Shaun M Lippow, K Dane Wittrup, and Bruce Tidor. Computational design of

antibody-affinity improvement beyond in vivo maturation. *Nature biotechnology*, 25(10):1171–1176, 2007.

[88] John Karanicolas and Brian Kuhlman. Computational design of affinity and specificity at protein–protein interfaces. *Current opinion in structural biology*, 19(4):458–463, 2009.

[89] Bruno E Correia, John T Bates, Rebecca J Loomis, Gretchen Baneyx, Chris Carrico, Joseph G Jardine, Peter Rupert, Colin Correnti, Oleksandr Kalyuzhniy, Vinayak Vittal, et al. Proof of principle for epitope-focused vaccine design. *Nature*, 507(7491):201–206, 2014.

[90] Alexander M Sevy and Jens Meiler. Antibodies: Computer-aided prediction of structure and design of function. *Microbiology spectrum*, 2(6), 2014.

[91] Johan Desmet, Marc De Maeyer, Bart Hazes, and Ignace Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Nature*, 356(6369):539, 1992.

[92] Julia M Shifman and Stephen L Mayo. Modulating calmodulin binding specificity through computational protein design. *Journal of molecular biology*, 323(3):417–423, 2002.

[93] Jordan R Willis, Bryan S Briney, Samuel L DeLuca, James E Crowe Jr, and Jens Meiler. Human germline antibody gene segments encode polyspecific antibodies. *PLoS computational biology*, 9(4):e1003045, 2013.

[94] Gurkan Guntas, Ryan A Hallett, Seth P Zimmerman, Tishan Williams, Hayretin Yumerefendi, James E Bear, and Brian Kuhlman. Engineering an improved light-induced dimer (ilid) for controlling the localization and activity of signaling proteins. *Proceedings of the National Academy of Sciences*, 112(1):112–117, 2015.

[95] Stanley C Howell, Krishna Kishore Inampudi, Doyle P Bean, and Corey J Wilson. Understanding thermal adaptation of enzymes through the multistate rational design and stability prediction of 100 adenylate kinases. *Structure*, 22(2):218–229, 2014.

[96] James A Davey and Roberto A Chica. Improving the accuracy of protein stability predictions with multistate design using a variety of backbone ensembles. *Proteins: Structure, Function, and Bioinformatics*, 82(5):771–784, 2014.

[97] Steven M Lewis, Xiufeng Wu, Anna Pustilnik, Arlene Sereno, Flora Huang, Heather L Rick, Gurkan Guntas, Andrew Leaver-Fay, Eric M Smith, Carolyn Ho, et al. Generation of bispecific igg antibodies by structure-based design of an orthogonal fab interface. *Nature biotechnology*, 32(2):191, 2014.

[98] James J Havranek and Pehr B Harbury. Automated design of specificity in molecular recognition. *Nature Structural and Molecular Biology*, 10(1):45, 2003.

[99] Andrew Leaver-Fay, Ron Jacak, P Benjamin Stranges, and Brian Kuhlman. A generic program for multistate protein design. *PloS one*, 6(7):e20937, 2011.

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

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

[102] Jingzhou Liu, Beth F. Kochin, Yonas I. Tekle, and Alison 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.

[103] Sheng-He Huang, Wensheng Zhou, Ambrose Jong, and Huan Qi. Game theory

models for infectious diseases. In *Frontiers in the Convergence of Bioscience and Information Technologies*, pages 265–269, 2007.

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

[105] Richard H Lathrop, Nicholas R Steffen, Miriam P Raphael, Sophia Deeds-Rubin, Michael J Pazzani, Paul J Cimoch, Darryl M See, and Jeremiah G Tilles. Knowledge-based avoidance of drug-resistant hiv mutants. *AI Magazine*, 20(1):13, 1999.

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

[107] Pablo Hernandez-Leal, Lindsey Fiedler-Cameras, Alma Rios-Flores, Jesús A González, L Enrique Sucar, and Sta María Tonantzintla. Contrasting temporal bayesian network models for analyzing hiv mutations.

[108] Lothar Richter, Regina Augustin, and Stefan Kramer. Finding relational associations in hiv resistance mutation data. In *Inductive Logic Programming*, pages 202–208. Springer, 2010.

[109] Andrew L Ferguson, Jaclyn K Mann, Saleha Omarjee, Thumbi Ndung'u, Bruce D Walker, and Arup K Chakraborty. Translating hiv sequences into quantitative fitness landscapes predicts viral vulnerabilities for rational immunogen design. *Immunity*, 38(3):606–617, 2013.

[110] Faruck Morcos, Andrea Pagnani, Bryan Lunt, Arianna Bertolino, Debora S Marks, Chris Sander, Riccardo Zecchina, José N Onuchic, Terence Hwa, and Martin Weigt. Direct-coupling analysis of residue coevolution captures native contacts across many

protein families. *Proceedings of the National Academy of Sciences*, 108(49):E1293–E1301, 2011.

[111] David T Jones, Daniel WA Buchan, Domenico Cozzetto, and Massimiliano Pontil. Psicov: precise structural contact prediction using sparse inverse covariance estimation on large multiple sequence alignments. *Bioinformatics*, 28(2):184–190, 2012.

[112] Hetunandan Kamisetty, Sergey Ovchinnikov, and David Baker. Assessing the utility of coevolution-based residue–residue contact predictions in a sequence- and structure-rich era. *Proceedings of the National Academy of Sciences*, 110(39):15674–15679, 2013.

[113] Sivaraman Balakrishnan, Hetunandan Kamisetty, Jaime G Carbonell, Su-In Lee, and Christopher James Langmead. Learning generative models for protein fold families. *Proteins: Structure, Function, and Bioinformatics*, 79(4):1061–1078, 2011.

[114] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. Graphical models of residue coupling in protein families. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 5(2):183–197, 2008.

[115] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. Protein design by sampling an undirected graphical model of residue constraints. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 6(3):506–516, 2009.

[116] Hetunandan Kamisetty, Eric P Xing, and Christopher J Langmead. Approximating correlated equilibria using relaxations on the marginal polytope. 2011.

[117] John Thomas, Naren Ramakrishnan, and Chris Bailey-Kellogg. Graphical models of protein–protein interaction specificity from correlated mutations and interaction data. *Proteins: Structure, Function, and Bioinformatics*, 76(4):911–929, 2009.

[118] Gwo-Yu Chuang, Priyamvada Acharya, Stephen D Schmidt, Yongping Yang, Mark K Louder, Tongqing Zhou, Young Do Kwon, Marie Pancera, Robert T Bailer, Nicole A Doria-Rose, et al. Residue-level prediction of hiv-1 antibody epitopes based on neutralization of diverse viral strains. *Journal of virology*, 87(18):10047–10058, 2013.

[119] Mark C Evans, Pham Phung, Agnes C Paquet, Anvi Parikh, Christos J Petropoulos, Terri Wrin, and Mojgan Haddad. Predicting hiv-1 broadly neutralizing antibody epitope networks using neutralization titers and a novel computational method. *BMC bioinformatics*, 15(1):1, 2014.

[120] Anna Feldmann and Nico Pfeifer. From predicting to analyzing hiv-1 resistance to broadly neutralizing antibodies. Technical report, PeerJ PrePrints, 2015.

[121] Haidong Wang, Eran Segal, Asa Ben-Hur, Daphne Koller, and Douglas L Brutlag. Identifying protein-protein interaction sites on a genome-wide scale. In *Advances in neural information processing systems*, pages 1465–1472, 2004.

[122] Hetunandan Kamisetty, Bornika Ghosh, Christopher James Langmead, and Chris Bailey-Kellogg. Learning sequence determinants of protein: Protein interaction specificity with sparse graphical models. In *Research in Computational Molecular Biology*, pages 129–143. Springer, 2014.

[123] Manish Jain, James Pita, Milind Tambe, Fernando Ordónez, Praveen Paruchuri, and Sarit Kraus. Bayesian stackelberg games and their application for security at los angeles international airport. *ACM SIGecom Exchanges*, 7(2):10, 2008.

[124] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. Stackelberg vs. nash in security games: An extended investigation of interchangeability, equivalence, and uniqueness. *Journal of Artificial Intelligence Research*, 41:297–327, 2011.

[125] Brenda Ng, Carol Meyers, Kofi Boakye, and John J Nitao. Towards applying interactive pomdps to real-world adversary modeling. In *IAAI*, 2010.

[126] Leanid Krautsevich, Fabio Martinelli, and Artsiom Yautsiukhin. Towards modelling adaptive attacker's behaviour. In *International Symposium on Foundations and Practice of Security*, pages 357–364. Springer, 2012.

[127] Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1):49–107, 2000.

[128] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11(1):94, 1999.

[129] Thomas G Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *J. Artif. Intell. Res.(JAIR)*, 13:227–303, 2000.

[130] Robert St-Aubin, Jesse Hoey, and Craig Boutilier. Apricodd: Approximate policy construction using decision diagrams. In *NIPS*, pages 1089–1095, 2000.

[131] Carlos Guestrin, Daphne Koller, and Ronald Parr. Max-norm projections for factored mdps. In *IJCAI*, volume 1, pages 673–682, 2001.

[132] Ryan O'Donnell. Some topics in analysis of boolean functions. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 569–578. ACM, 2008.

[133] Swetasudha Panda and Yevgeniy Vorobeychik. Near-optimal interdiction of factored mdps. In *Conference on Uncertainty in Artificial Intelligence*, 2017.

[134] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez,

Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[135] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[136] Joint United Nations Programme on HIV/AIDS et al. Fact sheet—latest statistics on the status of the aids epidemic, 2017.

[137] UNAIDS. Fact sheet-latest statistics on the status of the aids epidemic, 2016.

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

[139] CDC. Seasonal influenza, more information, 2018. https://www.cdc.gov/flu/about/qa/disease.htm.

[140] Rebecca F Alford, Andrew Leaver-Fay, Jeliazko R Jeliazkov, Matthew J O'Meara, Frank P DiMaio, Hahnbeom Park, Maxim V Shapovalov, P Douglas Renfrew, Vikram K Mulligan, Kalli Kappel, et al. The rosetta all-atom energy function for macromolecular modeling and design. *Journal of chemical theory and computation*, 13(6):3031–3048, 2017.

[141] Ivelin S Georgiev, Nicole A Doria-Rose, Tongqing Zhou, Young Do Kwon, Ryan P Staupe, Stephanie Moquin, Gwo-Yu Chuang, Mark K Louder, Stephen D Schmidt, Han R Altae-Tran, et al. Delineating antibody recognition in polyclonal sera from patterns of hiv-1 isolate neutralization. *Science*, 340(6133):751–756, 2013.

[142] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[143] Hetunandan Kamisetty, Bornika Ghosh, Christopher James Langmead, and Chris Bailey-Kellogg. Learning sequence determinants of protein: Protein interaction specificity with sparse graphical models. *Journal of Computational Biology*, 22(6):474–486, 2015.

[144] Tongqing Zhou, Ivelin Georgiev, Xueling Wu, Zhi-Yong Yang, Kaifan Dai, Andrés Finzi, Young Do Kwon, Johannes F Scheid, Wei Shi, Ling Xu, et al. Structural basis for broad and potent neutralization of hiv-1 by antibody vrc01. *Science*, 329(5993):811–817, 2010.

[145] Ron Diskin, Johannes F Scheid, Paola M Marcovecchio, Anthony P West, Florian Klein, Han Gao, Priyanthi NP Gnanapragasam, Alexander Abadir, Michael S Seaman, Michel C Nussenzweig, et al. Increasing the potency and breadth of an hiv antibody by using structure-based rational design. *Science*, page 1206727, 2011.

[146] Florian Klein, Ron Diskin, Johannes F Scheid, Christian Gaebler, Hugo Mouquet, Ivelin S Georgiev, Marie Pancera, Tongqing Zhou, Reha-Baris Incesu, Brooks Zhongzheng Fu, et al. Somatic mutations of the immunoglobulin framework are generally required for broad and potent hiv-1 neutralization. *Cell*, 153(1):126–138, 2013.

[147] Johannes F Scheid, Hugo Mouquet, Beatrix Ueberheide, Ron Diskin, Florian Klein, Thiago YK Oliveira, John Pietzsch, David Fenyo, Alexander Abadir, Klara Velinzon, et al. Sequence and structural convergence of broad and potent hiv antibodies that mimic cd4 binding. *Science*, 333(6049):1633–1637, 2011.

[148] Andrew Leaver-Fay, Matthew J O'meara, Mike Tyka, Ron Jacak, Yifan Song, Elizabeth H Kellogg, James Thompson, Ian W Davis, Roland A Pache, Sergey Lyskov, et al. Scientific benchmarks for guiding macromolecular energy function improvement. In *Methods in enzymology*, volume 523, pages 109–143. Elsevier, 2013.

[149] Brian J Bender, Alberto Cisneros III, Amanda M Duran, Jessica A Finn, Darwin Fu, Alyssa D Lokits, Benjamin K Mueller, Amandeep K Sangha, Marion F Sauer, Alexander M Sevy, et al. Protocols for molecular modeling with rosetta3 and rosettascripts. *Biochemistry*, 55(34):4748–4763, 2016.

[150] Tongqing Zhou, Jiang Zhu, Xueling Wu, Stephanie Moquin, Baoshan Zhang, Priyamvada Acharya, Ivelin S Georgiev, Han R Altae-Tran, Gwo-Yu Chuang, M Gordon Joyce, et al. Multidonor analysis reveals structural elements, genetic determinants, and maturation pathway for hiv-1 neutralization by vrc01-class antibodies. *Immunity*, 39(2):245–258, 2013.

[151] Xueling Wu, Zhenhai Zhang, Chaim A Schramm, M Gordon Joyce, Young Do Kwon, Tongqing Zhou, Zizhang Sheng, Baoshan Zhang, Sijy O'Dell, Krisha McKee, et al. Maturation and diversity of the vrc01-antibody lineage over 15 years of chronic hiv-1 infection. *Cell*, 161(3):470–485, 2015.

[152] Albin Sandelin and Wyeth W Wasserman. Constrained binding site diversity within families of transcription factors enhances pattern discovery bioinformatics. *Journal of molecular biology*, 338(2):207–215, 2004.

[153] Benjamin D Allen, Alex Nisthal, and Stephen L Mayo. Experimental library screening demonstrates the successful application of computational protein design to large structural ensembles. *Proceedings of the National Academy of Sciences*, 107(46):19838–19843, 2010.

[154] Andrew Leaver-Fay, Karen J Froning, Shane Atwell, Hector Aldaz, Anna Pustilnik, Frances Lu, Flora Huang, Richard Yuan, Saleema Hassanali, Aaron K Chamberlain, et al. Computationally designed bispecific antibodies using negative state repertoires. *Structure*, 24(4):641–651, 2016.

[155] Xueling Wu, Zhi-Yong Yang, Yuxing Li, Carl-Magnus Hogerkorp, William R

Schief, Michael S Seaman, Tongqing Zhou, Stephen D Schmidt, Lan Wu, Ling Xu, et al. Rational design of envelope identifies broadly neutralizing human monoclonal antibodies to hiv-1. *Science*, 329(5993):856–861, 2010.

[156] Jinghe Huang, Gilad Ofek, Leo Laub, Mark K Louder, Nicole A Doria-Rose, Nancy S Longo, Hiromi Imamichi, Robert T Bailer, Bimal Chakrabarti, Shailendra K Sharma, et al. Broad and potent neutralization of hiv-1 by a gp41-specific human antibody. *Nature*, 491(7424):406, 2012.

[157] Joseph Jardine, Jean-Philippe Julien, Sergey Menis, Takayuki Ota, Oleksandr Kalyuzhniy, Andrew McGuire, Devin Sok, Po-Ssu Huang, Skye MacPherson, Meaghan Jones, Travis Nieusma, John Mathison, David Baker, Andrew B. Ward, Dennis R. Burton, Leonidas Stamatatos, David Nemazee, Ian A. Wilson, and William R. Schief. Rational hiv immunogen design to target specific germline b cell receptors. *Science*, 340(6133):711–716, 2013.

[158] Gilad Ofek, F. Javier Guenaga, William R. Schief, Jeff Skinner, David Baker, Richard Wyatt, and Peter D. Kwong. Elicitation of structure-specific antibodies by epitope scaffolds. *Proceedings of the National Academy of Sciences*, 107(42):17880–17887, 2010.

[159] Bruno Correia, John Bates, Rebecca J Loomis, Gretchen Baneyx, Chris Carrico, Joseph Jardine, Peter Rupert, Colin Correnti, Oleksandr Kalyuzhniy, Vinayak Vittal, Mary J Connell, Eric Stevens, Alexandria Schroeter, Man Chen, Skye Macpherson, Andreia Serra, Yumiko Adachi, Margaret A Holmes, Yuxing Li, and William R Schief. Proof of principle for epitope-focused vaccine design. 507, 02 2014.

[160] Schrödinger, LLC. The PyMOL molecular graphics system, version 1.8. November 2015.

[161] M.A. Larkin, G. Blackshields, N.P. Brown, R. Chenna, P.A. McGettigan,

H. McWilliam, F. Valentin, I.M. Wallace, A. Wilm, R. Lopez, J.D. Thompson, T.J. Gibson, and D.G. Higgins. Clustal w and clustal x version 2.0. *Bioinformatics*, 23(21):2947–2948, 2007.

[162] Steven A Combs, Samuel L. DeLuca, Stephanie H Deluca, Gordon Lemmon, David P Nannemann, Elizabeth Dong Nguyen, Jordan R Willis, Jonathan H. Sheehan, and Jens Meiler. Small-molecule ligand docking into comparative models with rosetta. *Nature Protocols*, 8:1277–1298, 2013.

[163] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[164] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[165] Colin A. Smith and Tanja Kortemme. Backrub-like backbone simulation recapitulates natural protein conformational variability and improves mutant side-chain prediction. *Journal of molecular biology*, 380 4:742–56, 2008.

[166] Alexander M Sevy, Swetasudha Panda, James E Crowe Jr, Jens Meiler, and Yevgeniy Vorobeychik. Integrating linear optimization with structural modeling to increase hiv neutralization breadth. *PLoS computational biology*, 14(2):e1005999, 2018.

[167] Martin A. Nowak and Robert May. *Virus Dynamics: Mathematical Principles of Immunology and Virology*. Oxford University Press, 2001.

[168] Holger H. Hoos and Thomas Stutzle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann, 2004.

[169] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[170] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[171] Swetasudha Panda and Yevgeniy Vorobeychik. Stackelberg games for vaccine design. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1391–1399. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[172] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[173] Steven A Combs, Samuel L DeLuca, Stephanie H DeLuca, Gordon H Lemmon, David P Nannemann, Elizabeth D Nguyen, Jordan R Willis, Jonathan H Sheehan, and Jens Meiler. Small-molecule ligand docking into comparative models with rosetta. *Nature protocols*, 8(7):1277, 2013.

[174] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[175] Michail G Lagoudakis and Ronald Parr. Learning in zero-sum team markov games using factored value functions. In *Advances in Neural Information Processing Systems*, pages 1659–1666, 2003.