THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF CHEMICAL ENGINEERING

QUICK OPTMAVEN: AN EFFICIENT COMPUTATIONAL FRAMEWORK FOR THE DE NOVO DESIGN OF FULLY HUMAN MONOCLONAL ANTIBODIES

MATTHEW FREDERICK ALLAN
SPRING 2018

A thesis
submitted in partial fulfillment
of the requirements
for a baccalaureate degree
in Biochemistry and Molecular Biology
with honors in Chemical Engineering

Reviewed and approved* by the following:

Costas D. Maranas
Donald B. Broughton Professor of Chemical Engineering
Thesis Supervisor

Andrew Zydney
Distinguished Professor of Chemical Engineering
Honors Adviser

* Signatures are on file in the Schreyer Honors College.

ABSTRACT

Monoclonal antibodies (mAbs) have become one of the most promising classes of therapeutics. Currently, dozens of mAbs are FDA-approved for autoimmune disorders, cancers, and infectious diseases. While there exist robust laboratory methods for designing mAbs for new antigens, these expensive and time-consuming (i.e. 3 – 6 months) methods are incapable of developing therapeutic antibodies rapidly during an epidemic. Potentially, computational methods of engineering mAbs could expedite this process. The first software capable of designing antibody variable domains *de novo*—OptMAVEn—was published in 2014. Despite designing three mAbs that bound a dodecapeptide antigen, OptMAVEn proved too time- and storage-intensive to design mAbs for Zika during the 2015 – 2016 epidemic.

Here, we present Quick OptMAVEn, a new implementation of OptMAVEn. We show that Quick OptMAVEn can design variable domains of equivalent quality in 74% less time using 84% less disk storage, relative to OptMAVEn. Furthermore, we have used Quick OptMAVEn to design 50 antibodies for Zika, 9 of which are predicted to bind the antigen with greater affinity than an antibody isolated from a human patient. Quick OptMAVEn achieves better performance by using more efficient algorithms, more compact representations of antigen structures, and a novel *k*-means clustering step. We plan to create a web server to share Quick OptMAVEn with other labs.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Ratul Chowdhury for working closely with me throughout the entire development of Quick OptMAVEn. Ratul helped me address questions large (e.g. How should we benchmark Quick OptMAVEn?) and small (e.g. What color should the antibody be in this figure?). I do not expect that I would have been able to complete this project without Ratul.

I would like to thank Jordan Paulus, a fellow undergraduate who collaborated with me on Quick OptMAVEn. Among other things, Jordan contributed ten gigabytes of antigen PDB files—which enabled me to run them through OptMAVEn—and helped me to understand the MILP at the core of OptMAVEn.

I would also like to acknowledge Drs. Tong Li and Robert Pantazes, both former members of the Maranas lab who contributed extensively to the development of OptMAVEn.

I would like to thank my principal investigator, Dr. Costas Maranas, very much for supporting my research since the summer after my first year of college. It was in the Maranas lab that I first united my interests in biochemistry and computer science. This experience changed my research and academic trajectories; motivated me to enroll in more courses on mathematics, statistics, computer science, and bioinformatics; and convinced me to pursue a research career in computational biology.

Finally, I would like to dedicate this thesis to the memory of our friend and collaborator, Klaus Schulten. Dr. Schulten was a professor at the University of Illinois at Urbana-Champaign until he sadly passed away in 2016. He contributed extensively to the field of structural biochemistry, most notably through the development of VMD and NAMD software, without which Quick OptMAVEn would not exist.

Chapter 1

Antibodies as therapeutics

Background on therapeutic monoclonal antibodies

Antibodies are important components of the adaptive immune system. Monoclonal antibodies (mAbs, distinguished from polyclonal antibodies) are the active ingredients in an increasing number of therapeutics. Currently, an average of four mAbs are approved by the FDA each year, with over 300 in development, and it is expected that approximately 70 will be on the market by 2020 [1]. Such medicines have been approved to treat infectious diseases, autoimmune disorders, metabolic disorders, and various types of cancers [2].

Drugs based on mAbs and other proteins (collectively termed “biologics”) are particularly attractive for multiple reasons. They are capable of binding their intended targets with high selectivity, minimizing off-target binding and hence side-effects [3]. There exist robust platforms for producing mAbs on industrial scales, most commonly in mammalian cells and occasionally in *E. coli* [1]. New mAbs can be designed using multiple well-established technologies, including hybridomas, mice with humanized immune systems, and phage display [3]. However, designing a mAb typically takes three to six months [4], a significant amount of time in the case of an epidemic. Thus, rapid methods for designing mAbs (such as Quick OptMAVEn, described in Chapter 3) could lead to significant improvements in the attenuation of epidemics or other time-critical situations.

Antibody structure and function

Before describing the Quick OptMAVEn framework for rapid design of mAbs, it is necessary to understand the structure of antibodies and how they are generated naturally and artificially. Antibodies are proteins. Like all proteins, they consist of peptide chains, which are polymers of amino acids. Antibodies have four peptide chains: two identical "heavy" chains and two identical "light" chains. In natural antibodies, the heavy chains are identical, as are the light chains. Each heavy chain has four domains—collections of amino acids that perform a common function—arranged in tandem. Three of the domains (C_H1, C_H2, and C_H3) are termed "constant" domains because they are identical across all antibodies within a given individual. The fourth domain (V_H) lies on one end of the heavy chain and is termed the "variable" domain because it varies among antibodies within one individual; its shape and amino acid sequence determine whether it will bind to a given molecule. Each light chain has two domains: one constant (C_L) and one variable (V_L) [3].

The overall shape of an antibody resembles the letter Y. Each arm of the Y comprises one light chain (C_L and V_L) and two of the domains of a heavy chain (C_H1 and V_H) arranged as a two-by-two array, with the heavy domains on the inside of the Y and both variable domains at the tip. The stem of the Y comprises the two remaining constant domains of each heavy chain (C_H2 and C_H3), also arranged as a two-by-two array. The four chains are connected by disulfide bonds that form between residues of the amino acid cysteine located on separate chains [3].

Antibodies function by physically binding to target molecules, known as antigens [5]. An antibody that neutralizes (abrogates the infectivity of) a virus, for example, binds to the virus particles (the antigens), which physically blocks the virus from binding to and thereby infecting host cells. The region of the antigen to which the antibody binds is known as the epitope, and the

region of the antibody that binds to the epitope is known as the paratope. The ability of an antibody to bind a given antigen is determined by the shape of the paratope, which is located at the tip of variable domains [5] and composed of six co-called complementarity-determining regions (CDRs): three from the V_H domain (CDR-H1, CDR-H2, and CDR-H3) and three from the V_L domain (CDR-L1, CDR-L2, and CDR-L3) [6]. The remaining non-CDR portions of the variable domains are known as the framework regions, which maintain the structure of the CDRs but are not directly involved in binding the antigen [6]. Although the CDRs are the most variable portions of antibodies, the framework regions may also vary among different antibodies in the same individual [7].

Generation of antibodies by the immune system

V-(D)-J recombination

For decades, the mechanism by which antibodies are generated presented a paradox: generating enormous variability in the variable domains ($\sim 3 \cdot 10^8$ potential germline sequences [8]) while using a finite genome. Clearly, there could not exist one gene for each of the sundry possible antibodies. Instead, germline genomes contain a relatively small library of genes encoding separate parts of antibodies, which are selected at random and fused into a complete antibody [9]. In mice [9] and humans [7], there are three families of genes encoding the variable domains: heavy-chain genes (for segments of V_H), κ genes (for segments of V_L), and λ genes (also for segments of V_L). The light chains of a natural antibody are encoded either by κ or λ genes, but never by both; hence, V_L domains assembled from κ or λ genes are known, respectively, as V_L-κ and V_L-λ

domains [7]. Heavy, κ, and λ gene families differ in their composition and in how they are arranged and assembled into complete V domains: a process known as V-(D)-J recombination [9]. V-(D)-J recombination occurs in B cells—white blood cells that manufacture antibodies [10]. New B cells bear all heavy, κ, and λ genes; during a process called maturation, which entails V-(D)-J recombination, they assemble a subset of these genes into a gene for a complete antibody [11].

The λ gene family includes 86 so-called variable λ (V_λ) genes and 10 so-called joining λ (J_λ) genes [8]. In the germline genome, the V_λ genes are clustered together, as are the J_λ genes, and the clusters are separated by a DNA spacer. During V-(D)-J recombination, one V_λ gene and one J_λ gene are selected and the intervening DNA spacer is removed, such that the selected V_λ and J_λ genes are fused tail-to-head, and the other V_λ and J_λ genes are excised from the genome [9]. This step alone would be capable of generating up to $86 \times 10 = 860$ unique V_L-λ domains [8] (though this is an over-estimate: not all combinations are observed [9]). However, several nucleotides may be added and deleted randomly at the junction of the two genes; the resulting amino acid insertions and deletions and frameshift mutations increase the number of V_L domains that can be produced from the λ gene family [9].

The κ gene family includes 97 V_κ genes and 9 J_κ genes [8], which undergo V-(D)-J recombination in a manner similar to that of the λ gene family. However, there is a substantial difference, albeit one that does not affect the V_L domain. In the λ gene family, a gene C_λ encoding the constant λ domain (C_L-λ) is located after each J_λ gene, and the pair of J_λ and C_λ genes are chosen together. Thus, C_L-λ does vary among antibodies that use λ genes (but it is not subject to nucleotide insertions and deletions), and the choice of C_L-λ depends on which J_λ gene was selected. However, for antibodies that use κ genes, there is only one gene C_κ encoding the constant κ domain (C_L-κ), so all κ antibodies have the same C_L-κ domain [9].

The heavy-chain gene family includes 324 V_H genes, 13 J_H genes, and 44 so-called diversity (D) genes [8], which lie between the V_H and J_H genes and have no counterparts in the λ or κ gene families. V-(D)-J recombination joins one V_H, one D, and one J_H gene tail-to-head-to-tail-to-head into a complete gene encoding one V_H domain; the remaining V_H, D, and J_H genes are excised from the genome. Similar to the formation of the V_L domain, nucleotides are inserted and deleted at the junctions of the V_H, D, and J_H genes. However, because there are a larger number of genes in the heavy-chain family, each is the product of three (not two) genes, and there are two junctions, the potential diversity among V_H domains is much greater than that among V_L domains [9]. In particular, the CDR-H3 loop is extremely variable [8], as it is encoded by the D gene and often by the end of the V_H gene, the beginning of the J_H gene, and the junctions between these genes [9].

The genes encoding antibodies vary in length, and nucleotides are added and subtracted randomly at their junctions; hence, the resulting V domains may contain different numbers of amino acids. Variation in length is problematic when determining which amino acids from different antibodies correspond to each other. Thus, the international ImMunoGeneTics information system (IMGT) has developed a standard numbering system for the amino acids in V domains (Figure 1) [12]. In this system, the amino acids in CDR-H3, for example, are always numbered from 105 to 117, regardless of the actual number of amino acids. If there are fewer than 13 amino acids in CDR-H3, then gaps are left in the numbers: if there were 5 amino acids, they would be numbered $105, 106, 107, 116, 117$. In the case that a CDR or framework region is longer than the allotted number of residues, letters are appended to the numbers, e.g. $\ldots, 110, 111, 111A, 111B, 111C, 112B, 112A, 112, 113, \ldots$ [12].

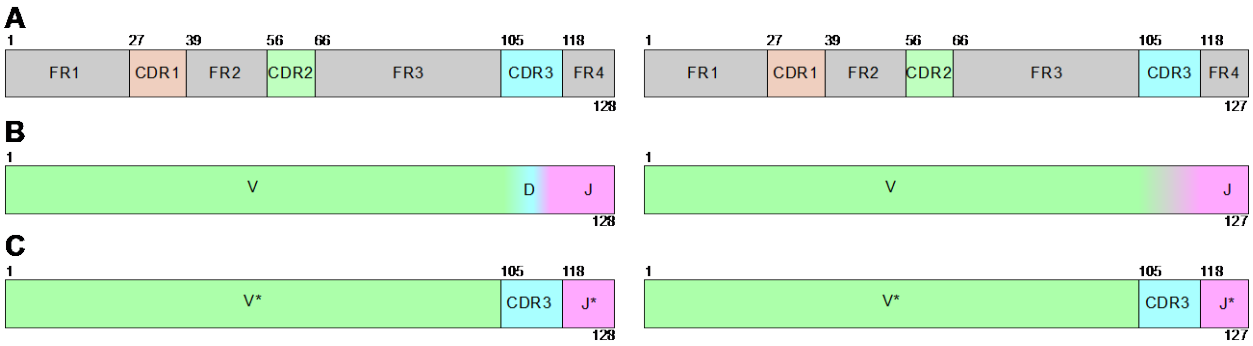

Figure 1: The ImMunoGeneTics (IMGT) numbering system for the heavy (left) and light (right) chains. All lengths are to scale. A) The IMGT numbers are given for the start and end of each framework region (FR) and complementarity-determining region (CDR). The number at the top left corner of each region labels its first residue. The number in its lower right corner (if present) labels its last residue. B) Correspondence of V, (D), and J germline genes to the FRs and CDRs. Because indels occur during V(D)J-recombination, the ranges of residues encoded by the genes vary among antibodies (indicated by the color gradients). C) Correspondence of the V*, CDR3, and J* regions of the MAPs database to the FRs, CDRs, and germline genes.

Affinity maturation

The process of V(D)J-recombination *in vivo* yields B cells that produce so-called "germline" antibodies with modest antigen affinities, e.g. $10^5 - 10^6\ \mathrm{M}^{-1}$ [13]. During the following several months, these B cells develop antibodies with higher affinities (e.g. $10^7 - 10^8\ \mathrm{M}^{-1}$) through a phenomenon known as affinity maturation. Affinity maturation entails two steps: 1) B cells deliberately make mutations to the genes encoding their antibodies—a process known as somatic hypermutation—and 2) B cells whose mutated antibodies have higher affinities proliferate, while those with lower affinities undergo apoptosis [14]. Iterating these two steps eventually yields B cells that produce high-affinity antibodies. Subsequent exposure to the antigen elicits the production of these affinity-matured antibodies, rather than the germline antibodies [13]. Somatic hypermutation preferentially targets amino acids closer to the antibody gene promoter, which lies just before the V gene [15]; and mutations also occur more frequently in the CDRs than in the framework regions [7].

Engineering therapeutic antibodies

Laboratory-based methods of engineering mAbs

Since the production of the first mAbs in 1984 [3], there have been extensive efforts to engineer therapeutic mAbs [3] [4] [6] [7]. Today, hybridoma technology is the most common method of developing a new mAb for a given antigen [10]. This process begins by immunizing a test animal by injecting the antigen of interest [10] [4]. The first mAb was generated using mice, which remain the most common test animals today [3], though rats, rabbits, hamsters, guinea pigs, goats, sheep, and chickens have also been used [4]. Additionally, in the case of an epidemic of a human disease, infected humans can be sources of antibodies, as happened during the recent epidemics of Zika [16] [17] and Ebola [18]. Immunizing the chosen animal elicits an immune response [10]. Prior to immunization, the animal harbors a large number of mature B cells. Through the randomized process of V-(D)-J recombination, a small number of them will have assembled antibodies capable of binding the antigen [11]. When these B cells—each with a unique antibody—recognize the antigen, they proliferate and begin to secrete antibodies. The resulting ensembles of multiple types of antibodies from multiple B cells are known as polyclonal antibodies (pAbs). The individual antibodies in a collection of pAbs recognize the same antigen but differ in terms of the epitopes that they bind and the affinities with which they bind the epitopes [10]. Recall that the initial pAbs (available in 1 – 2 weeks) are relatively low-affinity; creating high-affinity antibodies through natural affinity maturation requires over one month [13].

Relative to mAbs, pAbs are inexpensive to produce, demand less skill, and require less time (4 – 8 weeks versus 3 – 6 months for mAbs) [4]. However, mAbs are more consistent in terms of their antigen-binding properties, which is desirable for pharmaceuticals. To produce a mAb, the

spleen is removed from the immunized animal; the B cells within the spleen are fused with myeloma cells, yielding a chimeric type of cell known as a hybridoma. The hybridoma cells are cultured, and the antibodies they produce are screened using an enzyme-linked immunosorbent assay (ELISA), indicating which hybridomas produce antibodies capable of binding the antigen of interest [10]. Each such hybridoma cell, which produces a single type of mAb, serves as a stable and consistent source of the antibody [11].

A major challenge in antibody development is that the human body can mount an immune response against the mAbs themselves; mAbs that elicit this response are said to be immunogenic [3] [7]. Immunogenicity arises because the mAbs are developed in non-human animals, causing the human immune system to treat the mAbs as foreign pathogens [3] and produce so-called anti-drug antibodies (ADAs) [19]. ADAs have the potential to not only reduce the efficacy of therapeutic mAbs but also to cross-react with a patient's endogenous proteins, which can have severe consequences [19]. One approach to reduce immunogenicity is to create a chimeric antibody by fusing the V_H, V_L, C_H1, and C_L domains of a murine mAb to the C_H2 and C_H3 domains of a human mAb [3]. The risk for immunogenicity is lowered further through the process of humanizing the V_H, V_L, C_H1, and C_L domains—that is, making mutations that increase their similarities to human antibody sequences [3]. Recently, humanization has been taken a step further through genetically engineering mice with fully human immune systems, e.g. VelocImmune [20].

An alternative to hybridoma technology exists: phage display technology [10]. Phage display is a method of engineering mAbs *in vitro*, without the need to immunize mice or generate hybridomas. First, a library of 10^{12} *Inovirus* phage bearing $10^6 - 10^{11}$ unique antibodies (or other binder proteins) is created. The phage are incubated with the antigen, which is immobilized to a surface; a subsequent wash step removes unbound phage, leaving only those that bound with

sufficient affinity to the antigen. The phage that bound are then amplified in bacterial hosts; during replication, they undergo extensive genetic recombination, which generates further diversity in the antibodies or proteins they display. The amplified phage are again incubated with the antigen to identify the highest-affinity binders, and this process is repeated up to five times [10]. Note how this *in vitro* process recapitulates *in vivo* affinity maturation, in which B cells with high-affinity antibodies proliferate, while those with low-affinity antibodies die out [14]. Following the final round of phage display, the genomes of the surviving phage are sequenced to determine the sequences of the highest-affinity antibodies or proteins. Phage display is not yet as well-established as hybridoma technologies, is more difficult and expensive to perform, and yields a smaller number of unique mAbs. However, it offers the potential to generate antibodies free of immunogenic murine sequences in a relatively short amount of time. Furthermore, phage display can be extended from antibodies to proteins in general, while hybridoma technologies cannot [10].

Computational methods of engineering mAbs

Despite the large number of mAb-based pharmaceuticals currently on the market or in development [1], there exist many shortcomings of laboratory-based methods for engineering mAbs. A typical mAb requires 3 – 6 months to produce [4]. Furthermore, hybridoma and phage display technologies cannot deliberately target a particular epitope, ensure stability, or minimize immunogenicity [7]. Stability of mAbs is essential because unstable mAbs are prone to degrade in multiple ways. They may degrade chemically via deamidation of asparagine or glutamine residues, oxidation, hydrolysis of peptide bonds, or fragmentation of the chains [19]. Like many proteins, mAbs are also prone to aggregation. Degradation not only reduces the ability of mAbs to bind their

target antigens but may also create novel epitopes on the mAbs that can elicit an immune response [19] [21]. Even fully human mAbs may become immunogenic upon degradation [19]. Thus, methods of engineering antibodies should account for stability in addition to immunogenicity and affinity for the antigen.

There exist multiple computational methods for engineering mAbs with high antigen affinities (e.g. OptCDR [6], OptMAVEn [7], *AbDesign* [22], and Rosetta Antibody Design [23]), improving stability (e.g. Spatial Aggregation Propensity [21] and Rosetta Supercharge [24] [25]), and decreasing immunogenicity (e.g. molecular modeling [26] and human string content [27] [7]). Although computational methods are used widely to design small molecule drugs, virtually all therapeutic antibodies are still developed primarily via laboratory-based methods; there currently exists no widely adopted software for antibody design [28].

No existing software seems capable of performing *de novo* design and optimizing affinity, stability, and immunogenicity simultaneously; OptMAVEn, *AbDesign*, and Rosetta Antibody Design come closest. OptMAVEn [7] designs entire V_H and V_L domains *de novo*, i.e. without a starting structure of the antigen in complex with an antibody. OptMAVEn (discussed in detail in Chapter 2) first optimizes affinity using a fully deterministic algorithm, then iteratively optimizes affinity and immunogenicity but does not optimize stability of the mAbs. Poosarla et al. [29] used OptMAVEn to design five mAbs for a dodecapeptide, three of which were high affinity (though not as high as a natural antibody). Arginine hydrochloride prevented aggregation of the mAbs (which were produced in *E. coli*), and the mAbs refolded after exposure to urea; however, neither long-term stability nor immunogenicity was assessed.

In contrast to OptMAVEn, which designs mAbs *de novo* using a fully deterministic optimization procedure, Rosetta Antibody Design (RAbD) [23] requires the structure of an existing

antigen-antibody complex and uses an iterative Monte-Carlo-based method to optimize the antibody. In each iteration, RAbD randomly chooses a CDR and replaces it with a random CDR from a database of structures; the backbone, amino acid identities, and side chains of the new CDR are iteratively optimized using the Rosetta force field; and the new design is accepted or rejected based on a Metropolis criterion. Unlike in OptMAVEn, there is no humanization protocol to decrease immunogenicity. RAbD was verified experimentally for the design of mAbs targeting hyaluronidase. The 30 top-scoring designs were tested; none seemed to aggregate, and three bound with greater affinity than did the initial (wild-type) mAb. These three designs also had similar thermostabilities relative to the wild-type mAb. These data suggest that RAbD can improve the affinity of an existing antibody while maintaining similar stability, but there is no data on immunogenicity. However, OptMAVEn [7] is also capable of performing computational affinity maturation on its *de novo* designs while ensuring that their immunogenicities do not increase. Thus, OptMAVEn is capable of *de novo* design and humanization (while RAbD is not), and there seem to be no substantial capabilities of RAbD that are not implemented in OptMAVEn.

In addition to OptMAVEn and RAbD, *AbDesign* [22] aims to design high-affinity mAbs for a given antigen. Like RAbD, *AbDesign* requires an initial structure of the antigen in complex with an antibody. Like OptMAVEn, it assembles variable domains from a library of backbone canonical structures of the CDRs and framework regions. However, it uses two structures to assemble both the V_H domain (the structures are named VH and H3) and the V_L domain (VL and L3), while OptMAVEn uses three structures for each domain. The strategy behind *AbDesign* uses statistics of backbone conformations, rotamer conformations, and amino acid preferences at each sequence position in natural antibodies. From these statistics, it calculates a knowledge-based score indicating the similarity between the designed antibody and natural antibodies (assuming

that designs most similar to natural antibodies are most likely to be stable and have high affinities). The initial phase of *AbDesign* aligns a representative set of backbone canonical structures to the wild-type antibody (in essence, this step docks the antigen into the nascent antibody). This set comprises 5 VL, 2 L3, 9 VH, and 50 H3 structures, for a total of $5 \times 2 \times 9 \times 50 = 4500$ unique initial designs. Each design is mutated using natural amino acid preferences and then optimized with respect to backbone conformation. Subsequently, the affinity and stability are simultaneously optimized using a fuzzy logic energy function in which only the designs that are both stable and high-affinity score highly. The designs are finally filtered based on predicted binding energy, buried surface area, packing quality, and shape complementarity, yielding a set of designs for experimental validation. To date, there is no experimental data on the performance of *AbDesign*.

There are key similarities and differences among OptMAVEn [7], RAbD [23], and *AbDesign* [22]. All three methods rely on databases of canonical structures for regions of antibody variable domains. To the best of my knowledge and according to many sources [7] [6] [22] [23] [30], OptCDR and OptMAVEn are currently distinguished as the only computational methods capable of designing mAbs *de novo*. OptMAVEn explicitly optimizes affinity while preventing designs from becoming increasingly immunogenic, while RAbD and *AbDesign* explicitly optimize affinity and stability (and do not consider immunogenicity). Both RAbD and *AbDesign* use knowledge-based energy functions (i.e. structures are scored based on similarity to previously observed structures); OptMAVEn uses physics-based energy functions instead. Experimental evidence shows that both OptMAVEn and RAbD are capable of designing stable, high-affinity mAbs; there is yet no such evidence for *AbDesign*. OptMAVEn seems particularly well suited to design antibodies when the antigen structure is known but there is little to no structural information about how antibodies bind to the antigen. This was the case for over six months during the recent

Zika epidemic: the first structure of Zika envelope (E) protein was published on 31 March 2016 [31], while the first Zika-specific human antibodies were reported on 7 November 2016 [17]. Thus, OptMAVEn or similar software for *de novo* mAb design could potentially outpace other methods of developing therapeutic antibodies during epidemics.

It must be emphasized that computational methods are meant to suggest a small number of highly promising designs for experimental validation and/or refinement. They may expedite experimental testing but cannot fully replace it [28]. Many challenges in computational mAb design remain, such as simultaneously optimizing affinity, stability, and immunogenicity and predicting the structures of the highly variable CDR-H3 region [23] [22]. These challenges must be overcome before antibodies can be designed routinely via computational methods. However, given the extensive volume of research on computational mAb design and four significant software developments within the past five years [6] [7] [22] [23], the future of computational mAb design seems promising.

Thesis objectives

In this thesis, we introduce Quick OptMAVEn, software for *de novo* mAb design that improves upon shortcomings of OptMAVEn and includes novel features. First, we discuss how OptMAVEn works and identify six areas for improvement. Then, we describe how Quick OptMAVEn 1) features an intuitive and robust user interface, 2) positions the antigen with greater accuracy and speed, 3) eliminates sub-optimal designs earlier, 4) represents antigen positions using less disk storage, 5) reduces the expense of the design step without compromising quality, and 6) implements a novel clustering algorithm to retain promising designs that would otherwise be

discarded. We show that, on a set of ten antigens, Quick OptMAVEn uses 74% less CPU time and 84% less disk storage while designing antibodies of equivalent affinities. Finally, we use Quick OptMAVEn to design 77 *de novo* antibodies targeting Zika and identify 9 designs predicted to bind with greater affinity than a natural human antibody.

Chapter 2

OptMAVEn

Background of OptMAVEn

The need for a computational method to design fully human antibodies *de novo* motivated the development of OptMAVEn [7]. OptMAVEn is able to sample efficiently from a very diverse set of antibody structures by using a library of CDR structures. The structures of all CDRs except for CDR-H3 are usually limited to a set of so-called canonical structures [28]. The MAPs database [8] comprises 929 canonical structures and was developed to alleviate a major challenge in the computational design of antibodies and, more generally, of proteins: the enormous number of potential designs, which is illustrated as follows.

In the IMGT numbering system, CDRs 1, 2, and 3, respectively, comprise residues 27 – 38, 56 – 66, and 105 – 118 of the heavy and light chains [12]. Although the lengths of the CDRs vary among natural antibodies, such variation may be ignored for the purpose of this illustration: the six CDRs of an antibody comprise up to a total of $2 \times [(38 + 1 - 27) + (66 + 1 - 56) + (118 + 1 - 105)] = 74$ residues. Each residue can be one of twenty amino acids. If the amino acid identity of each residue is independent, then there are a total of $20^{74} \approx 2 \times 10^{96}$ possible antibodies with 74 CDR residues. A computer capable of predicting the binding affinities of one billion (10^9) antibodies per second would take approximately $2 \times 10^{96} \div 10^9 = 2 \times 10^{87}$ seconds (6×10^{79} years) to identify the antibody with the greatest binding affinity. This time is orders of magnitude longer than the lifetime of the universe.

To expedite computational antibody design, OptMAVEn mimics the natural process of V(D)J-recombination using the MAPs database. Rather than selecting amino acids individually, OptMAVEn selects one MAPs part for each CDR. OptMAVEn uses exclusively κ or λ domains when designing the light chain. Thus, the MAPs database can generate $[141 \times 428 \times 5] \times [(67 \times 199 \times 5) + (38 \times 39 \times 7)] \approx 2.3 \times 10^{10}$ antibodies (Table 1), a number more reasonable than 2×10^{96}. Moreover, some pairs of MAPs parts clash sterically, that is, at least one atom in one part is within 1 Å of at least one atom in the other part. Because these clashes lead to unstable structures, OptMAVEn does not consider any designs that contain clashes, further reducing the total number of potential antibodies. Among these potential designs, OptMAVEn identifies the optimal design efficiently using a mixed-integer linear program (MILP), which is discussed in more detail below.

Table 1: The number of parts in the MAPs database for each of the categories of parts. The number of parts in the original MAPs database is given in the OptMAVEn column. During the development of Quick OptMAVEn, five redundant parts were identified and removed from the MAPs database: the numbers in the Quick OptMAVEn column reflect these edits, and the categories in which edits occurred are indicated in the Edited column.

CDR	OptMAVEn	Quick OptMAVEn	Edited
HV	141	140	Yes
HCDR3	428	428	No
HJ	5	5	No
KV	67	64	Yes
KCDR3	199	199	No
KJ	5	5	No
LV	38	37	Yes
LCDR3	39	39	No
LJ	7	7	No
Total	**929**	**924**	

Workflow of OptMAVEn

OptMAVEn designs antibodies in three steps: preparation of input files, initial antigen positioning, grid search, energy calculations, germline design, and computational affinity maturation. These steps are detailed below. OptMAVEn is a command-line-based tool; commands must be entered on a Bash shell (e.g. Terminal on Linux or MacOS X) or an equivalent.

Preparation of input files

The user must first create a new directory (using **`mkdir`**), which will contain all files associated with the experiment; let this directory be called **`exp`**. The user must obtain a file of the antigen structure in Protein Data Bank (PDB) format. In many cases, this file can be downloaded from the PDB [32]. The user must then remove all atoms that are not part of the antigen, as well as all residues that are not supported by the input topology and parameter files (e.g. water, ions, modified amino acids, and small molecules). If such residues are essential for the design, the user must provide topology and parameter files that define these residues. Let the file of the antigen after removing these atoms be called **`ag.pdb`**, which must be located in **`exp`**. The user must specify the epitope of the antigen by listing the residue numbers of the epitope residues in a text file named **`Epitopes.txt`**, which is also located in **`exp`**. Each line of **`Epitopes.txt`** must begin with the name of the epitope, followed by a colon, followed by a space-separated list of the residues in the epitope. For example:

```
foo: 3 4 5 6 78 79 80 103 104 105 106
bar: 54 55 56 57 58 211 212 213 214 215
```

Figure 2: An example of the contents of an **`Epitopes.txt`** file. This file defines two epitopes named “foo” and “bar.”

The epitope may be identified using the structure of an existing antigen-antibody complex (see “Design of 77 antibodies for Zika E protein, including 9 predicted to be superior to native”). However, if such a structure is unavailable, there exist many tools for predicting epitopes [33], including Bcepred, Bepipred, ABCPred, and BEST.

Initial antigen positioning

OptMAVEn adds any atoms that may be missing from the antigen structure (e.g. hydrogen atoms) and then relaxes the structure to alleviate any steric clashes that may be present in the original PDB file or due to poorly guessed coordinates for atoms added in the previous step. The user must edit the script called **`minimizer.py`** and type the name of the file as an argument to the function **`MoleculeFile`** on line 19. Then, the user must run **`minimizer.py`** using the following command:

```
python
/gpfs/group/cdm8/legacy/group/tong/jordan/IPRO_Suite_all_probability/scripts/minimize
r.py
```

This script performs an energy minimization routine implemented in the CHARMM molecular dynamics program [34]. After the antigen structure has been relaxed, it must be rotated such that its epitope points towards the MAPs parts. This rotation ensures that the antibodies, once assembled from the MAPs parts, will contact the antigen at the intended epitope, rather than at an unspecified location on the surface of the antigen (Figure 3).

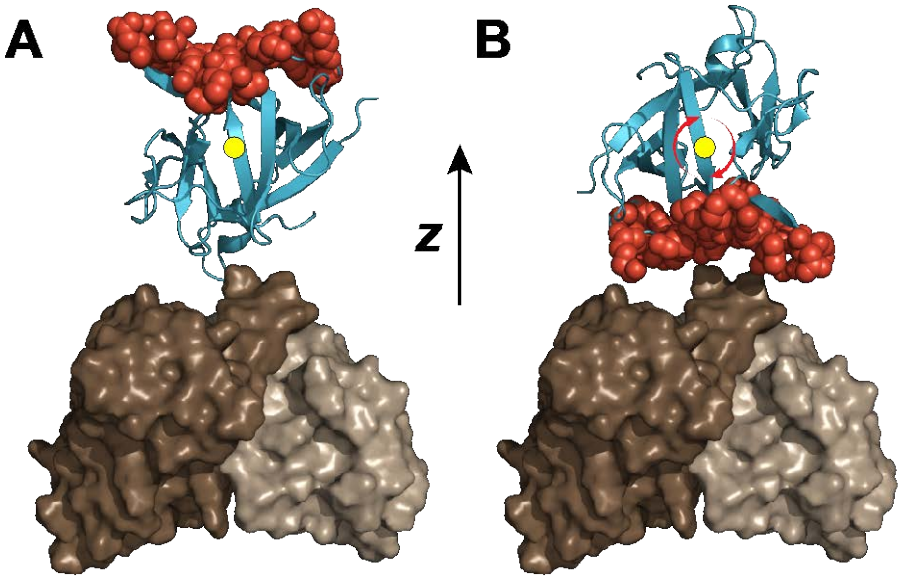

Figure 3: Illustration of the rotation of an antigen. The antigen (blue) is rotated around its geometric center (yellow) from its arbitrary initial position (A) to its final position (B) so as to minimize the sum of the z coordinates of its epitope (red); the z axis points upward, as shown. This rotation ensures that the epitope points towards the MAPs parts. One MAPs part for each CDR, forming a complete antibody variable domain, is shown; the heavy chain is brown and the light chain is tan. All other MAPs parts occupy similar positions, so the epitope in panel B will point toward an antibody assembled from any set of MAPs parts. The antigen shown is human interleukin 1-beta (PDB ID 46GM) [35]. Images were generated with PyMOL [36].

Mathematically, this rotation is achieved by minimizing the mean of the z coordinates of the atoms of the epitope while keeping the geometric center of the antigen fixed:

1)

$$\text{minimize } z = \sum_{i \in I^{\text{epitope}}} z_i$$

subject to

2)

$$Ac_0 = c$$

where I is the set of all atoms in the antigen, $I^{\text{epitope}} \subset I$ is the subset of antigen atoms that are within the epitope, $n = \text{card}(I)$ is the cardinality of (number of elements in) I, z_i is the z coordinate of atom i, $c = \begin{pmatrix} x_1 & & x_n \\ y_1 & \dots & y_n \\ z_1 & & z_n \end{pmatrix}$ is the set of final (post-rotation) x, y, and z coordinates of all atoms, c_0 is the set of initial (pre-rotation) coordinates of all atoms, and $A \in M_{3,3}\{\mathbb{R}\}$ is a three-

dimensional rotation matrix. This minimization is implemented as an exhaustive search represented by the following pseudocode:

```
c_0 = get_coordinates(initial_antigen_structure)  # initial coordinates
z_min = 1000000  # a large number
for x_rotation in {0, 3, 6, ..., 357}:  # 3-degree x increments
    for y_rotation in {0, 3, 6, ..., 357}:  # 3-degree y increments
        A = make_rotation_matrix(x_rotation, y_rotation)
        c = A * c0  # matrix multiplication rotates coordinates
        c_epi = get_epitope_coordinates(c)  # extract the rotated epitope coordinates
        z_epi = get_z_coordinates(c_epi)  # extract the rotated epitope z coordinates
        if sum(z_epi) < z_min:  # if the z coordinate sum is less than the smallest
                # sum so far
            c_best = c  # save the structure with the smallest epitope z coordinates
            z_min = sum(z_epi)  # update the smallest sum so far
return c_best  # output the coordinates of the best structure
```

Figure 4: Pseudocode for the epitope rotation algorithm in OptMAVEn.

The user implements this step, along with the next (grid search), by typing

```
python
/gpfs/group/cdm8/legacy/group/tong/jordan/IPRO_Suite_all_probability/scripts/grid_sea
rch.py
```

Before running this script, the user must edit it as follows:

In line 43, replace the key to the dictionary **`inputFile`** with the one-letter name of the chain of the antigen (for example, **`P`**):

```
G = inputFile["P"]
```

In line 45, replace the second argument of the function **`parameter_output_antigen`** with the name of a new text file in which to store the antigen (for example, **`5gzn.txt`**):

```
parameter_output_antigen(G, "5gzn.txt")
```

In line 60, replace the second string following **`cmd`** with the name of the aforementioned file:

```
cmd =
"/gpfs/home/tul12/work/soft/IPRO_Suite/modules/CPP/initialization/initialantigen_clas
h.out " + "5gzn.txt " + "epitope.txt " + "MoleculeH.txt " + "MoleculeK.txt " +
str(angle)
```

In line 61, replace the name of the first string after **runScriptFile** with the name of the file (minus the extension):

```
runScriptFile = "5gzn_" + str(angle)
```

In line 74, replace the text **chmod +x** with **qsub**:

Before:

```
cmd = "chmod +x " + runScriptFile
```

After:

```
cmd = "qsub " + runScriptFile
```

Grid search

The purposes of the grid search step are to 1) generate a set of antigen positions on which the subsequent steps rely and 2) quickly filter out positions that will inevitably cause steric clashes between the antibody and antigen. The developers of OptMAVEn defined a set of grid points to support the grid search. By examining the structures of 750 antibody-bound antigens, they found that the center of geometry of the epitope, (x, y, z), was always located within a parallelepiped defined by $-10 \leq x \leq 5$, $-5 \leq y \leq 10$, $3.75 \leq z \leq 16.25$, where all dimensions are in Å. The x , y, and z dimensions are partitioned into increments of 2.5 Å, 2.5 Å, and 1.25 Å, respectively. To increase the diversity of antigen conformations, the antigen is additionally rotated around the z axis in increments of 60°. Thus, OptMAVEn samples a total of $\left(\frac{5-(-10)}{2.5}+1\right) \times \left(\frac{10-(-5)}{2.5}+1\right) \times \left(\frac{16.25-3.75}{1.25}+1\right) \times \left(\frac{360-0}{60}\right) = 7 \times 7 \times 11 \times 6 = 3234$ antigen positions. Each position can thus be defined by a point (x, y, z) and a z rotation angle θ_z.

For each antigen position (x, y, z, θ_z), the antigen is translated such that the center of geometry of its epitope lies at the (x, y, z) point of the position; and the antigen is rotated around

the z axis such that it faces in the direction given by θ_z, where $\theta_z \equiv 0°$ for the structure produced by minimizing the epitope z coordinates. At each position, OptMAVEn counts the number of steric clashes between the antigen and a framework antibody. The framework antibody is an antibody for which all six CDRs are composed entirely of glycine, the smallest amino acid, whose side chain is a hydrogen atom. Thus, if the antigen in the given position clashes with the framework antibody, it will clash with any other antibody. OptMAVEn saves a PDB file of the antigen in each position for which there are two or fewer steric clashes; positions with more than two clashes are discarded.

Energy calculations

For each antigen position, OptMAVEn calculates the interaction energy between the antigen and every part in the MAPs database. The energy calculations are performed using a C++ program written and compiled in-house. This program produces the same results as an equivalent energy calculation implemented in CHARMM but is hard-coded to calculate interaction energies with every part in the MAPs database. Prior to running the program, the structure of the antigen must be converted from PDB format to an alternate format that specifies parameters for the energy calculation in addition to atom types and coordinates. This file is generated by running **`/gpfs/group/cdm8/legacy/group/tong/jordan/IPRO_Suite_all_probability/scripts/grid_search.py`**, as described in Initial antigen positioning. The MAPs database files are already provided in this format and need no pre-processing. The files may include parameters for electrostatic, van der Waals, and Lazaridus-Karplus solvation energies. Care must be taken to ensure that all files contain all desired parameters; OptMAVEn will, without informing the user,

disregard energy terms whose parameters are missing from any file. The user performs the energy calculations by typing

```
python
/gpfs/group/cdm8/legacy/group/tong/jordan/IPRO_Suite_all_probability/scripts/calculate_energy.py
```

Before running this script, the user must edit line 5 to replace the arguments to the **`range`** function so as to generate the correct range of angles (e.g. **`range(0, 360, 60)`** to generate angles from 0 to 300 in increments of 60).

Germline design

OptMAVEn generates antibody designs using a mixed-integer linear program (MILP) that selects, for each antigen position, six MAPs parts—one for each CDR—such that the sum of the interaction energies between the MAPs parts and the antigen is minimized. The MILP is formulated as follows:

3)

$$\text{minimize } E = \sum_{i \in I} \sum_{j \in J} \sum_{k \in K_{i,j}} y_{i,j,k} e_{i,j,k}$$

subject to

4)

$$d_H = 1$$

5)

$$d_K + d_L = 1$$

6)

$$\sum_{k \in K_{i,j}} y_{i,j,k} = d_i \; \forall \; (i,j) \in I \times J$$

7)

$$y_{i_1,j_1,k_1} + y_{i_2,j_2,k_2} \leq 1 \; \forall \; [(i_1,j_1,k_1),(i_2,j_2,k_2)] \in P^{\text{clash}}$$

where $I = \{H, K, L\}$ is the set of antibody chain loci, $J = \{V, CDR3, J\}$ is the set of gene segment types, $K_{i,j} = \{1, 2, \ldots, n_{i,j}\}$ is the set of MAPs parts of chain i and segment type j (where $n_{i,j}$ is the number of such parts), P^{clash} is the set of all pairs of MAPs parts between which steric clashes exist, $e_{i,j,k}$ is the interaction energy between the antigen and part (i,j,k) computed in the Energy Calculation step, $y_{i,j,k} = 1$ if part (i,j,k) is incorporated in the design and 0 otherwise, $d_i = 1$ if a part is to be selected for chain i and 0 otherwise, and $\times$ denotes the Cartesian product.

Equation 3, the objective function, causes the interaction energy E to be minimized (thus maximizing the affinity). Equation 4 tells OptMAVEn to use the heavy locus to design the heavy chain. Equation 5 tells OptMAVEn to use either (but not both) the κ or λ locus to design the light chain. Equation 6 ensures that for each locus i to be used, exactly one part is selected for each gene segment type; and for each locus to be avoided, no parts are selected. Equation 7 prevents steric clashes between MAPs parts.

For each antigen position, this MILP is solved five times in order to generate an ensemble of designs. The rationale is that there is a greater likelihood of obtaining a least one high-affinity antibody from a set of five designs than from a set of just one design. Each time the MILP is solved, an integer cut is added to the MILP formulation to prevent the same six parts from being chosen again. This process ensures that the first design has the most negative interaction energy, the second the second-most negative, and so on.

The user performs this step by typing

```
python
/gpfs/group/cdm8/legacy/group/tong/jordan/IPRO_Suite_all_probability/scripts/select_parts300.py
```

Computational affinity maturation

The germline designs with the most negative interaction energy are improved using a computational process that mimics *in vivo* affinity maturation. A modified version of the IPRO protocol [37] makes mutations to the germline antibodies so as to further decrease their interaction energies. This modified protocol is a cycle of five steps: selection of residues to mutate, backbone perturbation, humanized rotamer selection, antigen redocking, and energy calculation.

Selection of residues to mutate

One of the residues in either the light or heavy chain is selected for mutation. The probability of selecting a particular residue in a CDR is set to be 3 times the probability of selecting a particular residue in an FR. However, since the residues within the CDRs are at most $\frac{74}{128+127} = 0.29$ of the variable domains, the probability of selecting a residue within a CDR is at most $\frac{3\times0.29}{3\times0.29+(1-0.29)} = 0.55$. After selecting this residue, both, either, or neither of the 2 residues adjacent to the selected residue are also selected for mutation.

Backbone perturbation

OptMAVEn perturbs the ϕ and ψ dihedral angles of all residues within a 4.5 Å radius or a 5-residue window of the 1 – 3 residues selected for mutation. The perturbation θ for each angle is randomly selected from a Gaussian distribution of $\mu = 0°$ and $\sigma = 1.5°$ that is truncated to the domain of $-5° \leq \theta \leq +5°$. Subsequently, the perturbed structure is relaxed with CHARMM using strong restraints to enforce the values of the newly perturbed dihedral angles. The 5 residues on

either side of the perturbed region are temporarily mutated to glycine for the duration of the relaxation and are the only residues allowed to move.

Humanized rotamer selection

The user may specify a limited set of amino acids to which each site can be mutated. The perturbed residues, those mutated to glycine, and those in the vicinity are mutated to new amino acids using the IPRO rotamer library and MILP formulation [37]. Following these mutation, a humanization score is calculated for the new antibody sequence to prevent the human body from mounting an immune response against the antibody. To develop the humanization calculation, 69,032 human antibody sequences were analyzed to identify the set S_9 of all unique 9-mer sequences of amino acids, of which there were 1,309,657. For a query 9-mer sequence q, let $m_q^{\min}$ denote the minimum number of mutations that must be made to q to produce a mutated sequence $q' \in S_9$. Then, the humanness score h of a query sequence Q comprising $n \geq 9$ amino acids is given by

8)

$$h(Q) = \sum_{i=1}^{n-8} m_{Q_{i,i+8}}^{\min}$$

where $Q_{i,i+8}$ denotes the 9-mer sequence of positions $\{i, i+1, \ldots, i+8\}$ of Q. If h of the newly mutated antibody is less than h of the previous design, then the mutations are discarded.

Antigen redocking

Because redocking is time-intensive, it is performed once every third iteration of the cycle. Redocking is performed for 500 iterations. In each iteration, the antigen is translated along the x, y, and z axes by randomly selecting the translation along each axis from a Gaussian distribution with $\mu = 0$ Å and $\sigma = 0.2$ Å; and rotating the antigen around the x, y, and z axes by randomly selecting the rotation around each axis from a Gaussian distribution with $\mu = 0°$ and $\sigma = 2°$. The interaction energy between the antibody and the perturbed antigen is calculated; based on this energy and the pre-perturbation energy, the perturbation is accepted or rejected based on the Metropolis criterion [38].

Energy calculation

The antigen-antibody complex is relaxed using CHARMM with a full-atom energy function that incorporates electrostatic, Van der Waals, bond length, angle, dihedral, and Generalized Born solvation energy terms. The interaction energy between the antigen and antibody is calculated. A Metropolis criterion [38] based on the interaction energies before and after the mutations is used to determine whether or not to accept the mutations made to the antibody.

Output and validation of OptMAVEn

The output of OptMAVEn is a small number of antibody structures in PDB format. At the time of this writing, one study has experimentally validated antibodies generated using OptMAVEn [29]. The antigen was a dodecapeptide with the sequence DVFYPYPYASGS; a

crystal structure of this antigen bound to the scFv-2D10 was obtained from the PDB (ID 4H0H) [39]. OptMAVEn generated five antibodies, three of which bound the antigen with high affinities ($K_D = 8.9$, 14.4, and 23.8 nm). While the scFv-2D10 bound with greater affinity ($K_D = 3.8$ nm), these results demonstrated that OptMAVEn was capable of designing high-affinity antibodies without a need for an initial antibody structure.

Chapter 3

Quick OptMAVEn

Motivation for Quick OptMAVEn

Despite the success of OptMAVEn in one study [29], OptMAVEn provides several areas for changes that could potentially improve the performance of the software. These are:

Use of separate tools increases risk of user error

OptMAVEn integrates tools that must be run separately, often with manual manipulations (e.g. moving and editing files) between each step. The risk of the user making an error could be reduced via an integrated workflow that requires no manual intervention between the step in which the antigen and epitope are defined and the step in which the final antibody structures are generated.

Minimization of epitope *z* coordinates is inefficient and imprecise

The step in which the antigen is rotated so as to minimize the z coordinates of the epitope uses an inefficient exhaustive search that samples rotations along the x and y axes, each in increments of 3°. Because this search uses discrete steps, the optimal conformation sampled may be up to $\left(\frac{3\sqrt{2}}{2}\right)°$ off relative to the global optimum. This result is analogous to that of approximating

a point $p \in \mathbb{R}^2$ using the nearest point p' whose coordinates p'_x and p'_y are both divisible by 3. The maximum distance between p and p' is $\|p' - p\| = \frac{3\sqrt{2}}{2}$, as in the case of $p = \binom{1.5}{1.5}$.

A clash-permissive grid search increases cost of subsequent steps

The grid search permits up to two clashes between the antigen and the framework antibody. Thus, clashing antigen positions may be retained and passed to the energy calculation step. Because clashes lead to high energy penalties, clashing structures will ultimately fail to rank among the best (lowest-energy) designs. Thus, computing the energies and optimal sets of MAPs parts for clashing antigen positions yields no useful antibody designs and demands additional computational resources. Discarding antigen positions that clash would likely reduce computational demand without sacrificing results.

Representation of antigen positions as PDB files increases disk storage requirement

The grid search outputs a set of antigen positions, each represented as a separate PDB file. A PDB file can be large: for example, the first published Zika virus structure (ID 5IRE) [31] is approximately 1 MB. Thus, the total size of the files representing all antigen positions (of which there are potentially 3234) could be up to $3234 \times 1\ \text{MB} \approx 3\ \text{GB}$. While modern disk and solid-state drives are considerably larger, this storage requirement poses several problems: 1) writing, parsing, reformatting, copying, and deleting these files may take considerable time; 2) transferring these files between filesystems (e.g. over a wireless connection) may be excessively slow; and 3) there may be insufficient free storage space on the machine used to run OptMAVEn (e.g. a shared

supercomputing cluster). We encountered all of these problems when designing an antibody for Zika virus. However, because all antigen structures are identical up to a translation and rotation, it should be possible to represent them in a more compact format: namely, to store one reference antigen structure and a file listing the translations and rotations needed to generate the other structures from the reference.

Design of multiple antibodies for each antigen position does not increase design quality

During the MILP step, five designs are created for each antigen position. However, the MILP is formulated such that the estimated interaction energy of the designs becomes less negative with increasing design number. Thus, designs two through five are sub-optimal relative to design one and will thus be discarded during the subsequent ranking of designs.

Absence of a method to ensure diversity among designs

OptMAVEn seeks to maximize the diversity of the antibodies it designs for a given epitope, under the assumption that the likelihood of finding at least one high-affinity design is greater if the designs are diverse than if they are highly similar. However, the designs whose MILP interaction energies are most negative are not necessarily structurally diverse.

Development of Quick OptMAVEn

Several of the key steps in OptMAVEn were updated to improve their performance in accordance with the topics discussed in “Motivation for Quick OptMAVEn.”

New coherent directory structure

Quick OptMAVEn features a new organizational structure for its directories. All files are located within a main directory, **`Optmaven-2`**, which can be ported to and run in a different location without the need to modify any files or formally install the program (assuming all dependencies on third-party software are met). **`Optmaven-2`** contains subdirectories **`src`**, **`data`**, and **`experiments`** (**`experiments`** may be missing, in which case it is created upon running a new experiment). **`src`** contains all source code (Python and TCL modules); **`data`** contains all reference data files (antigen structures, the MAPs database, framework antibodies, and topology and parameter files); and **`experiments`** contains a subdirectory for each OptMAVEn experiment. **`Optmaven-2`** also contains five executable scripts: **`optmaven`** starts a new OptMAVEn experiment, **`interaction_energy`** calculates the interaction energy between a given antigen and antibody, **`find_contacts`** identifies the residues in the interface between an antigen and antibody (for example, to determine an epitope), **`check_status`** reports on the status of all experiments in the **`experiments`** directory, and **`remove_experiment`** permanently removes an experiment's directory and files. This structure organizes all necessary files in one directory tree, making it easier to locate files than it was in the original OptMAVEn, which did not feature a centralized location for all files.

Robust input-output methods

Quick OptMAVEn further reduces the risk of error by reading and writing structural files using software that is more robust than that of OptMAVEn. Quick OptMAVEn writes and parses PDB files using Biopython [40]: a robust, widely-used, and well-maintained set of modules for

processing biological data. OptMAVEn relies on in-house writers and parsers, which have not been validated as extensively as Biopython and are not safeguarded against future changes to the PDB format. Moreover, Quick OptMAVEn is capable of reading alternate structure formats (e.g. MMCIF) that are supported in Biopython.

Improved user interface

Original procedure: Preparation of input files

Problem: Use of separate tools increases risk of user error

While OptMAVEn required the user to create a directory for the experiment, edit the files of the antigen to remove all atoms not part of the antigen, and create a file specifying the epitope, Quick OptMAVEn automatically performs these steps, reducing the effort required of the user and the risk of error. Upon typing **`./optmaven`** at the command line, the user is first asked to name the experiment. After ensuring that no identically-named experiment already exists, Quick OptMAVEn creates a directory for that experiment. The user then types the name of the antigen file. Quick OptMAVEn identifies any heteroatoms and allows the user to choose which to exclude using an intuitive selection syntax new to Quick OptMAVEn: hyphens indicate ranges of residues to exclude, and commas separate residue numbers or ranges. For example, **`212-214, 216`** excludes residues 212, 213, 214, and 216 from the structure. To improve usability, **`all`** and **`none`** are also valid responses. Quick OptMAVEn ensures that all selected residues actually exist (OptMAVEn did not check to make sure the residues in **`Epitopes.txt`** actually existed) and that an appropriate number have been selected. After excluding heteroatoms, the user specifies the chain(s) that are part of the antigen and the epitope residues for each chain. Quick OptMAVEn

then automatically starts the experiment, needing no further action from the user to complete the designs.

Initial antigen positioning

Original procedure: Initial antigen positioning

Problem: Minimization of epitope z coordinates is inefficient and imprecise

Quick OptMAVEn uses an exact, robust, and efficient method to minimize the z coordinates of the epitope. The objective is to minimize the sum z of the epitope z coordinates (Equation 1) while holding fixed the center of geometry (defined in Equation 9) of the antigen (i.e. performing only rotations). For a set of atoms I, each with a coordinate $c_i = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \forall\, i \in I$, where $n = \mathrm{card}(I)$, the center of geometry $\overline{c_I}$ is defined herein as

9)

$$\overline{c_I} = \frac{1}{n}\sum_{i \in I} c_i$$

Without loss of generality, we can assume that, for the antigen A, $\overline{c_A} = \mathbf{0}$; if this is not the case, then we can first translate the antigen by $-\overline{c_A}$, which will ensure that $\overline{c_A} = \mathbf{0}$. Equation 1 is equivalent to minimizing the mean $\overline{z}$ of the z coordinates:

10)

$$\text{minimize } \overline{z} = \frac{1}{n^{\text{epitope}}} \sum_{i \in I^{\text{epitope}}} z_i$$

because n^{epitope}, the number of residues in the epitope, is a fixed positive integer. From the definition of center of geometry (Equation 9), it can be shown that $\overline{z}$ is the z coordinate of the geometric center of the epitope:

11)

$$\overline{c_{I^{\text{epitope}}}} = \frac{1}{n^{\text{epitope}}} \sum_{i \in I^{\text{epitope}}} \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \begin{pmatrix} \frac{1}{n^{\text{epitope}}} \sum_{i \in I^{\text{epitope}}} x_i \\ \frac{1}{n^{\text{epitope}}} \sum_{i \in I^{\text{epitope}}} y_i \\ \frac{1}{n^{\text{epitope}}} \sum_{i \in I^{\text{epitope}}} z_i \end{pmatrix} = \begin{pmatrix} \overline{x} \\ \overline{y} \\ \overline{z} \end{pmatrix}$$

where $\overline{x}$ and $\overline{y}$ are defined similarly to $\overline{z}$ in Equation 10. Because the structure of the antigen is held constant up to a rotation during the minimization of $\overline{z}$, all distances between coordinates within the antigen remain constant. The distance d between $\overline{c_{I^{\text{epitope}}}}$ and $\overline{c_A}$ is

12)

$$d = \|\overline{c_{I^{\text{epitope}}}} - \overline{c_A}\| = \|\overline{c_{I^{\text{epitope}}}} - \mathbf{0}\| = \|\overline{c_{I^{\text{epitope}}}}\| = \sqrt{\overline{x}^2 + \overline{y}^2 + \overline{z}^2}$$

Thus, the minimization problem becomes

13)

minimize $\overline{z}$

subject to

14)

$$\sqrt{\overline{x}^2 + \overline{y}^2 + \overline{z}^2} = d = \sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2 + {\overline{z}_0}^2}$$

where $\overline{x}_0$, $\overline{y}_0$, and $\overline{z}_0$ denote, respectively, the initial (pre-rotation) values for $\overline{x}$, $\overline{y}$, and $\overline{z}$. By rearranging Equation 14 to $\overline{z} = \pm\sqrt{d^2 - (\overline{x}^2 + \overline{y}^2)}$, differentiating, and equating to zero:

15)

$$0 = \frac{\partial \overline{z}}{\partial \overline{x}} = \frac{\overline{+}\overline{x}}{\sqrt{d^2 - (\overline{x}^2 + \overline{y}^2)}}$$

16)

$$0 = \frac{\partial \overline{z}}{\partial \overline{y}} = \frac{\overline{+}\overline{y}}{\sqrt{d^2 - (\overline{x}^2 + \overline{y}^2)}}$$

Assuming that $\sqrt{d^2 - (\overline{x}^2 + \overline{y}^2)} = \pm\overline{z} \neq 0$, the problem is solved when $\overline{x} = \overline{y} = 0$:

17)

$$\overline{z} = \pm\sqrt{d^2 - (\overline{x}^2 + \overline{y}^2)} = \pm\sqrt{d^2} = \pm d$$

Because the objective is to minimize $\overline{z}$, and because $d \geq 0$,

18)

$$\overline{z} = -d = -\sqrt{\overline{x}_0{}^2 + \overline{y}_0{}^2 + \overline{z}_0{}^2}$$

19)

$$\overline{c_{I^{\text{epitope}}}} = \begin{pmatrix} 0 \\ 0 \\ -d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ -\sqrt{\overline{x}_0{}^2 + \overline{y}_0{}^2 + \overline{z}_0{}^2} \end{pmatrix}$$

It is possible to construct a 3D rotation matrix A that rotates the initial epitope center of geometry $\overline{c_{I^{\text{epitope}}}}_0 = \begin{pmatrix} \overline{x}_0 \\ \overline{y}_0 \\ \overline{z}_0 \end{pmatrix}$ to the desired coordinate $\overline{c_{I^{\text{epitope}}}}$. While several formulations exist for A, the most convenient rotates by angle κ around a unit vector $u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix}$ (Figure 5) [41]. Note that because u is the axis of rotation, $\overline{c_{I^{\text{epitope}}}} \perp u \perp \overline{c_{I^{\text{epitope}}}}_0$. Thus, u can be calculated as $u =$

$(\overline{c_{I\text{epitope}}}_0 \times \overline{c_{I\text{epitope}}}) \div \left\|\overline{c_{I\text{epitope}}}_0 \times \overline{c_{I\text{epitope}}}\right\|$, because the cross product yields a vector perpendicular to the two given vectors; the division is necessary to normalize u. We obtain

$$\overline{c_{I\text{epitope}}}_0 \times \overline{c_{I\text{epitope}}} = \begin{pmatrix} \overline{x}_0 \\ \overline{y}_0 \\ \overline{z}_0 \end{pmatrix} \times \begin{pmatrix} 0 \\ 0 \\ -d \end{pmatrix} = \begin{pmatrix} -\overline{y}_0 d - \overline{z}_0 \cdot 0 \\ \overline{z}_0 \cdot 0 + \overline{x}_0 d \\ \overline{x}_0 \cdot 0 - \overline{y}_0 \cdot 0 \end{pmatrix} = \begin{pmatrix} -d\overline{y}_0 \\ d\overline{x}_0 \\ 0 \end{pmatrix}$$

20)

$$u = \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} -d\overline{y}_0 \\ d\overline{x}_0 \\ 0 \end{pmatrix} \div \left\| \begin{pmatrix} -d\overline{y}_0 \\ d\overline{x}_0 \\ 0 \end{pmatrix} \right\| = \begin{pmatrix} -\overline{y}_0 \div \sqrt{\overline{x}_0{}^2 + \overline{y}_0{}^2} \\ \overline{x}_0 \div \sqrt{\overline{x}_0{}^2 + \overline{y}_0{}^2} \\ 0 \end{pmatrix}$$

Furthermore, $\cos\kappa$ and $\sin\kappa$ can be obtained using the standard relations:

21)

$$\cos\kappa = \left(\overline{c_{I\text{epitope}}}_0 \cdot \overline{c_{I\text{epitope}}}\right) \div \left(\left\|\overline{c_{I\text{epitope}}}_0\right\| \left\|\overline{c_{I\text{epitope}}}\right\|\right) = \left(\begin{pmatrix} \overline{x}_0 \\ \overline{y}_0 \\ \overline{z}_0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 0 \\ -d \end{pmatrix} \right) \div (d \cdot d) = \frac{-\overline{z}_0}{d}$$

22)

$$\sin\kappa = \left\|\overline{c_{I\text{epitope}}}_0 \times \overline{c_{I\text{epitope}}}\right\| \div \left(\left\|\overline{c_{I\text{epitope}}}_0\right\| \left\|\overline{c_{I\text{epitope}}}\right\|\right) = \left\| \begin{pmatrix} -d\overline{y}_0 \\ d\overline{x}_0 \\ 0 \end{pmatrix} \right\| \div (d \cdot d)$$

$$= \frac{\sqrt{\overline{x}_0{}^2 + \overline{y}_0{}^2}}{d}$$

Finally, A can be found using this standard formula [41]:

23)

$$A = \begin{pmatrix} {u_x}^2 + ({u_y}^2 + {u_z}^2)\cos\kappa & u_x u_y(1-\cos\kappa) - u_z\sin\kappa & u_z u_x(1-\cos\kappa) + u_y\sin\kappa \\ u_x u_y(1-\cos\kappa) + u_z\sin\kappa & {u_y}^2 + ({u_z}^2 + {u_x}^2)\cos\kappa & u_y u_z(1-\cos\kappa) - u_x\sin\kappa \\ u_z u_x(1-\cos\kappa) - u_y\sin\kappa & u_y u_z(1-\cos\kappa) + u_x\sin\kappa & {u_z}^2 + ({u_x}^2 + {u_y}^2)\cos\kappa \end{pmatrix}$$

$$= \begin{pmatrix} \frac{{\overline{y}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2} + \left(\frac{{\overline{x}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\right)\frac{-\overline{z}_0}{d} & \frac{-\overline{x}_0\overline{y}_0}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\left(1 + \frac{\overline{z}_0}{d}\right) - 0 & 0 + \frac{\overline{x}_0}{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}\frac{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}{d} \\ \frac{-\overline{x}_0\overline{y}_0}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\left(1 + \frac{\overline{z}_0}{d}\right) + 0 & \frac{{\overline{x}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2} + \left(\frac{{\overline{y}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\right)\frac{-\overline{z}_0}{d} & 0 + \frac{\overline{y}_0}{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}\frac{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}{d} \\ 0 - \frac{\overline{x}_0}{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}\frac{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}{d} & 0 - \frac{\overline{y}_0}{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}\frac{\sqrt{{\overline{x}_0}^2 + {\overline{y}_0}^2}}{d} & 0 + \left(\frac{{\overline{y}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2} + \frac{{\overline{x}_0}^2}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\right)\frac{-\overline{z}_0}{d} \end{pmatrix}$$

$$= \begin{pmatrix} \frac{{\overline{y}_0}^2 - \frac{{\overline{x}_0}^2\overline{z}_0}{d}}{{\overline{x}_0}^2 + {\overline{y}_0}^2} & \frac{-\overline{x}_0\overline{y}_0}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\left(1 + \frac{\overline{z}_0}{d}\right) & \frac{\overline{x}_0}{d} \\ \frac{-\overline{x}_0\overline{y}_0}{{\overline{x}_0}^2 + {\overline{y}_0}^2}\left(1 + \frac{\overline{z}_0}{d}\right) & \frac{{\overline{x}_0}^2 - \frac{{\overline{y}_0}^2\overline{z}_0}{d}}{{\overline{x}_0}^2 + {\overline{y}_0}^2} & \frac{\overline{y}_0}{d} \\ \frac{-\overline{x}_0}{d} & \frac{-\overline{y}_0}{d} & \frac{-\overline{z}_0}{d} \end{pmatrix}$$

This rotation matrix A can be used in Equation 2 to minimize the sum of the epitope z coordinates. This rotation is performed with the robust molecular simulation program VMD [42] and implemented in the TCL script **`vmd_functions.tcl`** (in **`src`**). VMD performs this rotation in less than one second, while the original version of OptMAVEn required several minutes (depending on the antigen size) for the exhaustive search.

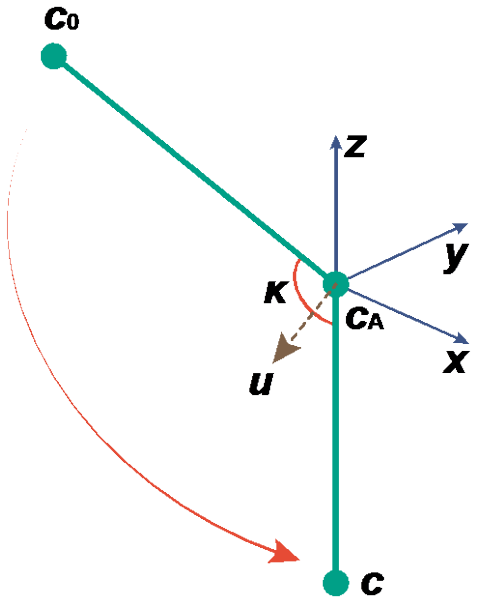

Figure 5: Schematic of the rotation of an antigen so as to minimize the z coordinates of the epitope. The rotation moves the initial geometric center of the epitope (an arbitrary point given by c_0) to a point c located along the negative z axis by rotating the antigen around a unit axis u ($c_0 \perp u \perp c$, $\|u\| = 1$) by an angle κ. The geometric center of the antigen (c_A) can be set, without loss of generality, to the origin. The x, y, and z unit axes are labeled as such.

Definition of antigen z angle

Original procedure: none

Problem: The z angle is not well defined and depends on the initial antigen structure.

After minimizing the epitope z coordinates, OptMAVEn generates an ensemble of antigen positions using translations on all axes and rotations around the z axis. However, the z angle θ_z is defined arbitrarily such that $\theta_z = 0°$ for the antigen position created by the minimization of the epitope z coordinates. This leads to three main problems: 1) two experiments that use different initial positions for the same antigen may have different antigen positions for the same θ_z; 2) it is impossible to determine θ_z given only the coordinates of an antigen position; 3) it is impossible to

generate the coordinates of an antigen position with a given θ_z from the coordinates of another position whose θ_z is unknown. Thus, Quick OptMAVEn defines θ_z as follows:

24)

$$\theta_z = \cos^{-1}\left(\frac{\text{proj}_{x,y}\, c_{A1,C\alpha} \cdot \vec{\imath}}{\left\|\text{proj}_{x,y}\, c_{A1,C\alpha}\right\| \|\vec{\imath}\|}\right) \cdot \text{sign}((\text{proj}_{x,y}\, c_{A1,C\alpha})_y)$$

$$= \cos^{-1}\left(\frac{(c_{A1,C\alpha})_x}{\sqrt{(c_{A1,C\alpha})_x{}^2 + (c_{A1,C\alpha})_y{}^2}}\right) \cdot \text{sign}((c_{A1,C\alpha})_y)$$

where $c_{A1,C\alpha}$ is the C-alpha atom of the first residue of the antigen; $\text{proj}_{x,y}\, c_{A1,C\alpha}$ is the projection of the coordinates of that atom onto the x, y plane; $\vec{\imath}$ is the unit vector in the positive x direction; and u_x and u_y denote, respectively, the x and y coordinates of a point u. Verbally, θ_z is the angle between the positive x axis and the C-alpha atom of the first residue of the antigen, when looking down on the x, y plane from the perspective of the positive z axis (Figure 6).

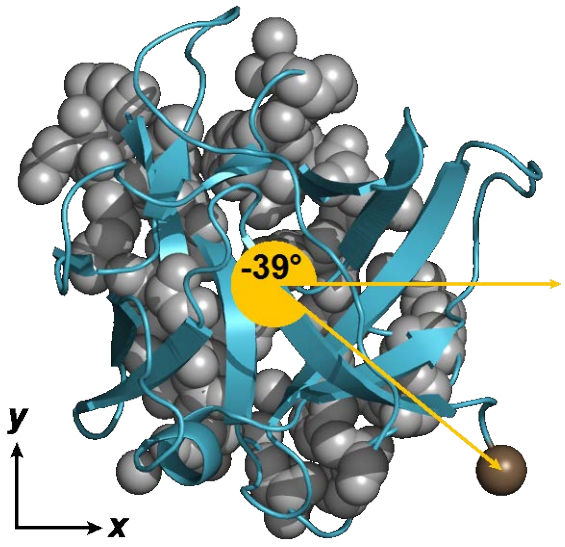

Figure 6: An illustration of θ_z. When the z coordinates of the epitope (grey spheres) are minimized, $\theta_z \in (-180°, 180°]$ is the angle between the positive x axis (rightward arrow) and the vector that leads from the antigen center of geometry (yellow circle) to the C-alpha atom of the first residue of the antigen (brown), when the x, y plane is in the plane of the page and the positive z axis extends toward the viewer. Here, $\theta_z = -39°$. The antigen shown in human interleukin 1-beta (PDB ID 46GM) [35]. The image was generated with PyMOL [36].

Grid search

Original procedure: Grid Search

Problem: A clash-permissive grid search increases cost of subsequent steps

The original version of OptMAVEn permitted up to two clashes between the framework antibody and the antigen before rejecting an antigen position. This criterion caused OptMAVEn to test an overly large number of positions using computationally expensive energy calculations in the next step. Quick OptMAVEn reduces the number of positions that must be tested by an average factor of $1 - 10^{-0.494} = 68\%$ (Table 9). While OptMAVEn implemented the grid search in a house-written script, Quick OptMAVEn uses the open-source software VMD [42] to reposition the antigen. VMD positions the antigen and checks for clashes more quickly, reducing the amount of time needed for the grid search. All performance comparisons are discussed thoroughly in "Direct comparison of OptMAVEn and Quick OptMAVEn on ten antigens."

Energy calculations

Original procedure: Energy calculations

Problem: Representation of antigen positions as PDB files increases disk storage requirement

OptMAVEn implements the energy calculations using a house-written script that requires a separate PDB file for each antigen position. As explained above, this requirement can impose a large burden on disk storage, particularly for large antigens such as Zika virus. To alleviate this burden, Quick OptMAVEn calculates energies with the NAMDEnergy plugin [43] from within VMD [42]. VMD is advantageous because, unlike the house-written script, it can rotate and

translate the antigen in memory without needing a PDB file of the pre-positioned antigen. Thus, only one reference antigen PDB file is needed, which can reduce the disk storage requirement by up to several gigabytes.

This new implementation causes a fundamental difference in the organization of energy calculations. OptMAVEn cannot reposition the antigen during the energy calculation step but can load all of the MAPs parts simultaneously: thus, it creates a directory for each antigen position, and the script within a given directory calculates the interaction energy between every MAPs part and the antigen at that position. Quick OptMAVEn can efficiently reposition the antigen during the energy calculation step but can load only one MAPs part at a time; thus, it creates a directory for each MAPs part, and the script within a given directory calculates the interaction energy between that MAPs part and the antigen in every position. Quick OptMAVEn can load only one MAPs part at a time because NAMDEnergy requires a protein structure file (PSF) specifying all atoms in the system. For each MAPs part, Quick OptMAVEn creates a PSF containing the atoms in the MAPs part and the antigen. To prevent these PSFs from occupying excessive disk storage, Quick OptMAVEn writes the PSF immediately before starting the energy calculations on a MAPs part and deletes the PSF immediately upon finishing the calculations. Because Quick OptMAVEn simultaneously calculates the energies of a limited number of MAPs parts (up to ~100, rather than 924), the PSF files do not occupy excessive disk storage.

Germline design

Original procedure: Germline design

Problem: Design of multiple antibodies for each antigen position does not increase design quality

OptMAVEn designs five antibodies for each antigen position, while Quick OptMAVEn designs only one. As described in "Germline design," OptMAVEn creates a series of five designs in order of decreasing predicted affinity. Because the first design has the greatest predicted affinity in each case, Quick OptMAVEn stops after creating this first design, which reduces the time needed to create the designs without sacrificing the affinity of the best design for each position.

Clustering of designs

Original procedure: none

Problem: Absence of a method to ensure diversity among designs

Quick OptMAVEn introduces a clustering procedure to maximize the diversity of the designs it selects for subsequent steps. The selected designs are subject to relaxation: relative to unrelaxed interaction energies, interaction energies of relaxed structures correlate more closely with and have been used as approximations of experimental binding affinities [44]. However, relaxing all of the designs generated during the MILP step (typically hundreds or thousands: see Table 10) is very computationally intensive. Thus, the goal is to predict which designs will have the most negative post-relaxation interaction energies before relaxing them.

OptMAVEn predicted post-relaxation energy solely on the basis of MILP (unrelaxed) energy. However, we have often observed designs whose MILP energies incorrectly predict their

post-relaxation energies. Furthermore, we assumed that the greater the diversity among designs that are predicted to have high affinities, the greater the likelihood that at least one of these designs will actually bind the antigen with high affinity. For example, suppose that the ten designs with the highest predicted affinities all have HV part 100 and HCDR3 part 300. If, in reality, these two parts interact unfavorably with the antigen in a way that Quick OptMAVEn cannot predict, then all of the top ten designs would have low affinities.

Quick OptMAVEn selects a feasibly small number of designs to relax, limits reliance on MILP energies, and prevents itself from selecting highly similar designs as follows. Each design is converted into a 23-dimensional vector that concatenates the x, y, and z coordinates of the antigen, $\cos\theta_z$ and $\sin\theta_z$ of the antigen, and a three-dimensional coordinate for each of the six MAPs parts. Subsequently, the designs are clustered using a *k*-means algorithm. Similar designs cluster together, and then the design with the lowest MILP energy is selected from each cluster. If more designs are needed, then the design with the second-lowest MILP energy is selected from each cluster, and so on. Thus, Quick OptMAVEn uses MILP energy to guide selection (within each cluster, designs are selected in order of ascending MILP energy), and it ensures diversity among designs (n designs must be selected from every cluster before $n + 1$ designs are selected from any cluster).

Creating coordinate vectors for the MAPs parts

The major challenge with this approach is simultaneously clustering the designs based on their antigen coordinates, which are numerical vectors, and their sets of six MAPs parts, which are not. We solved this challenge by converting each MAPs part into a three-dimensional coordinate

vector using a pre-processing step. We tried a number of approaches to convert the amino acid sequence of each MAPs part. For each approach, our strategy was to compute "distances" between each pair of MAPs parts within the same category and then embed these distances in Euclidean space by solving the distance geometry problem (DGP) [45]. In order for the distances to be embeddable in Euclidean space, which is a metric space, they must satisfy the following definition of a metric space M with an associated distance metric function d [45]:

25)

$$d(x,y) \geq 0 \; \forall \; x,y \in M$$

26)

$$d(x,y) = 0 \Leftrightarrow x = y \; \forall \; x,y \in M$$

27)

$$d(x,y) = d(y,x) \; \forall \; x,y \in M$$

28)

$$d(x,y) + d(y,z) \geq d(x,z) \; \forall \; x,y,z \in M$$

In this case, M is one category of MAPs parts (e.g. HV) and d is distance metric: a function that assigns a distance to each pair of parts in that category. We tested multiple distance metrics based on the normalized information distance proposed by Li et al. [46]. This distance is defined as follows:

29)

$$d(x,y) = \frac{\max\{K(x|y^*), K(y|x^*)\}}{\max\{K(x), K(y)\}}$$

where x and y are the two objects to compare (e.g. MAPs part sequences); x^* and y^* are the shortest programs capable of computing x and y, respectively, on a universal computer (e.g. a universal Turing machine); K is the Kolmogorov complexity function such that $K(x)$ equals the length of x^*, and $K(x|y)$ is the length of the shortest program capable of computing x given an

auxiliary input y [46]. Importantly, K is a metric, and so the distances between MAPs parts computed using Equation 29 are embeddable in Euclidean space (i.e. a Euclidean coordinate can be generated for each part such that the distances are satisfied).

However, K is not computable, which means that the exact Kolmogorov complexity $K(x)$ cannot be computed in finite time [46]. Nevertheless, $K(x)$ is upper semi-computable and can be approximated using a standard file compression program, such as **`gzip`**, under the assumption that the size of the compressed file x^* accurately estimates $K(x)$. Li et al. [46] note that this assumption is not necessarily valid and present an example of compressing a binary file x of the first 10^{23} bits of π. While x could be computed by a program x^* of a relatively small size (e.g 10^4 bits), most—if not all—file compressors would be unable to recognize the particular structure of π and would thus fail to produce an x^* significantly smaller than x. Neglecting such cases, Li et al. proposed the following equation to approximate the normalized information distance $NCD(x, y)$ in practice:

30)

$$NCD(x, y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}$$

where x and y are files that contain a representation of the objects whose distance is to be calculated (e.g. text files of the sequences of the MAPs parts), xy is the file produced by concatenating x and y, and $C(x)$ signifies the size of the compressed file x^* produced by using a compression program C to compress the file x.

We tested multiple variations of this procedure to calculate distances between MAPs parts. First, we created plain text files of the coordinates of the MAPs parts (one file per part, one space-delimited atomic coordinate per line) and computed the distances between the parts using Equation 30 with the compressor **`gzip`**. However, **`gzip`** appeared to perform poorly at compressing

redundant information in files in this format. In particular, the distances $NCD(x, x)$ (i.e. from each part x to itself) were significantly greater than their theoretical values of 0. Thus, we tested two alternate formats for representing the MAPs part coordinates: concatenating the coordinates for each part on one line and converting the coordinates to binary data. Both representations yielded distances similar to those of the first representation. Additionally, we tried explicitly writing the shortest program in Python capable of generating the coordinate files used in our first attempt to compare distances, but we were unable to prove that such programs were of minimal length. Therefore, we abandoned the structure-based normalized information distance in determining the distances between MAPs parts.

Instead, we adopted a sequence-based distance measure based on the BLOSUM62 matrix [47], which is commonly used to quantify the similarity between two protein sequences [48]. In order to quantify distances, not similarities, between sequences, we computed distance matrices using the following method adapted from Stojmirović [49].

First, let there be two amino acid sequences X and Y. Label each amino acid in X and Y with an integer such that these numbers increase monotonically but may have gaps and need not start at 1. Let A and B be, respectively, the sequences of amino acid numbers for X and Y. Let x_i be the amino acid numbered i in sequence X if $i \in A$ (i.e. if X has an amino acid numbered i); if not, then let x_i be a gap. Define y_i analogously. Then, the similarity score $S(X, Y)$ between two sequences X and Y was defined as follows:

31)

$$S(X, Y) = \sum_{i \in A \cup B} s(x_i, y_i)$$

where $s(x_i, y_i)$ is the BLOSUM62 similarity score between amino acids x_i and y_i if neither x_i nor y_i is a gap; and $s(x_i, y_i)$ is a gap penalty $-g$ if either x_i or y_i is a gap. Note that we did not know the optimal g *a priori*; we discuss later our method for finding the optimal g. It is never the case that both x_i and y_i are gaps, which would imply that $i \notin A \cup B$. The motivation for letting x_i denote the amino acid numbered i instead of the ith amino acid in X is that MAPs part sequences, by following the IMGT numbering system [12], may contain gaps and start at a number other than 1. For example, the alignment between $X =$ HCDR3-1 and $Y =$ HCDR3-400 is as follows:

$A = \{105,106,107,116,117\}$

$B = \{105,106,107,108,109,114,115,116,117\}$

$\boldsymbol{i}$	105	106	107	108	109	114	115	116	117
$\boldsymbol{x_i}$	Ala	Asn	Phe	(gap)	(gap)	(gap)	(gap)	Asp	Tyr
$\boldsymbol{y_i}$	Ala	Arg	Leu	Thr	Gly	Asn	Phe	Asp	Tyr
$\boldsymbol{s(x_i, y_i)}$	4	0	0	$-$g	$-$g	$-$g	$-$g	6	7

$S(X, Y) = 17 - 4g.$

It is possible to convert such similarity scores into quasi-metric distances if the following conditions hold [49] for every category $M_{ij} | \ i \in \{H, K, L\}, j \in \{V, CDR3, J\}$ of MAPs parts:

32)

$$S(X, X) \geq S(X, Y) \ \forall \ X, Y \in M_{ij}$$

33)

$$S(X, X) = S(X, Y) \wedge S(Y, X) = S(Y, Y) \Rightarrow X = Y \ \forall \ X, Y \in M_{ij}$$

34)

$$S(X, Y) + S(Y, Z) \leq S(X, Z) + S(Y, Y) \ \forall \ X, Y, Z \in M_{ij}$$

We verified these three conditions for all categories of MAPs parts. Five violations of condition 33 led us to discover five redundant MAPs parts: HV-135 = HV-136, KV-2 = KV-3, KV-25 = KV-26, KV-41 = KV-42, and LV-6 = LV-5. Thus, we removed from each redundant pair

the part with the larger number, reducing the total number of MAPs parts from 929 to 924. Thereafter, conditions 32, 33, and 34 held for all categories of MAPs parts. The similarity scores were converted into quasi-metric distances Q using the following [49]:

35)

$$Q(X,Y) = S(X,X) - S(X,Y)$$

Note that $Q(X,X) = S(X,X) - S(X,X) = 0$. This quasi-metric distance was then converted into its associated metric D_{align} [49]:

36)

$$D_{\text{align}}(X,Y) = \max\{Q(X,Y), Q(Y,X)\}$$

We verified that D_{align} satisfied the definition of a metric (Equations 25, 26, 27, and 28) for all categories of MAPs parts. However, we failed to embed any D_{align} matrix in Euclidean space using the analytical solution to the distance geometry problem [45] because every D_{align} matrix had at least one negative eigenvalue: thus, it is impossible to find Euclidean coordinates that satisfy exactly the distances specified in D_{align}.

There exist multiple software packages to solve distance geometry problems, e.g. MD-jeep [50], Xplor-NIH [51], TINKER [52], and DGSOL [53]. All of the aforementioned software was designed specifically to infer molecular coordinates from sparse and potentially erroneous nuclear magnetic resonance (NMR) data. Thus, they all embed into three-dimensional Euclidean space. In the general distance geometry problem, pairwise distances between N entities can be embedded in a space of dimensionality up to $N - 1$, assuming that the $N \times N$ distance matrix is positive semidefinite [45]. Thus, for MAPs categories with many members (e.g. HCDR3, $N = 428$), a three-dimensional embedding crushes $N - 1 - 3$ dimensions (424 for HCDR3). While reducing the dimensionality is likely to bring many distant parts closer than they should be, such

dimensionality reduction is routinely applied to high-dimensional data (e.g. via principal component analysis [54] and t-distributed stochastic neighbor embedding [55]) and could actually help mitigate the so-called "curse of dimensionality" that occurs when clustering sparse, high-dimensional data [56].

Among the aforementioned distance geometry solvers, DGSOL proved the most straightforward for us to install and was able to successfully embed the distances in three-dimensional space. DGSOL finds coordinates so as to minimize penalty function $p(D_{\text{embed}})$. DGSOL accepts a lower (LB_{ij}) and upper (UB_{ij}) bound on each pairwise distance $D_{\text{embed}_{ij}}$ produced by the embedding. If $LB_{ij} \leq D_{\text{embed}_{ij}} \leq UB_{ij}$, then $D_{\text{embed}_{ij}}$ does not contribute to the penalty function (i.e. $p_{ij} = p\left(D_{\text{embed}_{ij}}\right) = 0$). If $D_{\text{embed}_{ij}} < LB_{ij}$, then $p_{ij} = \left(\frac{{D_{\text{embed}_{ij}}}^2 - {LB_{ij}}^2}{{LB_{ij}}^2}\right)^2$; if $D_{\text{embed}_{ij}} > UB_{ij}$, then $p_{ij} = \left(\frac{{D_{\text{embed}_{ij}}}^2 - {UB_{ij}}^2}{{UB_{ij}}^2}\right)^2$. The overall penalty is:

37)

$$p(D_{\text{embed}}) = \sum_{i,j \in M_k} p_{ij}$$

where M_k is a category of MAPs parts.

We generated values for LB_{ij} and UB_{ij} using the alignment distances D_{align}. We did not know *a priori* the optimal width of the bounds, i.e. whether to let $LB_{ij} = D_{\text{align}_{ij}} = UB_{ij}$ or to use wider bounds ($LB_{ij} < D_{\text{embed}_{ij}} < UB_{ij}$). Thus, we tested multiple bounds given by the following equations:

38)

$$LB_{ij} = (1 - w) D_{\text{align}_{ij}}$$

39)

$$UB_{ij} = (1 + w)D_{\text{align}_{ij}}$$

where w is a width parameter. For each category M_k of MAPs parts, we varied w from 0 ($LB_{ij} = UB_{ij}$) to 0.5 in increments of 0.05. Note that $w = 0$ imposes the greatest penalty and $w = 0.5$ the least. At each level of w, we computed the Spearman rank correlation ρ_w between the flattened upper triangles of D_{align} and D_{embed}, as well as the root mean squared error $RMSE_w = \sqrt{\frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\left(D_{\text{embed}_{ij}} - D_{\text{align}_{ij}}\right)^2}{\frac{1}{2}(N^2-N)}}$. For the purposes of clustering, it is more important to perturb the ranks minimally (i.e. ρ_w is close to unity) during the embedding rather than to keep the $RMSE_w$ small. Thus, we let the optimal $w_{\text{opt}} = \underset{w}{\text{argmax}}(\rho_w)$. Ties were broken using $w_{\text{opt}} = \underset{w}{\text{argmin}}(RMSE_w)$. We used the coordinates generated at $w = w_{\text{opt}}$ as the definitive coordinates for each MAPs category. Recall that the above calculations require the gap penalty parameter g, which was unknown *a priori*. Thus, to find the optimal g, we performed all of the above steps (from computing D_{align} to finding w_{opt}) for $g = 4, 6, 8, 10$, and 12. Images of the coordinates generated for each MAPs category at each g are given in Figure 10. We computed $\rho_{w_{\text{opt}}}$ (Table 2) and $RMSE_{w_{\text{opt}}}$ (Table 3). Note that these values are independent of g for parts in the HJ, KJ, and LJ categories because the J parts do not contain any gaps.

Table 2: The values of $\rho_{w_{\text{opt}}}$ for each gap penalty and each MAPs category.

g	HV	HCDR3	HJ	LV	LCDR3	LJ	KV	KCDR3	KJ
4	0.932	0.804	0.982	0.987	0.818	1.000	0.934	0.910	0.996
6	0.935	0.831	0.982	0.988	0.838	1.000	0.935	0.922	0.996
8	0.939	0.855	0.982	0.987	0.852	1.000	0.939	0.931	0.996
10	0.935	0.774	0.982	0.986	0.839	1.000	0.921	0.894	0.996
12	0.948	0.891	0.982	0.987	0.875	1.000	0.941	0.946	0.996

Table 3: The values of $RMSE_{w_{\text{opt}}}$ for each gap penalty and each MAPs category.

g	HV	HCDR3	HJ	LV	LCDR3	LJ	KV	KCDR3	KJ
4	26.313	15.441	0.593	13.005	9.833	0.321	21.669	8.484	1.124
6	26.793	15.133	0.593	13.146	9.585	0.321	21.983	8.508	1.124
8	27.179	15.306	0.593	13.244	9.585	0.321	21.906	8.574	1.124
10	27.360	15.902	0.593	13.513	10.096	0.321	22.877	9.551	1.124
12	27.404	15.439	0.593	13.620	9.674	0.321	22.468	8.699	1.124

To determine the optimal g, for each part category (except the J parts), we ranked the levels of g from most to least optimal on the basis of $\rho_{w_{\text{opt}}}$ (higher is more optimal) and $RMSE_{w_{\text{opt}}}$ (lower is more optimal) (Table 4). We then computed the average rank of each g across both $\rho_{w_{\text{opt}}}$ and $RMSE_{w_{\text{opt}}}$ (Table 5). This analysis revealed that $g = 8$, with an average rank of 2.1, yielded the most optimal embeddings according to our criteria. Thus, we let $g = 8$ in our benchmarking tests for Quick OptMAVEn. However, the user has the option of selecting a different g from among the levels tested herein.

Table 4: The rank of each g level in terms of each criterion and each part. Rank 1 corresponds to the g level that optimized the criterion (i.e. maximized $\rho_{w_{\text{opt}}}$ or minimized $RMSE_{w_{\text{opt}}}$); rank 5 corresponds to the g level that yielded the least optimal value of the criterion.

Category	Criterion	Rank 1	Rank 2	Rank 3	Rank 4	Rank 5
HV	$\rho_{w_{\text{opt}}}$	12	8	6	10	4
HCDR3	$\rho_{w_{\text{opt}}}$	12	8	6	4	10
KV	$\rho_{w_{\text{opt}}}$	12	8	6	4	10
KCDR3	$\rho_{w_{\text{opt}}}$	12	8	10	6	4
LV	$\rho_{w_{\text{opt}}}$	12	8	6	4	10
LCDR3	$\rho_{w_{\text{opt}}}$	12	8	6	4	10
HV	$RMSE_{w_{\text{opt}}}$	4	6	8	10	12
HCDR3	$RMSE_{w_{\text{opt}}}$	6	8	12	4	10
KV	$RMSE_{w_{\text{opt}}}$	4	8	6	12	10
KCDR3	$RMSE_{w_{\text{opt}}}$	8	6	12	4	10
LV	$RMSE_{w_{\text{opt}}}$	4	8	6	12	10
LCDR3	$RMSE_{w_{\text{opt}}}$	4	6	8	12	10

Table 5: The average rank assigned to each g level according to the rankings in Table 4.

g	Average rank
4	3.2
6	2.7
8	2.1
10	4.7
12	2.4

k-*means clustering*

k-means clustering groups similar antibody designs on the basis of the antigen position (x, y, z, and θ_z) and the identities of the six MAPs parts in the antibody. As stated at the beginning of the section "Clustering of designs," this step enables Quick OptMAVEn to avoid selecting designs that are highly similar. Prior to clustering, for each design, a 23-dimensional vector concatenating the antigen's x, y, and z coordinates; $\cos(\theta_z)$ and $\sin(\theta_z)$; and the 3D coordinates representing the six MAPs parts. The designs are then clustered by clustering their corresponding vectors.

The *k*-means clustering algorithm was written in-house and implemented in the module **`kmeans.py`**. The following pseudocode describes how the algorithm functions:

```
k  # the number of clusters
vectors  # a list of vectors, each representing one design
max_iter  # the maximum number of iterations
centroids = random_choice(vectors, k)  # randomly choose k vectors without
replacement to be the initial cluster centroids
iter = 1  # number of iterations
converged = False  # Have the centroids converged (stopped moving)?
convergence_limit  # maximum mean square movement to be considered not moving
while not converged and iter <= max_iter:
    clusters = empty_list(length=k, data_type=set)  # a list of clusters; each
            # cluster is a set of vectors in the cluster
    # Assign each vector to the cluster with the closest centroid.
    for vector in vectors:
        index = argmin(distance(vector, centroids))
        clusters[index].add(vector)
    # If there are any empty clusters, move a random vector from another cluster
            # into each empty cluster.
    for cluster in clusters:
        if cluster.is_empty():
            other_cluster = random_choice(clusters, 1)  # choose one cluster
            # Ensure the other cluster can relinquish a vector without becoming
                    # empty
            while other_cluster.size() > 1
                other_cluster = random_choice(clusters, 1)
            # Transfer one random vector from the other cluster.
            vector = random_choice(other_cluster, 1)
            other_cluster.remove(vector)
            cluster.add(vector)
    # Move each cluster's centroid to the mean coordinate of the cluster's
            # vectors.
    movement_distances = empty_list(length=k, data_type=numeric)
    for index in {1, 2, ..., k}:
        mean_coordinate = mean(clusters[index])
        movement_distances[index] = distance(mean_coordinate, centroids[index])
        centroids[index] = mean_coordinate
    if square_root(sum(movement_distances^2)) <= convergence_limit:
        converged = True
    iter += 1
return clusters
```

Figure 7: Pseudo-code for implementing the *k*-means clustering step. The full implementation is located in the module **`kmeans.py`**.

Note that in the above implementation, the value of k is fixed. The optimal value of k is unknown *a priori*. In order to find the optimal value of k, the algorithm calculates the maximum variance ${\sigma^2}_k$ of the data in each cluster using the following equation:

40)

$$\sigma^2{}_k = \max\left\{\frac{\sum_{j=1}^{n_i}\left(\left\|v_{ij} - \bar{c_i}\right\|\right)^2}{n_i} | i \in C_k\right\}$$

where C_k is the set of clusters for a given value of k, n_i is the number of vectors in cluster i, $\bar{c_i}$ is the centroid of cluster i, and v_{ij} denotes vector j of cluster i. The $\sigma^2{}_k$ value measures the efficacy of the clustering—specifically the maximum intra-cluster variance: superior clustering will yield a lower $\sigma^2{}_k$. For each $k > 1$, the ratio $\frac{\sigma^2{}_k}{\sigma^2{}_1}$ was calculated; the optimal k was chosen to be the minimum k for which $\frac{\sigma^2{}_k}{\sigma^2{}_1} \leq t$, where $0 < t < 1$ is a pre-defined threshold. We used $t = 0.2$. The maximum possible value for k is n_v (where n_v is the total number of vectors being clustered) because when $k = n_v$, there is exactly one vector in each cluster, so each centroid must equal its corresponding vector ($v_{i1} = \bar{c_i}\ \forall\ i$), thus $\sigma^2{}_{n_v} = 0 < t$.

Selection of clustered designs

Quick OptMAVEn select a diverse set of antibodies with high predicted affinities in the following manner. Let E_{ij} be the interaction energy (see Equation 3) of antibody j in cluster i (denoted C_i). Let $E_{i,\min} = \min\{E_{ij} | j \in C_i\}$, i.e. the interaction energy of the antibody in cluster i with the most negative interaction energy (greatest predicted affinity). Quick OptMAVEn ranks the C_is from most to least negative $E_{i,\min}$. Indexing along i, in order of increasing $E_{i,\min}$, it selects from each C_i the antibody j for which $j = \operatorname{argmin}\{E_{ij} | j \in C_i\}$, i.e. the antibody with the most negative interaction energy from cluster i. Quick OptMAVEn stops when the number of selected antibodies n_{selected} reaches a user-specified goal n_{Abs}. By default, $n_{\text{Abs}} = 30$. If $n_{\text{selected}} = n_{\text{Abs}}$

before an antibody has been selected from each of the k clusters (i.e. $k > n_{\mathrm{Abs}}$), then no antibody is selected from any of the remaining $n_{\mathrm{Abs}} - k$ clusters.

If, after selecting one antibody from each cluster, an insufficient number of antibodies have been selected (i.e. $n_{\mathrm{selected}} = k < n_{\mathrm{Abs}}$), then Quick OptMAVEn repeats the above steps, starting by re-ranking the C_is from most to least negative $E_{i,\mathrm{min}}$, but ignoring any antibodies that have already been selected. This procedure is repeated either until $n_{\mathrm{selected}} = n_{\mathrm{Abs}}$ or, if the total number of antibodies designed is less than n_{Abs}, until all antibodies have been selected.

After selecting these antibodies, Quick OptMAVEn performs a structural relaxation on every antigen-antibody complex using CHARMM. The relaxed structures are saved in both PDB and FASTA format in the **`Results`** directory within the experiment's directory.

Benchmarking of Quick OptMAVEn

To compare the performances of OptMAVEn and Quick OptMAVEn, we used both programs to design antibodies for 10 antigens and used Quick OptMAVEn to design antibodies for an additional 54 antigens that had been tested previously with OptMAVEn [7]. During the benchmarking we used the default parameters of Quick OptMAVEn (Table 6). The antigen chains and epitopes are given in Table 12. The computations were performed on the (now decommissioned) Lion-XF cluster at the Pennsylvania State University.

Table 6: The setup and parameters used for the benchmarking of Quick OptMAVEn.

Software	Version	CHARMM settings	Value	*k*-means settings	Value	Grid search settings	Value
Python	2.7.13	CHARMM energy terms	angl, bond, dihe, elec, impr, urey, vdw, gbener	Gap penalty	8	MAPs clash cutoff	1.25 Å
CHARMM	34b1	CHARMM iterations	5000	*k*-means max iterations	1000	$x_{\min}$, $x_{\max}$, x_{step}	-10.0, 5.0, 2.5 Å
VMD	1.9.3			*k*-means tolerance	0.01	$y_{\min}$, $y_{\max}$, y_{step}	-5.0, 10.0, 2.5 Å
NAMD	2.12			*k*-means threshold	0.20	$z_{\min}$, $z_{\max}$, z_{step}	3.75, 16.25, 2.25 Å
						$\theta_{z_{\min}}$, $\theta_{z_{\max}}$, $\theta_{z_{\text{step}}}$	0, 300, 60°

Direct comparison of OptMAVEn and Quick OptMAVEn on ten antigens

We had the computational resources to directly compare the performance of OptMAVEn and Quick OptMAVEn on ten antigens. We randomly selected ten of the antigens against which OptMAVEn had previously designed antibodies [7]: 1NSN, 2IGF, 2R0W, 2VXQ, 2ZUQ, 3BKY, 3FFD, 3G5V, 3L5W, and 3MLS. Performance was assessed on the basis of the times taken for the Initial antigen positioning plus the Grid search (T_{pos}; we were able to measure only the sum of these times), the Energy calculations (T_{ener}), the Germline design (T_{MILP}), and the total CPU time ($T_{\text{CPU}} = T_{\text{pos}} + T_{\text{ener}} + T_{\text{MILP}}$); the maximum amount of disk storage used at any point ($D_{\max}$); and the interaction energy between the antigen and the antibody with the most negative predicted interaction energy ($E_{\min}$). Additionally, we recorded the number of positions sampled (N_{pos}) and energy obtained from the MILP (E_{MILP}; i.e. the value of Equation 3).

The results are given in Table 7, Table 8, and Table 9. Quick OptMAVEn performed significantly better ($P < 0.05$) in every performance measure except for $E_{\min}$ and E_{MILP}. For $E_{\min}$, the programs yielded equivalent results ($P = 0.79$). Note that E_{MILP} is a temporary energy used to select designs for which to calculate the more computationally intensive quantity $E_{\min}$. Thus, the significantly worse performance of Quick OptMAVEn in terms of E_{MILP} does not suggest that the final, relaxed designs are worse. Thus, these results indicate that, relative to OptMAVEn, Quick OptMAVEn designs antibodies of equivalent affinities using approximately $10^{-0.591} = 26\%$ of the CPU time and $10^{-0.788} = 16\%$ of the disk storage on average.

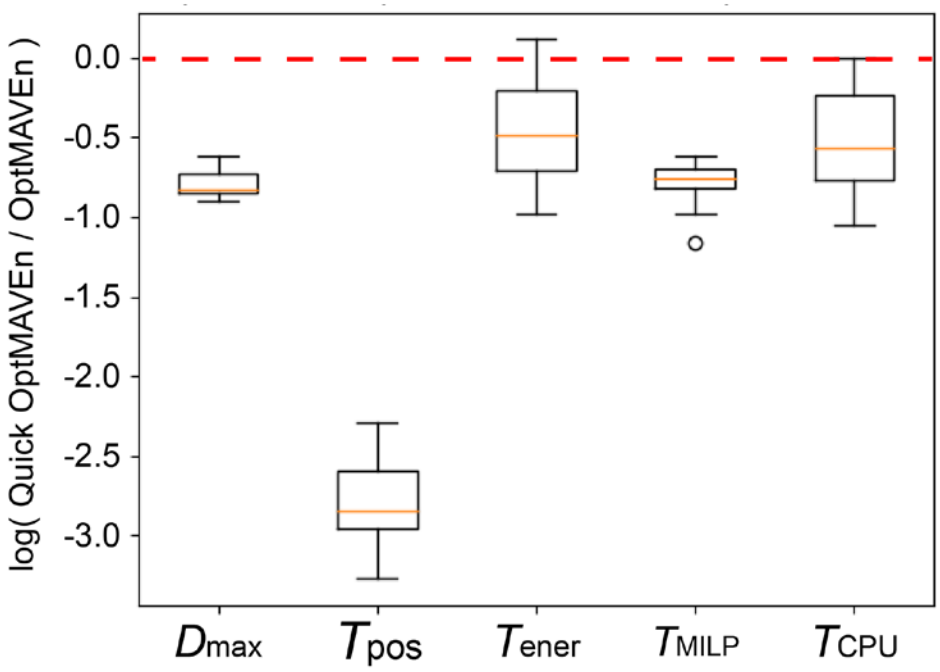

Figure 8: Quick OptMAVEn performs significantly better than OptMAVEn in terms of $D_{\max}$, T_{pos}, T_{ener}, T_{MILP}, and T_{CPU}. Values are base-10 logarithms of the ratio of the Quick OptMAVEn and OptMAVEn performance measures. Box plots report the minimum, Q1, median (orange lines), Q3, and maximum values.

Table 7: The performance of OptMAVEn. Times are in hours, sizes in megabytes, and energies in kcal/mol.

Antigen	T_{pos}	T_{ener}	T_{MILP}	T_{CPU}	D_{max}	E_{min}	E_{MILP}	N_{pos}
1NSN	32.7	214.2	26.8	273.7	1004	-658.7	-850.1	2428
2IGF	2.1	20.0	26.4	48.4	820	-76.4	-383.9	3023
2R0W	2.0	17.8	20.2	40.0	779	-277.0	-480.5	2955
2VXQ	26.1	174.4	19.6	220.1	970	-174.5	-576.4	2711
2ZUQ	41.6	290.9	18.8	351.4	1094	-346.0	-363.6	2645
3BKY	5.0	54.8	33.7	93.5	824	-216.1	-356.4	3035
3FFD	5.3	35.0	19.5	59.8	657	+576.6	-397.4	2347
3G5V	22.0	33.1	20.8	75.9	808	-309.9	-413.0	2976
3L5W	29.6	173.9	24.4	227.9	1008	-281.4	-698.2	2798
3MLS	5.8	53.0	21.9	80.7	809	-249.6	-395.1	2903

Table 8: The performance of Quick OptMAVEn. Times are in hours, sizes in megabytes, and energies in kcal/mol.

Antigen	T_{pos}	T_{ener}	T_{MILP}	T_{CPU}	D_{max}	E_{min}	E_{MILP}	N_{pos}
1NSN	0.036	22.3	1.8	24.2	142.4	-438.1	-255.6	442
2IGF	0.009	26.1	5.6	31.7	169.7	-118.5	-106.6	1374
2R0W	0.010	22.4	4.9	27.4	152.9	-127.9	-186.0	1204
2VXQ	0.033	33.7	3.6	37.4	135.4	-235.3	-185.0	893
2ZUQ	0.046	40.4	3.2	43.6	167.3	-131.3	-99.4	774
3BKY	0.011	33.9	6.7	40.6	197.4	-208.4	-77.7	1647
3FFD	0.014	10.9	2.0	13.0	83.8	+92.6	-153.9	492
3G5V	0.012	21.0	4.2	25.2	137.6	-458.5	-185.0	1035
3L5W	0.033	36.4	3.8	40.2	144.7	-394.0	-169.7	910
3MLS	0.009	18.0	3.3	21.3	114.7	-171.2	-229.4	807

Table 9: Comparison of the performance of OptMAVEn and Quick OptMAVEn. For $E_{\min}$ and E_{MILP}, entry ij is (Table 8_{ij} − Table 7_{ij}) because the interest is the difference in energy. For all other measures, entry ij is $\log_{10}\left(\frac{\text{Table } 8_{ij}}{\text{Table } 7_{ij}}\right)$ because the interest is the ratio of the values. For all measures, negative values show better performance for Quick OptMAVEn. Statistics are as follows: Shapiro, *P*-value of Shapiro-Wilk test for normality of the measure: because all $P > 0.05$, the assumption that all differences are normally distributed is supported; mean, mean difference between Quick OptMAVEn and OptMAVEn; s. d., standard deviation of measurements; *P*-value, two-tailed t-test of H_0: $\mu = 0$, H_A: $\mu \neq 0$, $P < 0.05$ in **bold**.

Antigen	T_{pos}	T_{ener}	T_{MILP}	T_{CPU}	$D_{\max}$	$E_{\min}$	E_{MILP}	N_{pos}
1NSN	-2.96	-0.982	-1.162	-1.053	-0.848	+220.6	+594.5	-0.740
2IGF	-2.35	+0.116	-0.674	-0.184	-0.684	-42.1	+277.2	-0.342
2R0W	-2.29	+0.102	-0.613	-0.165	-0.707	+149.1	+294.5	-0.390
2VXQ	-2.90	-0.714	-0.732	-0.770	-0.855	-60.8	+391.4	-0.482
2ZUQ	-2.95	-0.857	-0.774	-0.906	-0.815	+214.6	+264.3	-0.534
3BKY	-2.65	-0.208	-0.700	-0.362	-0.620	+7.7	+278.7	-0.265
3FFD	-2.58	-0.505	-0.984	-0.663	-0.895	-484.0	+243.5	-0.679
3G5V	-3.27	-0.198	-0.698	-0.479	-0.769	-148.6	+228.1	-0.459
3L5W	-2.95	-0.680	-0.806	-0.753	-0.843	-112.6	+528.6	-0.488
3MLS	-2.80	-0.469	-0.823	-0.578	-0.848	+78.4	+165.7	-0.556
Shapiro	6.0E-01	5.8E-01	1.0E-01	8.2E-01	1.8E-01	2.8E-01	6.9E-02	9.4E-01
mean	-2.77	-0.440	-0.797	-0.591	-0.788	-17.8	+326.7	-0.494
s. d.	0.303	0.383	0.164	0.296	0.090	209.4	137.0	0.145
P-value	**3.5E-10**	**5.5E-03**	**9.2E-08**	**1.4E-04**	**5.0E-10**	7.9E-01	**3.5E-05**	**1.9E-06**

Test of Quick OptMAVEn on 54 additional antigens

We used Quick OptMAVEn to design antibodies against 54 additional antigens for which OptMAVEn had previously been used to design antibodies (Table 10). In addition to the performance measures for the Direct comparison of OptMAVEn and Quick OptMAVEn on ten antigens, we report here the numbers of residues and atoms in each antigen.

Table 10: The performance of Quick OptMAVEn on 54 additional antigens, as well as those from the Direct comparison of OptMAVEn and Quick OptMAVEn on ten antigens (the first ten entries) and the Zika virus E protein (from 5GZN). Every antigen comprised one chain. N_{res}: number of residues in the antigen; N_{res}: number of atoms in the antigen; N_{pos}, T_{CPU}, $D_{\max}$, and $E_{\min}$ are defined above in the Direct comparison of OptMAVEn and Quick OptMAVEn on ten antigens.

Antigen	N_{res}	N_{atom}	N_{pos}	T_{CPU}	$D_{\max}$	$E_{\min}$
1NSN	138	2262	442	28.1	142.4	-1045.4
2IGF	7	125	1374	33.7	169.7	-376.3
2R0W	7	112	1204	29.3	152.9	-634.7
2VXQ	92	1430	893	40.3	135.4	-674.8
2ZUQ	148	2390	774	47.2	167.3	-404.8

3BKY	17	242	1647	42.8	197.4	-228.1
3FFD	18	328	492	15.2	83.8	-730.2
3G5V	16	229	1035	27.1	137.6	-447.6
3L5W	101	1577	910	43.5	144.7	-566.1
3MLS	20	288	807	23.3	114.7	-509.8
1ACY	10	156	1558	40.9	188.3	-370.6
1CE1	8	93	1694	44.3	200.9	-513.3
1CFT	5	84	1554	38.9	187.9	-253.5
1DZB	129	1958	749	42.2	136.3	-775.8
1EGJ	101	1643	650	34.1	106.9	-618.6
1F90	9	156	1328	35.0	165.8	-377.5
1FPT	11	162	1478	38.4	180.0	-455.6
1HH6	11	159	718	20.8	104.6	-385.5
1I8I	9	142	1480	38.4	179.7	-350.8
1JHL	129	1962	985	53.7	132.3	-766.6
1JRH	95	1491	397	21.9	99.6	-541.4
1KC5	8	119	1299	36.8	162.1	-376.1
1KIQ	129	1968	730	41.4	119.1	-750.1
1MLC	129	1968	618	35.9	111.2	-752.0
1N64	16	241	990	28.1	132.9	-386.6
1NAK	10	166	1192	41.5	154.1	-393.3
1OBE	13	195	417	13.5	77.9	-397.0
1ORS	132	2146	1001	55.7	162.4	-625.5
1PZ5	8	124	1348	34.1	167.4	-419.5
1QNZ	18	301	575	18.5	91.4	-367.3
1SM3	9	126	1354	34.8	167.9	-454.2
1TQB	102	1659	489	26.8	104.1	-534.6
1V7M	145	2258	588	37.5	115.4	-561.0
1XGY	6	85	1811	45.4	212.8	-293.1
1ZA3	91	1346	71	7.5	91.8	-758.7
2A6I	9	136	1093	29.1	141.8	-365.2
2BDN	68	1106	810	35.2	115.1	-740.6
2DQJ	129	1968	590	34.0	111.6	-852.4
2FJH	98	1565	312	18.4	99.7	-528.8
2H1P	11	182	561	17.0	90.4	-355.0
2HH0	9	151	1062	28.6	140.0	-282.7
2HRP	10	177	1013	27.9	135.4	-366.5
2IFF	129	1966	595	33.9	126.7	-594.4
2JEL	85	1293	596	28.1	101.9	-539.5

2OR9	11	181	734	21.1	106.7	-387.8
2QHR	11	185	761	20.3	111.3	-340.2
2R29	97	1553	641	33.2	105.3	-698.4
3AB0	136	1955	380	23.9	107.8	-765.0
3BDY	95	1521	779	36.3	133.9	-439.7
3CVH	8	142	1168	30.7	149.6	-333.7
3D85	133	2074	441	27.9	109.8	-717.0
3E8U	11	136	1481	38.1	180.8	-431.4
3ETB	144	2332	296	21.8	111.3	-898.6
3F58	11	136	1317	34.6	168.5	-322.6
3G6D	106	1667	418	24.2	103.2	-876.8
3GHB	10	146	1341	33.5	166.7	-383.4
3GHE	15	255	773	26.9	112.2	-430.1
3HR5	9	142	1340	38.4	166.5	-478.7
3KS0	92	1443	1148	54.3	148.0	-578.5
3MLX	14	235	621	20.5	94.7	-367.7
3NFP	124	1909	292	19.7	104.5	-771.6
3P30	84	1437	32	4.7	65.2	-714.9
3QG6	6	105	1425	36.1	175.2	-362.4
3RKD	146	2185	776	46.1	124.5	-793.7
5GZN	402	6081	77	18.9	176.7	-1244.8

To understand the factors affecting T_{CPU} and $D_{\max}$, we investigated correlations between these performance measures and the inputs. We excluded 5GZN from the analysis because its $N_{\mathrm{res}} = 402$ and $N_{\mathrm{atom}} = 6081$ were outliers. Among the other antigens, the Pearson correlation r between T_{CPU} and both N_{res} ($r = 0.144$) and N_{atom} ($r = 0.136$) was small, indicating that the size of the antigen did not affect T_{CPU}. Correlations between $D_{\max}$ and N_{res} ($r = -0.391$) and N_{atom} ($r = -0.396$) were also modest. However, there were stronger correlation between N_{pos} and both $D_{\max}$ ($r = 0.928$) and T_{CPU} ($r = 0.634$), suggesting that one of the most important factors in the performance is the number of positions sampled. Of note, there appeared to be two clusters in each plot: these clusters separated perfectly by partitioning the antigens into those small ($N_{\mathrm{res}} \leq 50$) and large ($N_{\mathrm{res}} > 50$). There were no substantial differences in correlations after

partitioning. Thus, it seems that the most significant predictor of T_{CPU} and $D_{\max}$ is N_{pos}, which is determined by the antigen shape as well as the size and grid search settings.

Design of 77 antibodies for Zika E protein, including 9 predicted to be superior to native

We used Quick OptMAVEn to design 77 antibodies that targeted the Zika envelope (E) protein. We tested two Zika epitopes that we identified in PDBs 5GZN and 5GZO [16]. Both PDBs comprise several complexes of Zika E protein bound to neutralizing antibodies that were isolated from an infected patient. We used chain A as the antigen for both 5GZN and 5GZO. We defined the epitope as the set of all residues of chain A for which at least one atom lay within 4.0 Å of any atom in the antibody complexed with chain A. The epitope residues were as follows. 5GZN: 46, 47, 52, 136, 138, 140, 156, 158, 159, 166, 168, 276, 277, 278, 279, 280, 281, 283. 5GZO: 64, 65, 66, 67, 68, 69, 84, 87, 89, 90, 118, 119, 120, 233, 252. Note that if no structures of Zika in complex with an antibody had been available, we could have predicted these epitopes using existing software [33]. We used the default settings for Quick OptMAVEn (Table 6).

Quick OptMAVEn successfully designed 77 antibodies against the 5GZN epitope but failed to find any non-clashing positions for the epitope of 5GZO. However, using a different grid could enable Quick OptMAVEn to find non-clashing positions for this epitope as well. By default, Quick OptMAVEn clusters the designs and then relaxes and calculates $E_{\min}$ for the top 30 designs. In order to evaluate the clustering algorithm, we computed the relaxed interaction energies of all 77 designs. We compared retaining the top 30 identified by the Quick OptMAVEn clustering algorithm to retaining the top 30 on the sole basis of MILP energy (the OptMAVEn procedure). We defined a true positive (TP) as a retained design with a relaxed energy in the top 30 and a false

positive (FP) as a retained design whose relaxed energy was not in the top 30. Likewise, a true negative (TN) was a discarded design whose relaxed energy was not in the top 30, and a false negative (FN) was a discarded design whose relaxed energy was in the top 30. We found that Quick OptMAVEn yielded more TPs (15 vs 8) and TNs (32 vs 25) than did the OptMAVEn procedure, as well as a higher Matthews Correlation Coefficient (MCC) [57] of 0.18 vs -0.20. These results indicate that the clustering algorithm is superior to the method of OptMAVEn.

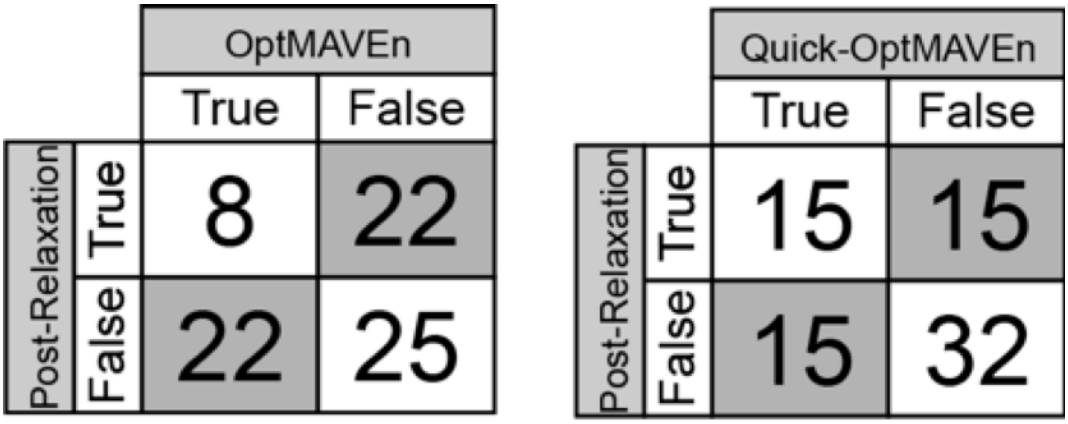

Figure 9: The confusion matrices for the OptMAVEn and Quick OptMAVEn procedures of selecting designs. Designs among the top 30 in terms of relaxed energy fall into Post-Relaxation True: otherwise, False. Designs retained by OptMAVEn and Quick OptMAVEn fall into OptMAVEn and Quick-OptMAVEn True, respectively: otherwise, False.

For the 5GZN epitope, 9 of the 77 antibodies (12%) had a value for $E_{\min}$ more negative than the interaction energy of the native 5GZN antibody, meaning that their affinities for Zika E protein were predicted to be greater than that of an antibody isolated from a human patient (Table 13). Furthermore, the sum of the humanization scores [7] (a measure of immunogenicity) of the light and heavy chains for four of these antibodies (11, 19, 39, and 1) were less than that of the native human antibody in 5GZN, suggesting that these antibodies are unlikely to elicit an immune response. However, these promising results await experimental validation.

Chapter 4

Future directions and conclusion

This thesis describes the development of Quick OptMAVEn, a new version of the OptMAVEn [7] framework for *de novo* mAb design. During benchmarking, Quick OptMAVEn took an average of 74% less time and used 84% less storage on disk than its predecessor. To the best of my knowledge, OptMAVEn is the only published software capable of designing entire variable domains (V_H and V_L) of mAbs *de novo*, i.e. without requiring a structure of an antigen-antibody complex. The ability to design antibodies *de novo* could offer an enormous advantage in time-critical scenarios, such as an outbreak of a disease. For example, during the recent Zika epidemic, the structure of the Zika E protein [31] was published over six months before the identification of the first human antibodies capable of neutralizing Zika [17].

OptMAVEn failed to design mAbs for Zika during this time interval due to excessive CPU time and disk storage requirements. However, Quick OptMAVEn successfully designed 50 mAbs for Zika E protein within 24 hours of real time (albeit not CPU time) on the ACI-b supercomputing cluster at Pennsylvania State University. Nine designs are predicted to bind with greater affinity than the natural antibody (PDB ID: 5GZN [16]) isolated from an infected human. Our collaborator, Klaus Schulten (whose lab developed VMD [42] and NAMD [43]), had planned to validate our designs experimentally but tragically passed away in 2016. Thus, an important future step is to measure the affinities of our designs for Zika E protein. If our designs do bind Zika, comparing predicted and experimentally-determined structures of their antigen-antibody complexes would

reveal if Quick OptMAVEn can not only design high-affinity antibodies but also predict atomistic interactions between the antigen and antibodies.

Despite the significant improvements of Quick OptMAVEn over OptMAVEn, several challenges remain for the future development of software based on OptMAVEn. Quick OptMAVEn currently does not explicitly account for stability when designing mAbs, which is considered or even optimized by other software (e.g. *AbDesign* [22] and Rosetta Antibody Design [23]). Quick OptMAVEn does prevent clashes between the six MAPs parts (which would destabilize the antibodies) by incorporating a pre-computed set of clashing parts into the MILP. However, there is no means to predict—much less optimize—thermostability, propensity to aggregate, or shelf life of the mAbs, which are all clinically relevant indicators of stability [19]. Potentially, the MILP could be modified to incorporate the interaction energies among the MAPs parts—not just their interaction energies with the antigen—which could simultaneously optimize stability and affinity. Alternatively, the MILP could remain unchanged, but designs below a certain stability threshold would be filtered out after the MILP step.

Another limitation of Quick OptMAVEn is that it uses a substantially simplified energy function to improve speed and throughput. During the MILP step, Quick OptMAVEn assumes that the sum of the interaction energies of the MAPs parts with the antigen will accurately predict experimental affinity. As Li et al. [7] state about OptMAVEn, this simplification ignores entropy, which may be significant for antigens that are especially flexible. Moreover, it ignores the potential to alleviate steric clashes during the structural relaxation following the assembly of the variable domains. Because NAMDEnergy [43] does not support solvation, the interaction energies are calculated using only electrostatic and van der Waals energy terms, which could overestimate the interaction energies between charged residues and underestimate interactions between

hydrophobic residues. Generalized Born solvation energies are incorporated into the energy calculations (using CHARMM [34]) on the assembled variable domains. We can see no straightforward solutions to these limitations in the energy function without abandoning NAMDEnergy, which is a primary contributor to the performance of Quick OptMAVEn. A potential solution is to remove these limitations by re-implementing them in customized C++ code, as was done in OptMAVEn. However, a major disadvantage to this approach is that it would increase the difficulty of keeping Quick OptMAVEn up to date with changes in file formats or energy functions.

Currently, Quick OptMAVEn is configured to run on only the ACI-b supercomputing cluster at Pennsylvania State University. We could take two approaches to enable other labs to use Quick OptMAVEn. One approach would be to share the source code on the website of the Maranas lab (http://www.maranasgroup.com/) or on GitHub. However, Quick OptMAVEn depends on CHARMM [34], which is available only through a commercial license, and so we cannot assume that any lab would be able to run Quick OptMAVEn on its own computers. Thus, we plan to establish a web server with which anyone may submit jobs to Quick OptMAVEn. These jobs will be executed on the ACI-b cluster or another machine in our lab that is equipped with the necessary software. It is our hope that Quick OptMAVEn, as well as any future versions of OptMAVEn, will facilitate the computational design of therapeutic mAbs, which hold promise to treat a wide variety of diseases that have so far remained intractable.

Chapter 5 Appendix A

Clustering procedure supplementary information

Table 11: The BLOSUM62 similarity matrix for amino acids. Amino acids are shown using standard one-letter abbreviations.

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4

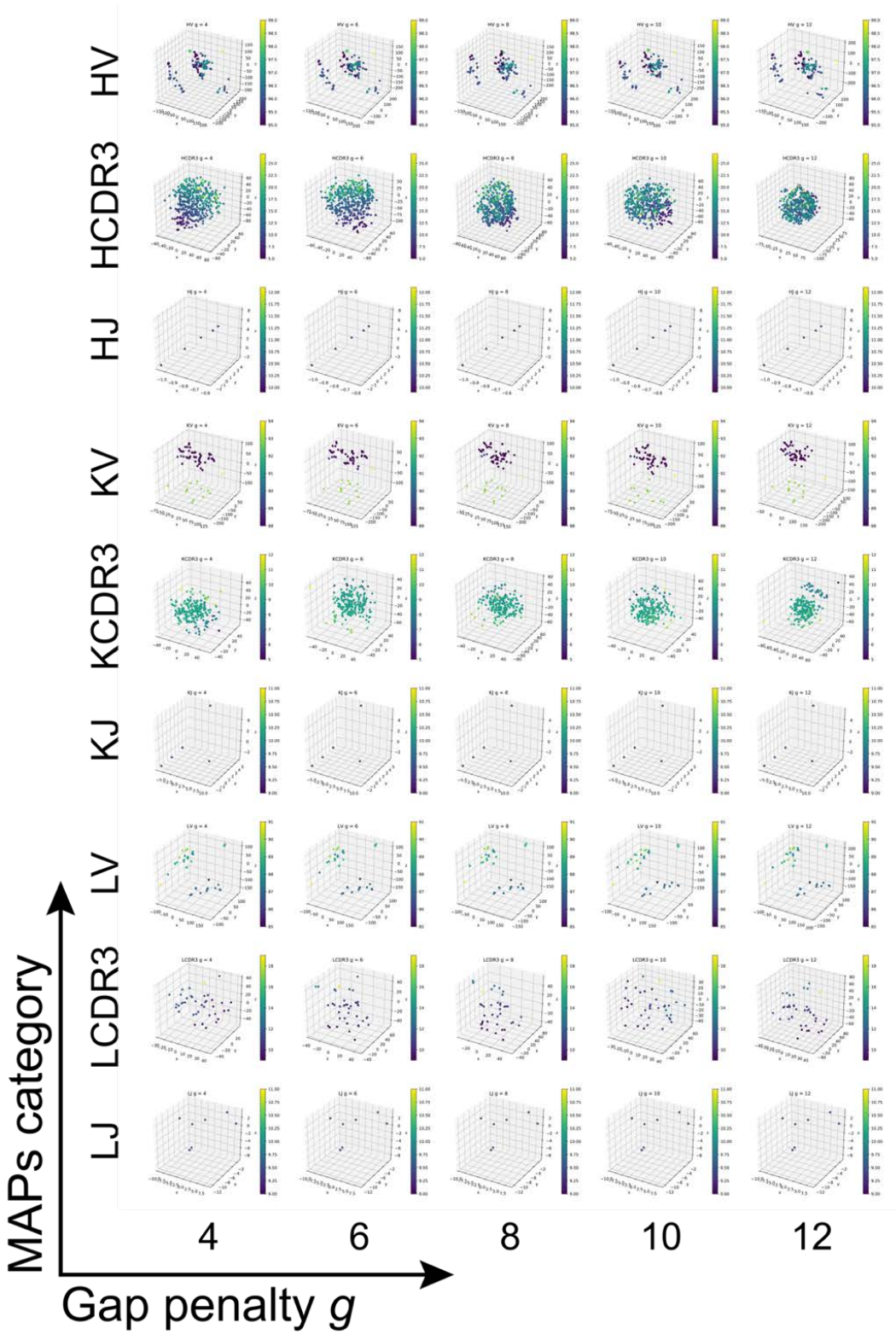

Figure 10: The embedded coordinates for each category of MAPs parts at each gap penalty g. Colors represent the number of amino acids in each MAPs part. Coordinates for the J parts are independent of g because the J parts do not have gaps.

Chapter 6 Appendix B

Inputs and outputs of Quick OptMAVEn

Table 12: The chains and epitopes of the 64 antigens used in benchmarking Quick OptMAVEn.

PDB	Antigen Chain	Heavy Chain	Light Chain	Epitope
1NSN	S	H	L	57, 60, 61, 64, 68, 70, 93, 95, 96, 97, 98, 105, 106, 120, 121, 123, 124, 127
2IGF	P	H	L	70, 71, 72, 73, 74, 75
2R0W	Q	H	L	3, 4, 5, 6, 7, 8
2VXQ	A	H	L	32, 34, 39, 40, 41, 43, 67, 68, 69, 74, 75, 76, 77, 78, 79, 80
2ZUQ	A	C	B	95, 96, 97, 98, 99, 100, 101, 132, 133, 134, 141
3BKY	P	H	L	169, 170, 171, 172, 173, 174, 175
3FFD	P	A	B	16, 17, 19, 20, 21, 23, 24, 26, 27, 28, 30, 31
3G5V	C	B	A	288, 289, 293, 296, 297, 298, 299, 300, 301, 302
3L5W	I	H	L	14, 15, 101, 104, 105, 107, 108, 109
3MLS	P	H	L	3, 5, 6, 10, 11, 12, 13, 14, 15, 16, 17, 18
1ACY	P	H	L	316, 319, 320, 321, 322, 323, 324
1CE1	P	H	L	0, 1, 2, 3, 4, 5, 6, 7
1CFT	C	B	A	1, 2, 3, 4, 5
1DZB	X	A	A	20, 21, 23, 62, 63, 73, 75, 96, 97, 98, 100, 101, 102, 103, 104, 106, 112, 116
1EGJ	A	H	L	362, 363, 364, 365, 366, 367, 395, 416, 417, 418, 419, 421
1F90	E	H	L	1, 2, 3, 4, 5, 6, 7, 8
1FPT	P	H	L	96, 97, 98, 99, 100, 101, 102, 103
1HH6	C	B	A	2, 3, 4, 5, 6, 7, 8, 9, 10, 11
1I8I	C	B	A	502, 503, 505, 506, 507, 508, 509
1JHL	A	H	L	21, 22, 23, 102, 103, 104, 106, 111, 112, 113, 116, 117, 118, 119, 121
1JRH	I	H	L	47, 49, 50, 51, 52, 53, 54, 55, 56, 76, 78, 79, 80, 82, 84, 98, 99
1KC5	P	H	L	1, 2, 3, 4, 5, 6, 7
1KIQ	C	B	A	18, 19, 22, 23, 24, 25, 27, 102, 103, 116, 117, 118, 119, 120, 121, 124, 125
1MLC	E	B	A	41, 43, 45, 46, 47, 48, 49, 50, 51, 53, 66, 67, 68, 70, 79, 81, 84

1N64	P	H	L	26, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38
1NAK	P	H	L	313, 314, 315, 316, 319, 320, 321, 322
1OBE	P	H	L	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
1ORS	C	B	A	111, 112, 113, 114, 115, 116, 117, 119, 120, 123
1PZ5	C	B	A	1, 2, 3, 4, 5, 6, 7, 8
1QNZ	P	H	L	233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 248
1SM3	P	H	L	2, 3, 4, 5, 6, 7, 8, 9, 10
1TQB	A	B	C	127, 128, 158, 159, 185, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199
1V7M	V	H	L	57, 58, 61, 68, 71, 75, 98, 101, 102, 105, 106, 109, 110, 111, 112, 113, 114
1XGY	P	H	L	1, 2, 3, 4, 5, 6
1ZA3	R	H	L	25, 26, 27, 34, 36, 37, 38, 53, 54, 56, 57, 58, 59, 61, 62
2A6I	P	B	A	3, 4, 5, 6, 7, 8, 9, 10
2BDN	A	H	L	28, 30, 31, 32, 34, 37, 38, 39, 40, 41, 55, 56, 61, 64, 65, 68, 69
2DQJ	Y	H	L	14, 15, 16, 19, 20, 21, 62, 63, 73, 74, 75, 77, 93, 96, 97, 98, 100, 101, 102, 103
2FJH	V	H	L	16, 17, 18, 19, 21, 22, 23, 25, 61, 66, 101, 104
2H1P	P	H	L	602, 603, 604, 605, 606, 608, 609, 610, 611, 612
2HH0	P	H	L	2, 3, 4, 5, 6, 7, 8, 9, 10
2HRP	P	H	L	36, 37, 38, 39, 40, 41, 42, 43, 44, 45
2IFF	Y	H	L	41, 43, 45, 46, 47, 48, 49, 51, 53, 67, 68, 69, 70, 81, 84
2JEL	P	H	L	1, 2, 3, 4, 34, 36, 41, 64, 66, 67, 68, 70, 71, 72, 75, 76
2OR9	P	H	L	1, 2, 3, 4, 5, 6, 7, 8, 9, 10
2QHR	P	H	L	405, 406, 407, 408, 409, 410, 411, 412, 413, 414
2R29	A	H	L	306, 307, 308, 309, 310, 311, 312, 325, 362, 363, 364, 387, 388, 389, 390, 391
3AB0	A	B	C	102, 104, 113, 119, 120, 121, 122, 123, 125, 154, 156, 157, 158, 186, 188, 190, 192
3BDY	V	H	L	48, 81, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93
3CVH	C	H	L	4, 5, 6, 7, 8
3D85	C	B	A	82, 86, 87, 88, 91, 92, 93, 94, 95, 106, 110, 133, 134, 135, 136, 137, 139, 140
3E8U	P	H	L	4, 5, 6, 7, 8, 9, 10, 11, 13
3ETB	J	H	L	649, 651, 652, 653, 654, 655, 657, 680, 681, 682, 683, 684, 685, 686, 687, 716, 718
3F58	P	H	L	315, 316, 319, 320, 321, 322, 323, 324, 10, 11

3G6D	A	H	L	5, 6, 14, 18, 19, 21, 22, 23, 24, 97, 100, 101, 104, 105, 107, 108, 109
3GHB	P	H	L	306, 307, 308, 309, 312, 313, 314, 315
3GHE	P	H	L	304, 305, 306, 307, 308, 309, 312, 313, 314, 315, 316
3HR5	R	H	L	7, 8, 9, 10, 11, 12, 13, 14, 15
3KS0	B	H	L	30, 63, 64, 65, 66, 67, 68, 69, 70, 72, 74
3MLX	P	H	L	305, 306, 307, 308, 309, 312, 313, 314, 315, 316, 317, 318, 319
3NFP	I	H	L	1, 2, 3, 4, 5, 6, 25, 27, 43, 45, 116, 118, 120, 149, 152, 153, 154, 155
3P30	A	H	L	640, 643, 647, 650, 651, 653, 654, 657, 658, 661
3QG6	C	H	L	3, 4, 5, 6, 7, 8
3RKD	A	H	L	476, 477, 479, 484, 485, 496, 497, 498, 499, 508, 510, 512, 513, 514, 515, 534, 572, 573, 574, 575, 576, 577, 578, 592

Table 13: The nine designs whose values of $E_{\min}$ (in kcal/mol) were more negative than that of the native 5GZN antibody (shown below). Design gives the number Quick OptMAVEn assigned to each design; *HS* gives the humanization score [7] of each sequence.

Design	$E_{\min}$	Chain	Sequence	*HS*
31	-1244.8	H	QLQLQESGPGLVKPSETLSLTCTVSGGSISSSSYYWGWIRQPPGK GLEWIGSIYYSGSTYYNPSLKSRVTISVDTSKNQFSLKLSSVTAA DTAVYYCTWFAYWGRGTLVTVSS	19
		L	QAGLTQPPSVSKGLRQTATLTCTGNSNNVGNQGAAWPEQQGP PKLLSYRNNNRPSGISERLSASRSGNTASLTITGLQPEDEADYYC QSYDSSLSAVFGGGTQLTVL	38
28	-1031.5	H	EVQLVESGGGLVQPGGSLRLSCSASGFTFSSYAMWVRQAPGKGL EYVSAISSNGGSTYYADSVKGRFTISRDNSKNTLYVQMSSLRAE DTAVYYCVSYGYGGDRFSYWGQGTLVTVSS	46
		L	DIQMTQSPSSLSASVGDRVTITCRASQGISNSLAWYQQKPGKAP KLLLYAASRLESGVPSRFSGSGSGTDYTLTISSLQPEDFATYYCQS RELPPWTFGQGTKLEIK	29
11	-1017.2	H	EVQLVQSGAEVKKPGESLKISCKGSGYSFTSYWIGWVRQMPGK GLEWMGIIYPGDSDTRYSPSFQGQVTISADKPISTAYLQWSSLK ASDTAMYYCARGVDYYAMDYWGKGTTVTVSS	17
		L	EIVLTQSPGTLSLSPGERATLSCRASQSVSSSYLAWYQQKPGQAP RLLIYGASSRATGIPDRFSGSGSGTDFTLTISRLEPEDFAVYYCFQ GSVPTFGPGTKVDIK	25
19	-991.5	H	QVQLQQWGAGLLKPSETLSLTCAVYGGSFSGYYWSWIRQPPGK GLEWIGEIISGSTNYNPSLKSRVTISVDTSKNQFSLKLSSVTAAD TAVYYCARDRSYYFDYWGKGTTVTVSS	24
		L	EIVLTQSPATLSLSPGERATLSCGASQSVSSSYLAWYQQKPGLAP RLLIYDASSRATGIPDRFSGSGSGTDFTLTISRLEPEDFAVYYCQN DSYPLTFGQGTRLEIK	16

39	-973.6	H	QVQLVQSGAEVKKPGASVKVSCKVSGYTLTELSMWVRQAPGKG LEWMGGFDPEDGETIYAQKFQGRVTMTEDTSTDTAYMELSSL RSEDTAVYYCARYFDYWGKGTTVTVSS	25
		L	DIQMTQSPSSLSASVGDRVTITCRASQGIRNDLGWYQQKPGKAP KRLIYAASSLQSGVPSRFSGSGSGTEFTLTISSLQPEDFATYYCQN DSYPLTFGPGTKVDIK	16
0	-963.3	H	EVQLVESGGGVVRPGGSLRLSCAASGFTFDDYGMSWVRQAPGK GLEWVSGINWNGGSTGYADSVKGRFTISRDNAKNSLYLQMNSL RAEDTALYCTRSDGRNDMDSWGQGTTVTVSS	39
		L	EIVLTQSPATLSLSPGERATLSCRASQGVSSYLAWYQQKPGQAP RLLIYDASNRATGIPARFSGSGPGTDFTLTISSLEPEDFAVYYCQ QSKEVPLTFGPGTKVDIK	23
1	-941.6	H	EVQLVESGGGLVQPGRSLRLSCAASGFTFDDYAMWVRQAPGKG LEWVSGISWNSGSIGYADSVKGRFTISRDNAKNSLYLQMNSLRA EDTALYYCARDRSYYFDYWGQGTLVTVSS	18
		L	EIVLTQSPATLSLSPGERATLSCRASQSVSSYLAWYQQKPGQAP RLLIYDASNRATGIPARFSGSGSGTDFTLTISSLEPEDFAVYYCQQ NEYPWTFGPGTKVDIK	21
2	-898.2	H	QVQLQESGPGLVKPSDTLSLTCAVSGYSISRSSNWWGWIRQPPG KGLEWIGYIYYSGSTYYNPSLKSRVTMSVDTSKNQFSLKLSSVTA VDTGVYYCAKVKFYDPAPNDYWGKGTTVTVSS	67
		L	EIVMMQSPATLSVSPGERATLSCRASQSVSSNLAWYQQKPGQA PRLLIYGASTRATGIPARFSGSGSGTEFTLTISSLQSEDFAVYYCQ QYWSTWTFGPGTKVDIK	13
16	-873.9	H	EVQLVESRGVLVQPGGSLRLSCAASGFTVSSNEMSWVRQAPGK GLEWVSSISGSGGSTYYADSRKGRFTISRDNSKNTLLQMNSLRA EDTAVYYCARGDYYAMDYWGQGTLVTVSS	51
		L	DIQMTQSPSTLSASVGDRVTITCRASQSISSWLAWYQQKPGKAP KLLIYKASSLESGVPSRFSGSGSGTEFTLTISSLQPDDFATYYCQQ GQSYPFTFGPGTKVDIK	15
Native	-856.4	H	EVQLVESGGGVVQPGRSLRLSCAASGFTFSSYAMHWVRQAPGK GLEWVAVISYDGSNKYYADSVKGRFTISRDNSKSTLYLQMNNL RAEDTAVYYCARDHLGWSSIWSAPESFLDYWGQGTLVTVSS	52
		L	QSVLTQPPSVSAAPGQKVTISCSGSSSNIGNNYVSWYQQLPGTA PKLLIYDSNKRPSGIPDRFSGSKSGTSATLGITGLQTGDEADYYC GTWDSSLSVWVFGGGTKLTVL	4

Chapter 7 Bibliography

[1] D. M. Ecker, S. D. Jones and H. L. Levine, "The therapeutic monoclonal antibody market," *mAbs,* vol. 7, no. 1, pp. 9-14, 2014.

[2] A. Mahmuda, F. Bande, K. J. K. Al-Zihiry, N. Abdulhaleem, R. A. Majid, R. A. Hamat, W. O. Abdullah and Z. Unyah, "Monoclonal antibodies: A review of therapeutic applications and future prospects," *Tropical Journal of Pharmaceutical Research,* vol. 16, no. 3, pp. 713-722, 2017.

[3] M. H. Shepard, G. L. Phillips, C. D. Thanos and M. Feldmann, "Developments in therapy with monoclonal antibodies and related proteins," *Clinical Medicine,* vol. 17, no. 3, pp. 220-232, 2017.

[4] M. Leenaars and C. F. M. Hendriksen, "Critical Steps in the Production of Polyclonal and Monoclonal Antibodies: Evaluation and Recommendations," *ILAR Journal,* vol. 46, no. 3, pp. 269-279, 2005.

[5] P. J. Klasse, "Neutralization of Virus Infectivity by Antibodies: Old Problems in New Perspectives," *Advances in Biology,* vol. 2014, pp. Article ID 157895, 24 pages, 2014.

[6] R. J. Pantazes and C. D. Maranas, "OptCDR: a general computational method for the design of antibody complementarity determining regions for targeted epitope

binding," *Protein Engineering, Design & Selection,* vol. 23, no. 11, pp. 849-858, 2010.

[7] T. Li, R. J. Pantazes and C. D. Maranas, "OptMAVEn - A New Framework for the de novo Design of Antibody Variable Region Models Targeting Specific Antigen Epitopes," *PLOS One,* vol. 9, no. 8, p. e105954, 2014.

[8] R. J. Pantazes and C. D. Maranas, "MAPs: a database of modular antibody parts for predicting tertiary structures and designing affinity matured antibodies," *BMC Bioinformatics,* vol. 14, no. 168, p. n.p., 2013.

[9] S. Tonegawa, "Somatic generation of antibody diversity," *Nature,* vol. 302, no. 5909, pp. 575-581, 1983.

[10] A. F. U. H. Saeed, R. Wang, S. Ling and S. Wang, "Antibody Engineering for Pursuing a Healthier Future," *Frontiers in Microbiology,* vol. 8, p. 495, 2017.

[11] T. W. LeBien and T. F. Tedder, "B lymphocytes: how they develop and function," *Blood,* vol. 112, no. 5, pp. 1570-1580, 2008.

[12] M.-P. Lefranc, "IMGT Unique Numbering for the Variable (V), Constant (C), and Groove (G) Domains of IG, TR, MH, IgSF, and MhSF," *Cold Spring Harbor Protocols,* vol. 2011, no. 6, p. 633–642, 2011.

[13] J. Foote and H. N. Eisen, "Kinetic and affinity limits on antibodies produced during immune responses," *Proceedings of the National Academy of Sciences,* vol. 92, no. 5, pp. 1254-1256, 1995.

[14] G. D. Victoria and M. C. Nussenzweig, "Germinal Centers," *Annual Review of Immunology,* vol. 30, pp. 429-457, 2012.

[15] G. Teng and F. N. Papavasiliou, "Immunoglobulin Somatic Hypermutation," *Annual Reviews of Genetics,* vol. 41, pp. 107-120, 2007.

[16] Q. Wang, H. Yang, X. Liu, L. Dai, T. Ma, J. Qi, G. Wong, R. Peng, S. Liu, J. Li, S. Li, J. Song, J. Liu, J. He, H. Yuan, Y. Xiong, Y. Liao, J. Li, J. Yang, Z. Tong, B. Griffin, Y. Bi, M. Liang, X. Xu, C. Qin, G. Cheng, X. Zhang, P. Wang, X. Qiu, G. Kobinger, Y. Shi, J. Yan, and G. F. Gao, "Molecular determinants of human neutralizing antibodies isolated from a patient infected with Zika virus," *Science Translational Medicine,* vol. 8, no. 369, p. 369ra179, 2016.

[17] G. Sapparapu, E. Fernandez, N. Kose, B. Cao, J. Fox, R. Bombardi, H. Zhao, C. Nelson, A. Bryan, T. Barnes, E. Davidson, I. Mysorekar, D. Fremont, B. Doranz, M. Diamond and J. Crowe, "Neutralizing human antibodies prevent Zika virus replication and fetal disease in mice," *Nature,* vol. 540, no. 7633, pp. 443-447, 2016.

[18] A. Wec, A. Herbert, C. Murin, E. Nyakatura, D. Abelson, J. Fels, S. He, R. James, L. V. M. de, W. Zhu, R. Bakken, E. Goodwin, H. Turner, R. Jangra, L. Zeitlin, X. Qiu, J. Lai, L. Walker, A. Ward, J. Dye, K. Chandran and Z. Bornholdt, "Antibodies from a Human Survivor Define Sites of Vulnerability for Broad Protection against Ebolaviruses," *Cell,* vol. 169, no. 5, pp. 878-890, 2017.

[19] S. S. W. X. R. B. G. D. Kumar S, "Coupling of Aggregation and Immunogenicity in Biotherapeutics: T- and B-Cell Immune Epitopes May Contain Aggregation-Prone Regions," *Pharmaceutical Research,* vol. 28, no. 5, pp. 949-961, 2011.

[20] A. Murphy, L. Macdonald, S. Stevens, M. Karow, A. Dore, K. Pobursky, T. Huang, W. Poueymirou, L. Esau, M. Meola, W. Mikulka, P. Krueger, J. Fairhurst, D. Valenzuela, N. Papadopoulos and G. Yancopoulos, "Mice with megabase humanization of their immunoglobulin genes generate antibodies as efficiently as normal mice," *Proceedings of the National Academy of Sciences,* vol. 111, no. 14, pp. 5153-5158, 2014.

[21] N. Chennamsetty, V. Voynov, V. Kayser, B. Helk and B. L. Trout, "Design of therapeutic proteins with enhanced stability," *Proceedings of the National Academy of Sciences,* vol. 106, no. 29, pp. 11937-11942, 2009.

[22] G. D. Lapidoth, D. Baran, G. M. Pszolla, C. Norn, A. Alon, M. D. Tyka and S. J. Fleishman, "AbDesign: an algorithm for combinatorial backbone design guided by natural conformations and sequences," *Proteins,* vol. 83, no. 8, pp. 1385-1406, 2015.

[23] J. Adolf-Bryfogle, O. Kalyuzhniy, M. Kubitz, B. D. Weitzner, X. Hu, Y. Adachi, W. R. Schief and R. L. J. Dunbrack, "Rosetta Antibody Design (RAbD): A General Framework for Computational Antibody Design," *bioRxiv,* no. 183350, p. doi: https://doi.org/10.1101/183350 , 2017.

[24] B. S. Der, C. Kluwe, A. E. Miklos, R. Jacak, S. Lyskov, J. J. Gray, G. Georgiou, A. D. Ellington and B. Kuhlman, "Alternative Computational Protocols for Supercharging Protein Surfaces for Reversible Unfolding and Retention of Stability," *PLoS One,* vol. 8, no. 5, p. e64363, 2013.

[25] A. Miklos, C. Kluwe, B. Der, S. Pai, A. Sircar, R. Hughes, M. Berrondo, J. Xu, V. Codrea, P. Buckley, A. Calm, H. Welsh, C. Warner, M. Zacharko, J. Carney, J. Gray, G. Georgiou, B. Kuhlman and A. Ellington, "Structure-based design of supercharged, highly thermoresistant antibodies," *Chemical Biology*, vol. 19, no. 4, pp. 449-455, 2012.

[26] V. Pulito, V. A. Roberts, J. R. Adair, A. Rothermel, e. M. Collins, S. S. Varga, C. Martocello, M. Bodmer, L. K. Jolliffe and R. A. Zivin, "Humanization and molecular modeling of the anti-CD4 monoclonal antibody, OKT4A," *The Journal of Immunology*, vol. 156, no. 8, pp. 2840-2850, 1996.

[27] G. Lazar, J. Desjarlais, J. Jacinto, S. Karki and P. Hammond, "A molecular immunology approach to antibody humanization and functional optimization," *Molecular Immunology*, vol. 44, no. 8, pp. 1986-1998, 2007.

[28] D. Kuroda, H. Shirai, M. P. Jacobson and H. Nakamura, "Computer-aided antibody design," *Protein Engineering, Design & Selection*, vol. 25, no. 10, pp. 507-521, 2012.

[29] V. G. Poosarla, T. Li, B. C. Goh, K. Schulten, T. K. Wood and C. D. Maranas, "Computational De Novo Design of Antibodies Binding to a Peptide With High Affinity," *Biotechnology and Bioengineering*, vol. 114, no. 6, pp. 1331-1342, 2017.

[30] K. C. Entzminger, J.-m. Hyun, R. J. Pantazes, A. C. Patterson-Orazem, A. N. Qerqez, Z. P. Frye, R. A. Hughes, A. D. Ellington, R. L. Lieberman, C. D. Maranas and J. A. Maynard, "De novo design of antibody complementarity determining

regions binding a FLAG tetra-peptide," *Scientific Reports,* vol. 7, no. 1, p. 10295, 2017.

[31] D. Sirohi, Z. Chen, L. Sun, T. Klose, T. C. Pierson, M. G. Rossmann and R. J. Kuhn, "The 3.8 Å resolution cryo-EM structure of Zika virus," *Science,* vol. 352, no. 6284, pp. 467-470, 2016.

[32] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov and P. E. Bourne, "The Protein Data Bank," *Nucleic Acids Research,* vol. 28, no. 1, pp. 235-242, 2000.

[33] R. E. Soria-Guerra, R. Nieto-Gomez, D. O. Govea-Alonso and S. Rosales-Mendoza, "An overview of bioinformatics tools for epitope prediction: Implications on vaccine development," *Journal of Biomedical Informatics,* vol. 53, pp. 405-414, 2015.

[34] B. R. Brooks, C. L. Brooks, A. D. MacKerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma and Ovchi, "CHARMM: The Biomolecular Simulation Program," *Journal of Computational Chemistry,* vol. 30, no. 10, pp. 1545-1614, 2009.

[35] M. Blech, D. Peter, P. Fischer, M. M. T. Bauer, M. Hafner, M. Zeeb and H. Nar, "One Target—Two Different Binding Modes: Structural Insights into Gevokizumab and Canakinumab Interactions to Interleukin-1β," *Journal of Molecular Biology,* vol. 425, no. 1, pp. 94-111, 2013.

[36] Schrödinger, LLC, "The PyMOL Molecular Graphics System, Version 1.8".

[37] M. C. Saraf, G. L. Moore, N. M. Goodey, V. Y. Cao, S. J. Benkovic and C. D. Maranas, "IPRO: An Iterative Computational Protein Library Redesign and Optimization Procedure," *Biophysical Journal,* vol. 90, no. 11, pp. 4167-4180, 2006.

[38] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth and A. H. Teller, "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics,* vol. 21, no. 6, pp. 1087-1092, 1953.

[39] S. Tapryal, V. Gaur, K. J. Kaur and D. M. Salunke, "Structural Evaluation of a Mimicry-Recognizing Paratope: Plasticity in Antigen–Antibody Interactions Manifests in Molecular Mimicry," *The Journal of Immunology,* vol. 191, no. 1, pp. 456-463, 2013.

[40] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski and M. J. L. de Hoon, "Biopython: freely available Python tools for computational molecular biology and bioinformatics," *Bioinformatics,* vol. 25, no. 11, pp. 1422-1423, 2009.

[41] P. R. Evans, "Rotations and rotation matrices," *Acta Crystallographica Section D,* vol. 57, no. 10, pp. 1355-1359, 2001.

[42] W. Humphrey, A. Dalke and K. Schulten, "VMD: Visual Molecular Dynamics," *Journal of Molecular Graphics,* vol. 14, no. 1, pp. 33-38, 1996.

[43] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kale and K. Schulten, "Scalable molecular dynamics with NAMD," *Journal of Computational Chemistry,* vol. 26, no. 16, pp. 1781-1802, 2005.

[44] M. J. Grisewood, N. P. Gifford, R. J. Pantazes, Y. Li, P. C. Cirino, M. C. Janik and C. D. Maranas, "OptZyme: Computational Enzyme Redesign Using Transition State Analogues," *PLoS One,* vol. 8, no. 10, p. e75358, 2013.

[45] T. F. Havel, I. D. Kuntz and G. M. Crippen, "The Theory and Practice of Distance Geometry," *Bulletin of Mathematical Biology,* vol. 45, no. 5, pp. 665-720, 1983.

[46] M. Li, X. Chen, X. Li, B. Ma and P. M. B. Vitányi, "The Similarity Metric," *IEEE Transactions on Information Theory,* vol. 50, no. 12, pp. 3250-3264, 2004.

[47] S. Henikoff and J. G. Henikoff, "Amino acid substitution matrices from protein blocks," *Proceedings of the National Academy of Sciences,* vol. 89, no. 22, pp. 10915-10919, 1992.

[48] S. R. Eddy, "Where did the BLOSUM62 alignment score matrix come from?," *Nature Biotechnology,* vol. 22, no. 8, pp. 1035-1036, 2004.

[49] A. Stojmirović, "Quasi-metric spaces with measure," *Topology Proceedings,* vol. 28, no. 2, pp. 655-671, 2004.

[50] A. Muncherino, L. Liberti and C. Lavor, "MD-jeep: An Implementation of a Branch and Prune Algorithm for Distance Geometry Problems," in *Mathematical Software - ICMS 2010*, Berlin, 2010.

[51] C. Schwieters, J. Kuszewski and G. Clore, "Using Xplor-NIH for NMR molecular structure determination," *Progress in Nuclear Magnetic Resonance Spectroscopy,* vol. 48, pp. 47-62, 2006.

[52] J. W. Ponder, *TINKER: Software tools for molecular design,* St. Louis, MO: Washington University School of Medicine, 2004.

[53] J. J. Moré and Z. Wu, "Distance Geometry Optimization for Protein Structures," *Journal of Global Optimization,* vol. 15, no. 3, pp. 219-234, 1999.

[54] J. Lever, M. Krzywinski and N. Altman, "Principal Component Analysis," *Nature Methods,* vol. 14, no. 7, pp. 641-642, 2017.

[55] L. van der Maaten and G. Hinton, "Visualizing Data using t-SNE," *Journal of Machine Learning Research,* vol. 9, pp. 2579-2605, 2008.

[56] R. Marimont and M. Shapiro, "Nearest Neighbour Searches and the Curse of Dimensionality," *IMA Journal of Applied Mathematics,* vol. 24, no. 1, pp. 59-70, 1979.

[57] D. M. W. Powers, "Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation," *Journal of Machine Learning Technologies,* vol. 2, no. 1, pp. 37-63, 2011.

Academic Vita

Matthew F. Allan

EDUCATION

BS in Biochemistry and Molecular Biology, Minor in Statistics **August 2014 – May 2018**
The Pennsylvania State University University Park, PA
Schreyer Honors College

RESEARCH EXPERIENCE

Undergraduate Research Assistant **Academic Year, June 2015 – present**
Pennsylvania State University University Park, PA
Department of Chemical Engineering
Laboratory of Dr. Costas Maranas

- Explained structure-function relationships for an *E. coli* thioesterase enzyme
- Developing and benchmarking a new version of OptMAVEn antibody design software
- Designing and testing therapeutic antibodies for Zika using new version of OptMAVEn

DAAD RISE Research Scholar **May – August 2016**
Forschungszentrum Jülich Jülich, Germany
Institute of Complex Systems 6: Structural Biochemistry
Laboratory of Dr. Birgit Strodel

- Created a computational method to engineer product selectivity of enzymes
- Designed 38 variants of a cytochrome P450 enzyme to demonstrate efficacy of method

PROFESSIONAL EXPERIENCE

Quality Control Summer Intern **May – August 2017**
Regeneron Pharmaceuticals Rensselaer, NY
Quality Control Analytical Sciences

- Researched factors affecting the robustness of a microchip capillary electrophoresis assay
- Quantified relationships between factor levels and assay results
- Performed Design of Experiment and statistical analysis using JMP software

TEACHING EXPERIENCE

Learning Assistant for General Biochemistry I (BMB 401) **August – December 2017**
Pennsylvania State University University Park, PA

- Assisted ~170 students with learning structures/functions of biological macromolecules and using software including PyMOL
- Organized workshop outside of class to help students learn structures of amino acids

Learning Assistant for Introduction to Research (SC 297) **August – December 2016**
Pennsylvania State University University Park, PA

- Mentored 10 students to help them identify and apply to research opportunities

PAPERS CO-AUTHORED

Petrović, D., Ansgar Bokel, **Matthew Allan**, Vlada B. Urlacher, and Birgit Strodel (2018), "A Simulation Guided Design of Cytochrome P450 for Chemo- and Regioselective Macrocyclic Oxidation," *J. Chem. Inf. Model.*, Just Accepted Manuscript.

Grisewood, M., Néstor Hernández-Lozada, James Thoden, Nathanael Gifford, Daniel Mendez-Perez, Haley Schoenberger, **Matthew Allan**, Martha Floy, Rung-Yi Lai, Hazel Holden, Brian Pfleger, and Costas Maranas (2017), "Computational Redesign of Acyl-ACP Thioesterase with Improved Selectivity toward Medium-Chain-Length Fatty Acids," *ACS Catal.*, Vol. 7, Issue 6, 3837-3849.

PRESENTATIONS

"Robust Parameter Design of a Microchip Capillary Electrophoresis Assay," Regeneron Pharmaceuticals, Rensselaer NY, August 2017

"Engineering Product Selectivity into a Cytochrome P450 Enzyme," The Pennsylvania State University, University Park PA, October 2016

SKILLS

Programming Languages: Bash, C++, Python, R

Software: Burrows-Wheeler Aligner, CHARMM, Empower, Git, GROMACS, JMP, NAMD, PyMOL, SAMtools, VMD

Operating Systems: Linux (Ubuntu), MacOS X, Windows

Laboratory Skills: agarose gel electrophoresis, enzyme activity assays, microchip capillary electrophoresis, molecular cloning, PCR, SDS PAGE, small-scale protein purification

HONORS AND AWARDS

Student Marshal of Biochemistry and Molecular Biology, Penn State University	2018
Ronald Venezie Scholarship, Penn State University	2017
Evan Pugh Senior Scholar Award, Penn State University	2017
Morrow Family Endowment Prize, Penn State University	2016
DAAD RISE Scholarship, German Academic Exchange Service (DAAD)	2016
President Sparks Award, Penn State University	2016
President's Freshman Award, Penn State University	2015
Braddock Scholarship, Penn State University	2014
Academic Excellence Scholarship, Penn State University	2014
National AP Scholar	2014
National Merit Finalist	2014