

# A Combinatorial Approach to MHC Class I and II Peptide Vaccine Design with Application to SARS-CoV-2

Sara C. Schulte

A thesis presented for the degree of  
Master of Science



Supervisors:

Prof. Dr. Gunnar W. Klau

Prof. Dr. Alexander T. Dilthey

Algorithmic Bioinformatics

Heinrich Heine University Düsseldorf

October 4, 2022

## Acknowledgments

I would like to express my deepest gratitude to my supervisor Prof. Dr. Gunnar Klau for the countless, valuable discussions, for his inexhaustible patience, for all his advice and feedback, and for the support including the coffee breaks. I am extremely grateful to Prof. Dr. Alexander Diltthey for giving me the opportunity to carry out my research project, for taking the time to answer my many questions, for his continuous advice, and for his support.

Special thanks to Prof. Eric Rivals for his work and the correspondence on the hierarchical overlap graph, and for the valuable discussions at the DSB workshop. I like to thank Sven Schrinner for his help and practical suggestions on ILP formulations. Many thanks to Philipp Spohr for his constructive criticism and advice. Without you, HOGVAX would be without a name or a logo.

Last but not least, I would like to thank Martin Jürgens for his critical proofreading, suggestions, and support. Lastly, I would like to thank Marcel Käufler from the very bottom of my heart for his endless support over the last six months and especially in the last few days before the submission of this thesis. Thank you for your constant encouragement, your critical view and proofreading, your patience, and for bringing me tea and snacks.

# Abstract

Vaccination is the key component to overcoming the global COVID-19 pandemic. Peptide vaccines present a safe and cost-efficient alternative to traditional vaccines. They are rapidly produced and adapted for new viral variants. The vaccine's efficacy essentially depends on two components: the peptides included in the vaccine and the ability of major histocompatibility complex (MHC) molecules to bind and present these peptides to cells of the immune system. Due to the high diversity of MHC alleles and their diverging specificities in binding peptides, choosing a set of peptides that maximizes population coverage is a challenging task. Further, peptide vaccines are limited in their size allowing only for a small set of peptides to be included. Thus, they might fail to immunize a large part of the human population or protect against upcoming viral variants. Here, we present HOGVAX, a combinatorial optimization approach to select peptides that maximize population coverage. Furthermore, we exploit overlaps between peptide sequences to include a large number of peptides in a limited space and thereby also cover rare MHC alleles. We model this task as a theoretical problem, which we call the *Maximal Scoring k-Superstring Problem*. Additionally, HOGVAX is able to consider haplotype frequencies to take linkage disequilibrium between MHC loci into account. Our vaccine formulations contain significantly more peptides compared to vaccine sequences built from concatenated peptides. We predicted over 98 % population coverage for our vaccine candidates of MHC class I and II based on single-allele and haplotype frequencies. Moreover, we predicted high numbers of per-individual presented peptides leading to a robust immunity in the face of new virus variants.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Immune Response and Vaccination . . . . .	2
1.2	Combinatorial Vaccine Design . . . . .	4
1.3	Related Work . . . . .	5
1.4	Contribution . . . . .	7
<b>2</b>	<b>Preliminaries</b>	<b>8</b>
2.1	General Notations . . . . .	8
2.2	Shortest Common Superstrings and Overlap Graphs . . . . .	8
2.3	Tries and Trees . . . . .	10
2.4	Integer Linear Programming . . . . .	12
<b>3</b>	<b>Vaccine Design using Maximal Scoring k-Superstrings</b>	<b>13</b>
3.1	Data and Preprocessing . . . . .	13
3.2	Maximal Scoring k-Superstring Problem . . . . .	16
3.3	Solving the MSKS Problem using an Overlap Cost Graph . . . . .	21
3.4	Solving the MSKS Problem using a Hierarchical Overlap Graph . . . . .	22
3.5	Implementation . . . . .	33
3.6	Evaluation Metrics . . . . .	35
<b>4</b>	<b>Experimental Results</b>	<b>38</b>
4.1	Runtime Comparisons . . . . .	38
4.2	HOGVAX Significantly Increases Vaccine Robustness . . . . .	40
4.2.1	Population Coverage Optimization on Allele Data . . . . .	40
4.2.2	Population Coverage Optimization on Haplotype Data . . . . .	48
4.3	Designing Vaccines from the SARS-CoV-2 Spike Protein . . . . .	56
4.4	Combined Vaccine for MHC Class I and II . . . . .	56
4.5	Increased Population Coverage by Leveraging Overlaps . . . . .	58
4.6	Regional Context Embedding . . . . .	59
<b>5</b>	<b>Discussion and Conclusions</b>	<b>63</b>
<b>6</b>	<b>Code</b>	<b>67</b>
<b>A</b>	<b>Appendix ILP Formulations</b>	<b>74</b>
A.1	MSKS OCG Problem ILP Formulation . . . . .	74
A.2	Concatenation ILP Formulation . . . . .	75

<b>B</b>	<b>Appendix Results</b>	<b>76</b>
B.1	Minimum Number of Required Hits . . . . .	76
B.2	Comparison of HOGVAX and OptiVax-Robust on Genotype Data . . . . .	76
B.3	Designing Vaccines from the SARS-CoV-2 Spike Protein . . . . .	79
B.4	Combined Vaccine for MHC Class I and II . . . . .	82
B.5	Vaccine Efficacy of Concatenation Approach compared to HOGVAX . . . . .	82

# 1 Introduction

In our third year of the COVID-19 pandemic, there is no doubt about the necessity of SARS-CoV-2 vaccines. Despite that such vaccines already exist, further research is important in the face of viral mutations and the upcoming Covid wave.

Traditional vaccines include live attenuated or inactivated pathogens that are known for long-lasting immunity. Live attenuated vaccines include pathogens that have been modified such that they are no longer virulent but can still replicate in the host's cells. Inactivated vaccines contain killed or fragmented pathogens that do not replicate. Other common forms of vaccines are replicating or non-replicating viral vector vaccines. Here, harmless viruses, called vectors, are loaded with the genetic material of a pathogen which they transport to the host's cells. Either the vector envelope is spiked with proteins from the pathogen, or the vector encapsulates the genetic material. The traditional vaccines bear several risks like allergic reactions or becoming virulent again [51]. In contrast, modern approaches such as mRNA or peptide vaccines allow for a more targeted immunization with fewer side effects, and with application to non-infectious diseases like cancer or Alzheimer's disease [40, 6]. Such vaccines focus on those parts of a pathogen that trigger immune responses, whereas other parts might contribute only little to the immunizing effect. For example, peptides of the nucleocapsid (N) protein of SARS-CoV-2 might be interesting for peptide vaccine design [53], while the whole N protein can cause inflammatory lung injuries by interacting with a signaling pathway of the immune system and would therefore not be included in a vaccine [19]. mRNA vaccines contain encoded information of specific proteins that are produced in the cells after vaccination and trigger an immune response. The components of peptide vaccines are small, synthetically produced fragments from proteins, i.e., peptides, of a target pathogen that are known for being immunogenic. These could also be presented as a contiguous peptide sequence, which can additionally be translated into mRNA. The immunogenic material is usually enclosed by a delivery system such as liposomes that are vesicles with a lipid membrane. They transport the vaccine components to the cells of the immune system where they cause an immune response. In contrast to traditional vaccines, the advantage of modern vaccines is the fast, cost-efficient production and adaptability to new pathogen variants [51]. As they are synthetically produced, these vaccines have a high degree of purity such that they barely cause any allergic reactions [51]. On the downside, multiple vaccinations or vaccine adjuvants that enhance the immune response are often necessary to achieve strong, lasting immunity [40].

The *severe acute respiratory syndrome coronavirus type 2* is a single-stranded RNA virus of approximately 30 kilobases. It consists of four structural proteins: the 1273 amino acid (AA) long spike protein (S), the envelope protein (E) of length 75 AA, the membrane

protein (M) of 222 AA, and the 419 AA long nucleocapsid protein (N) [15]. Further, the virus RNA contains six open reading frames (ORFs): ORF1ab, ORF3ab, ORF6, ORF7ab, ORF8, and ORF10 [15]. Open reading frames are areas of the genetic material between a start and a stop codon and therefore likely encode proteins. However, for ORF10 it is not clear if the corresponding protein is expressed [15].

In this thesis, we focus on peptide vaccines and present a novel *in silico* method for vaccine design. We address the major challenge of selecting an optimal set of peptides to be included in the vaccine that maximizes population coverage, i.e., the fraction of individuals in a population that is immunized by the vaccine. Even though we limit ourselves to the coronavirus in this thesis, our model can be used for other purposes, such as cancer vaccines.

This thesis is structured as follows. We begin with biological background on peptide vaccine design and a short presentation of related work as well as our contribution to the field of *in silico* vaccine design. Our combinatorial approach is based on a theoretical problem. To fully understand this, we first explain some basics before presenting the problem. We continue with presenting experimental results and compare our method to a machine learning approach for peptide vaccine design presented by Liu et al. [39]. Afterward, the results are discussed and we conclude with an outlook on future work.

## 1.1 Immune Response and Vaccination

The immune system is distinguished into the innate and the adaptive immune system. Cells like macrophages, natural killer cells, or dendritic cells belong to the innate immune system and occur in all tissues. They are the first to respond to an infection and activate the specific immune response of the adaptive immune system, to which we count B and T cells [51]. Those carry highly diverse receptors that can recognize antigens. Antigens are molecular structures, usually peptides, to which antibodies can bind. Those antibodies are produced by B cells, also known as B lymphocytes. Naïve B and T cells are cells that were not yet exposed to antigens. They naturally occur in the human body and become active during an infection if their specific receptors recognize the antigens of the pathogen. The activation is followed by the proliferation of the specific cells that inactivate and destroy the pathogens [10]. Here, cytotoxic T cells play a crucial role in eliminating infected cells. After the infection is cleared, most of these cells die, but some cells remain and diverge into memory cells that circulate in the bloodstream. In case of a new infection with the same pathogen, the adaptive immune system can respond much faster by activating the memory cells that initiate the production of B and T cells with the fitting receptors. Memory cells are key components of long-lasting immunity. Peptide

and mRNA vaccines, for example, target T cell activation to induce immunity based on T memory cells.

The B cell receptor can bind antigens directly [40]. In contrast, the T cell receptor (TCR) only recognizes antigens that are presented on the cell surface by major histocompatibility complex (MHC) molecules [40]. The MHC, in humans also known as the human leukocyte antigen (HLA) complex or system, is a gene cluster on chromosome six and the most polymorphic region in human DNA. As humans are diploid, each MHC gene exists twice. The nucleotide sequence of a gene is termed allele. Due to the high diversity of the HLA system, the alleles of the MHC genes on the two chromosome copies naturally differ from each other. We refer to a single set of MHC alleles originating from the same chromosome as a haplotype, while genotype denotes the double set of alleles from both chromosomes.

When an antigenic peptide is presented by an MHC molecule, the two components form a peptide-MHC complex (pMHC). MHC proteins are distinguished into class I and class II proteins. Class I molecules are found on the outer cell surface of all nucleated cells and responsible for the presentation of peptides from the cell interior [55]. This is also known as *endogenous* antigen presentation. In contrast, MHC class II molecules present *exogenous* antigens on the outer cell membrane of antigen-presenting cells (APCs) [55]. These antigens are absorbed by the APC through a process called endocytosis. The cell membrane envelopes the extracellular particles and transports them into the cell interior in form of a vesicle. In this thesis, we consider the three classical MHC I loci HLA-A, HLA-B, and HLA-C, as well as the three classical MHC II loci HLA-DP, HLA-DQ, and HLA-DR. Thus, a haplotype of class I or class II consists of three alleles and correspondingly a genotype refers to six alleles.

The presentation pathways of MHC class I and II are usually different. However, cross-presentation makes it possible for exogenous antigens to be processed via the MHC class I pathway and presented by MHC class II molecules. mRNA and peptide vaccines encoded in RNA target the cross-presentation mechanism in APCs. We describe this process in the following based on [49].

The mRNA vaccine gets into the APC by endocytosis and is first translated into a protein by the ribosomes. A protein complex, called the proteasome, degrades proteins into smaller fragments, i.e., peptides. This is also known as the proteasomal cleavage process. The breakdown is not arbitrary, but also not completely understood, yet. However, there are most likely preferred cleavage sites where the proteasome cuts [60]. In general, protein sequences are cut at the so-called *C-terminus*, that is the end of a protein where the flanking amino acid has a free carboxylic acid, i.e., a chemical group identifying organic acids. The other end of the protein or peptide is termed *N-terminus*.



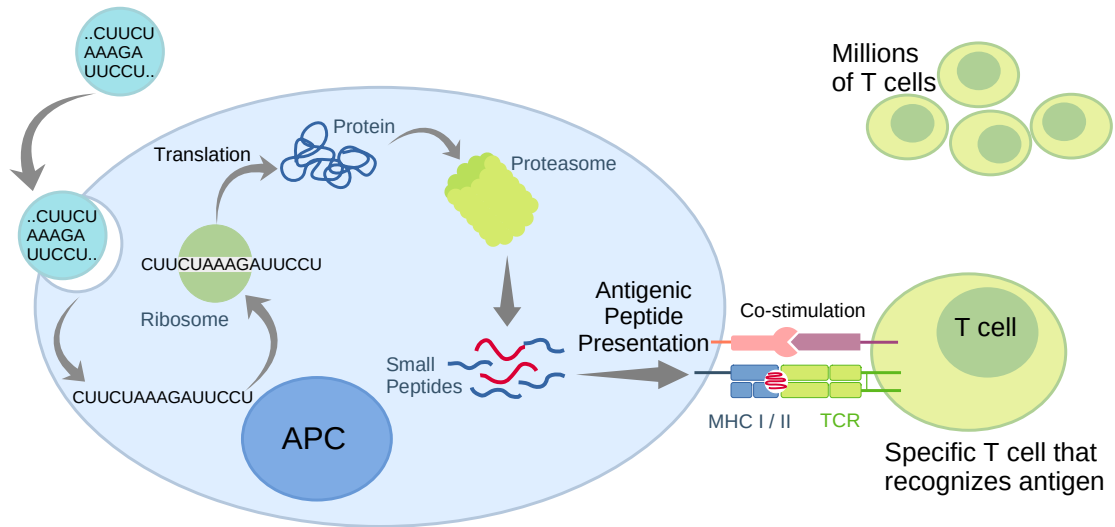
The decoded vaccine protein is also cut into smaller pieces which are loaded into the binding grooves of MHC class I and II proteins at the endoplasmic reticulum (ER). The ER is the transport system of the cell. From here, the loaded MHC molecules pass the Golgi apparatus where they are encapsulated into vesicles and sent to the membrane. Once on the outside of the cell, they present the bound peptides to the T cells. Figure 1 shows a simplified representation of this process.

MHC class I molecules present peptides to CD8<sup>+</sup> cytotoxic T cells, whereas peptide-MHC class II complexes are recognized by CD4<sup>+</sup> T cells, i.e., helper T cells [49]. The latter might further activate the so-called humoral immune response to which B cells and antibodies belong. The cytotoxic T cell-based immune response is also known as cell-mediated immunity. For a cytotoxic T cell to become active, the APC must not only present a foreign antigen but additionally a co-stimulatory molecule. In the absence of this co-stimulation signal, the T cell becomes unresponsive, which is termed *anergic*.

The translated protein could also be released into the extracellular space, from where it is then absorbed by an APC. This is the beginning of the general MHC class II presentation pathway which is roughly described as follows. After an APC absorbed extracellular particles, these are digested, for example, by lysosomes. Lysosomes are vesicles within the cell whose contents are acidic. They merge with the encapsulated particles and the acid breaks down the ingested proteins into their fragments. The fragments are then loaded into the binding groove of the MHC class II molecules and presented on the cell surface. This process is also called the endocytic pathway and target of, for example, viral vector, live attenuated, and inactivated vaccines.

## 1.2 Combinatorial Vaccine Design

The goal of combinatorial peptide vaccine design is to create a vaccine that maximizes population coverage by choosing an optimal set of peptides. This is a challenging task for several reasons. Not every peptide is recognized by T cells, for example, self-peptides. T cells that would recognize them are naturally eliminated to not harm the organism. Distinct peptides can trigger immune responses of different strengths, as also shown recently for SARS-CoV-2 peptides [20]. Moreover, the binding affinities between diverse peptides and MHC molecules can vary greatly [5]. If a peptide and an MHC allele bind, we speak of a peptide-HLA hit and say the allele is covered by the peptide. The greatest challenge, however, presents the diversity of MHC molecules. The distinct MHC allele frequencies differ greatly in different populations [5]. Additionally, MHC alleles of different loci may occur more or less often together than expected if the loci are considered to be independent. Such a phenomenon is called *linkage disequilibrium*. A vaccine must be effective across populations and also cover low-frequency MHC alleles to provide immunity for



**Figure 1:** Cross-presentation of an antigen-presenting cell. A vesicle carrying mRNA is transported into the cell via endocytosis. Ribosomes translate the mRNA into a protein which is cleaved by the proteasome. The resulting fragments are loaded into the binding grooves of MHC I and II molecules and presented on the outer cell surface. From millions of distinct T cells, those with the specific receptor that recognizes the presented antigen are activated and elicit an immune response.

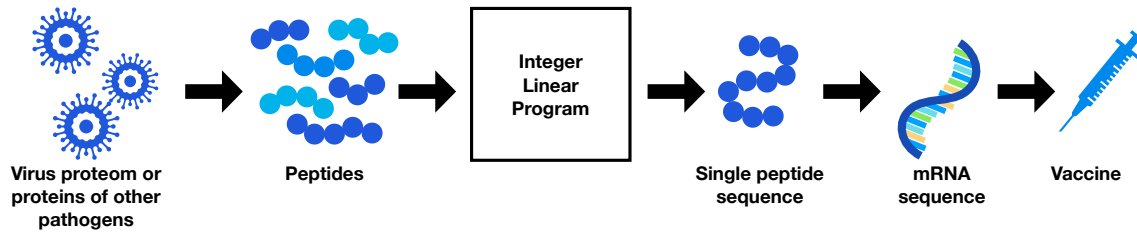
the whole population. It should also induce a robust immunity which requires a large set of peptides to be presented by the MHC molecules of an individual, i.e., a large number of peptide-HLA hits [57].

From these aspects, we infer three steps of combinatorial vaccine design. First, we split a protein into peptides and identify likely immunogenic peptides. For these, we use binding affinity prediction tools that compute the likelihood for a peptide and an MHC allele to form a complex, which is necessary to elicit a T cell response. Secondly, we must choose a set of peptides that maximizes population coverage, and thirdly, we use the chosen peptides to construct a vaccine sequence.

In this thesis, we use an integer linear programming approach to address the above challenges. We select a set of peptides that maximizes population coverage based on the HLA alleles covered relative to their frequency. These are then used to construct a so-called polypeptide sequence which can further be translated into mRNA. Figure 2 shows the basic overview of our combinatorial vaccine design.

### 1.3 Related Work

Bioinformatics has long been well established in vaccine development, and numerous publications exist using various strategies such as integer linear programming, machine learning, or genetic algorithms [54, 39, 57, 44]. Toussaint et al. presented an integer linear programming approach to select a diverse set of peptides that maximize immunogenicity [54]. Vider-Shalit et al. compute a peptide vaccine sequence that maximizes population

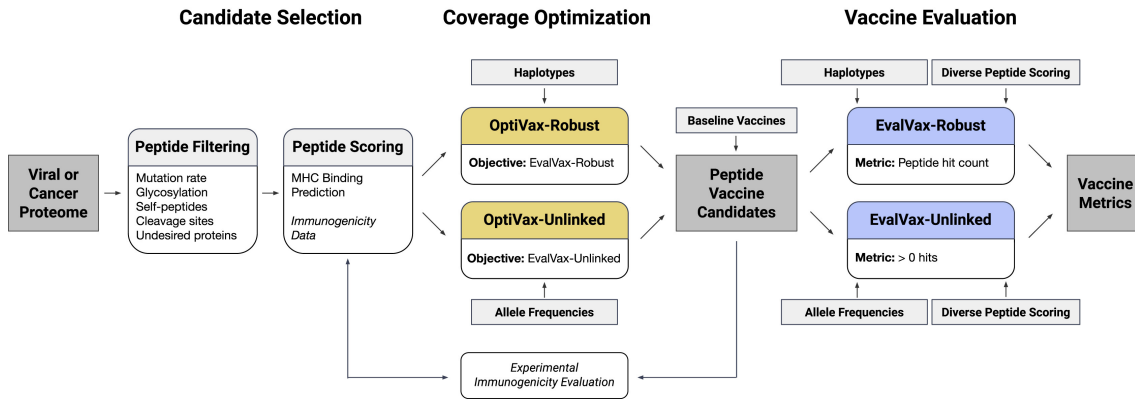


**Figure 2:** General overview of our combinatorial vaccine design approach. In the preprocessing phase, peptides are generated from pathogen proteins and immunogenic peptides are identified. We use an integer linear program to select an optimal vaccine set from which we construct a vaccine sequence. This can then be translated into mRNA.

coverage and induces a large number of peptide-HLA hits by further optimizing the peptide cleavage probability for each chosen peptide [57]. Schubert and Kohlbacher follow the approach to design peptide vaccines by concatenating peptides with spacer sequences in between [50]. This is called a string-of-beads vaccine, where the beads should ensure that the peptides are correctly cut by the proteasome [50].

In this thesis, we focus on the method proposed by Liu et al. in [39]. They developed a combinatorial machine learning method to choose a set of peptides that maximizes population coverage based on the HLA allele frequencies of the target population. It consists of two components: OptiVax which optimizes coverage, and EvalVax which evaluates a vaccine. OptiVax uses a beam search, which is a heuristic breadth-first search, to compute a set of peptides that optimizes the EvalVax function. The beam search starts with an empty set that is iteratively extended by one peptide. In each iteration, only the top  $b$ -many solutions are explored further, where  $b$  is the so-called beam size. After each iteration, the current candidate sets are evaluated by EvalVax. In the first iteration, all input peptides are evaluated. Only the peptides with the  $b$  highest EvalVax-scores are considered further and expanded with a further peptide. Again, all peptides from the input are used as extensions, with no duplicates in the vaccine candidates. This is because choosing the same peptide multiple times does not have an advantage regarding population coverage. The process of extending current solutions and keeping the best  $b$  peptide sets is repeated until a minimal set with the desired population coverage is computed or the pre-defined maximum of allowed peptides is reached. Furthermore, Liu et al. pay attention not to select peptides that have very similar amino acid sequences. They design vaccine candidates for MHC class I and class II separately with a beam size of  $b = 10$  for MHC class I, and  $b = 5$  for MHC class II [39]. The whole vaccine design approach is shown in Figure 3.

Liu et al. distinguished between OptiVax-Unlinked and OptiVax-Robust with corresponding evaluation functions, EvalVax-Unlinked and EvalVax-Robust. EvalVax-Unlinked considers single allele frequencies of each HLA locus independently and computes the



**Figure 3:** Vaccine design approach by Liu et al. [39]. Binding affinities are predicted with machine learning tools. OptiVax creates a set of peptides that is evaluated with EvalVax.

probability of at least one peptide presented by an MHC molecule per individual [39]. In contrast, EvalVax-Robust uses haplotype frequencies and thereby takes linkage disequilibrium into account. It computes the probability for at least  $n$  peptide-HLA hits per individual [39]. Thus, OptiVax-Robust optimizes the vaccine candidate to reach at least the number of  $n$  desired hits, while OptiVax-Unlinked maximizes the fraction of the population with at least one peptide-HLA hit. A detailed explanation of the two evaluation metrics follows in Section 3.6.

## 1.4 Contribution

Usually, peptide vaccines consist of a limited amount of peptides that are concatenated to sequences of limited length [57]. However, a small set of peptides will likely not cover rare MHC alleles nor induce immunity that is robust to virus mutations [57]. If a peptide of the vaccine is mutated in a later virus variant, e.g., due to a single nucleotide polymorphism, the immune system is not trained to recognize this new peptide. Thereby, a virus might escape the detection of the T cells [57]. To conquer these limitations, we exploit overlaps between peptide sequences and thereby significantly increase the number of peptides included in a vaccine of limited length. Without exceeding the space used by a regular peptide vaccine, we can increase the population coverage, cover rare MHC alleles, and elicit a strong immunity that is likely robust to viral mutations.

## 2 Preliminaries

In this chapter, we introduce the fundamental notations and background knowledge that is necessary to fully understand the methods presented in Section 3. Most definitions are given where they are used first, while the definitions in this chapter are intended to provide basic terms that will facilitate understanding later.

### 2.1 General Notations

We consider strings over a finite alphabet  $\Sigma$ . Here, this could either be the DNA or RNA alphabet  $\{A, C, G, T\}$ , or  $\{A, C, G, U\}$ , respectively, or the 20-letter amino acid alphabet  $\{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$ . We denote the empty string by  $\epsilon$  and the length of a string  $s$  is given by  $|s|$ . Let  $S$  be a set of strings. Then  $||S||$  is the total length of all strings in  $S$ .

The string  $s[i..j]$  is the substring of  $s$  which starts at position  $i$  and ends at position  $j$  with  $1 \leq i \leq j \leq |s|$ . If  $i = 1$  we call  $s[i..j]$  a *prefix* of  $s$  and if  $j = |s|$ , then  $s[i..j]$  is a *suffix* of  $s$ . A *proper* prefix, or suffix, of  $s$  is a prefix, or suffix, that is different from  $s$  by at least one character. We call a prefix of some string that is also a suffix of another string an *overlap* and define it as follows.

**Definition 1** (Overlap). Given two strings  $s$  and  $t$ . We say that the two strings share an overlap if  $s[i..|s|] = t[1..j]$ , where  $s[i..|s|]$  and  $t[1..j]$  are a proper suffix and prefix of  $s$  and  $t$ , respectively.

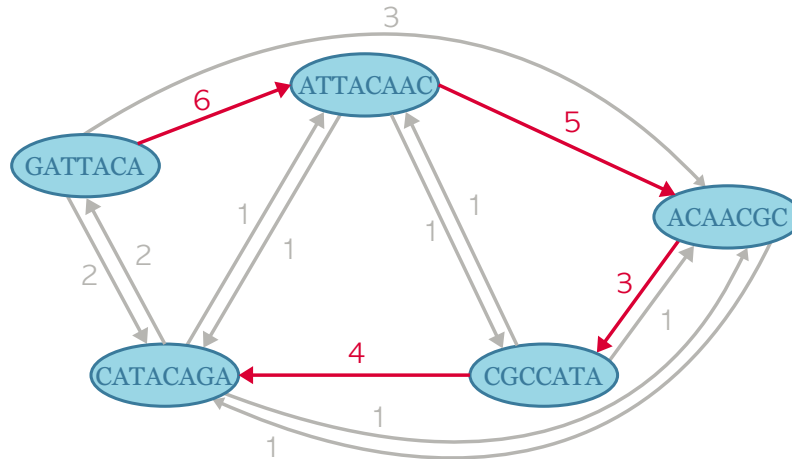
**Example 1.** The two strings  $s = \text{banana}$  and  $t = \text{ananas}$  share multiple overlaps, for example *ana* on the left. However, the longest overlap is *anana* as presented on the right.

$s$	b   a   n   a   n   a		b   a   n   a   n   a
	_____		_____
$t$	a   n   a   n   a   s		a   n   a   n   a   s
	_____		_____

Note, that overlaps are directional. As in the example, there are multiple overlaps from  $s$  to  $t$ , but there is none from  $t$  to  $s$ . We denote an overlap from a string  $s$  to a string  $t$  by  $ov(s, t)$  and the longest overlap from  $s$  to  $t$  by  $ov^*(s, t)$ .

### 2.2 Shortest Common Superstrings and Overlap Graphs

A well-known problem in bioinformatics is the *shortest common superstring* problem. For example, it is applied in genome assembly, where fragments of a genetic sequence, so-called reads, are rearranged into their genetically correct order. One approach to finding



**Figure 4:** Given is the overlap graph for six input strings with the maximum-weight Hamiltonian path colored in red. The edges are labeled with the length of the maximal overlap between the strings of the adjacent nodes. We omit edges with zero edge weight for a clearer figure. Traversing the maximum-weight Hamiltonian path results in the 19-character SCS *GATTACAACGCCATACAGA*. For this example, the SCS is a unique string and no other SCS exists.

such an arrangement is to compute the shortest common superstring (SCS) over all input strings. A superstring is defined as follows.

**Definition 2** (Superstring [25]). Given a set of strings  $S$ . A superstring of  $S$  is a string that contains each string of  $S$  as a substring.

The SCS problem deals with finding the superstring with minimal length of all superstrings for a given set of input strings. Such a string does not have to be unique, but multiple shortest superstrings can exist. Since the SCS problem was proven to be NP hard [18], many approximation algorithms [56, 3, 32] were published, and further research is still ongoing as a recent study shows [16].

Usually, the SCS problem is defined over a substring-free set of strings, which is a set  $S$ , where no string of  $S$  is a substring of some other string in  $S$ . If the set is not substring-free from the beginning, substrings can be united with their superstring to form a substring-free set without changing the SCS of  $S$  [52]. Thus, overlaps alone lead to superstrings that are shorter than the concatenation of all strings. With this assumption, we can approximate the SCS problem by finding a maximum-weight *Hamiltonian path* on a graph that encodes the overlaps of the input string set [52]. Such a graph is known as an *overlap graph* and the following definition is based on the definition by Tarhio and Ukkonen [52]. However, we deviate from the original formulation to a more general definition.

**Definition 3** (Overlap Graph [52]). The overlap graph of a set  $S$  of strings is a fully-connected, directed graph  $G = (V, E)$  with node set  $V = S$  and edges  $E = \{(u, v) \in V \times V \mid u \neq v\}$ . Each edge  $(u, v) \in E$  is labeled with the length of the maximal overlap  $ov^*(u, v)$ .

In order to compute the overlap graph, the *all-pair suffix-prefix* problem has to be solved. This is, computing all longest pairwise overlaps between the input strings, which are needed for labeling the edges of the overlap graph. Gusfield et al. presented an algorithm to solve this problem for a collection of strings  $S = \{s_1, \dots, s_n\}$  that runs in time  $\mathcal{O}(\|S\| + n^2)$ , which is optimal as  $\|S\|$  is the input size and  $n^2$  is the output size [26].

A Hamiltonian path visits each node of a graph exactly once. Like the SCS problem, the Hamiltonian path problem is NP hard [33]. Tarhio and Ukkonen have shown that such a path in an overlap graph corresponds to a superstring, and especially that a maximum-weight Hamiltonian path is equivalent to an SCS of the input strings. Figure 4 shows an overlap graph with a maximum-weight Hamiltonian path highlighted in red.

### 2.3 Tries and Trees

Another prominent data structure in bioinformatics constitutes the *keyword trie*. It is also known as *pattern* or *prefix trie* and often used for any flavors of pattern search, such as i) finding a pattern  $p$  in a string  $s$ , ii) counting the occurrences of  $p$  in  $s$ , or iii) querying the position of  $p$  in  $s$ . The latter use case is interesting for mapping reads to a reference genome, where the keyword trie would be built from the reference and the reads present the search patterns. We define a keyword trie as follows.

**Definition 4** (Keyword Trie [25]). Given a set  $S$  of strings over an alphabet  $\Sigma$ . A keyword trie  $K = (V, E)$  is defined by the following characteristics:

1. each edge  $e \in E$  is labeled by exactly one character,
2. any two outgoing edges at a node  $v \in V$  are labeled by distinct characters of  $\Sigma$ ,
3. each node  $v \in V$  corresponds to at least one string  $s_i \in S$  such that the concatenated characters on the path from the root of  $K$  to  $v$  spells out a prefix of  $s_i$ , and
4. each string  $s_i \in S$  maps to a leaf of  $K$  such that the path from the root to the leaf exactly spells out  $s_i$ .

While a keyword trie is a tree structure, the characteristics that define a trie as such are given by Definition 4.1 and 4.2. To save space, the keyword trie is often compressed such that nodes with a single outgoing edge are collapsed with their child node. The edges of the resulting tree are then labeled with the concatenation of the collapsed edge labels. If a tree does not contain nodes that have only a single child, we call this tree *compact*.

Example 2 demonstrates the construction and compression of a keyword trie.

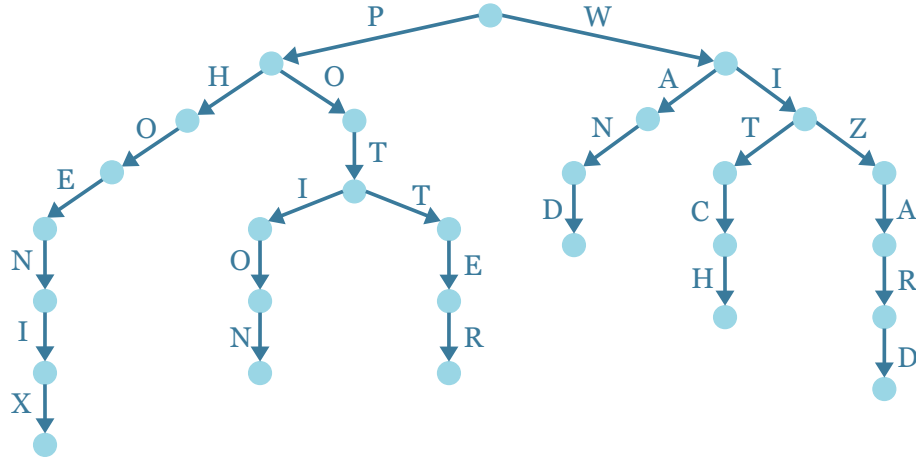


Figure 5: Keyword trie for the string set  $S = \{phoenix, potion, potter, wand, witch, wizard\}$ .

**Example 2.** Given the set of strings  $S = \{phoenix, potion, potter, wand, witch, wizard\}$ . The keyword trie of  $S$  is presented in Figure 5. We use Algorithm 1 by Compeau and Pevzner to construct this trie [11]. We start by adding a root node first and proceed with iteratively inserting the strings of  $S$ . This is shown in Algorithm 1 lines 1 and 2. The first string *phoenix* can be inserted directly. Next, we add *potion*, but as the trie is not empty anymore, we need to check if the root already has an outgoing edge labeled with *p*, see Algorithm 1 line 6. This is the case, such that we follow this edge and continue with the suffix *otion*. Again, we need to test for an edge with label *o*. Such an edge does not exist at the current node and we insert the remaining characters below.

A string  $s$  is added to the trie by identifying the longest existing prefix of  $s$  in the trie

---

**Algorithm 1** Constructing a Keyword Trie [11]

---

**Require:** Input set of strings  $S$

**Ensure:** Keyword trie of  $S$

```

1:  $Trie \leftarrow$  create empty graph and add root node
2: for  $s$  in  $S$  do
3:    $currentNode \leftarrow root$ 
4:   for  $i \leftarrow 1$  to  $|s|$  do
5:      $currentChar \leftarrow i$ -th character of  $s$ 
6:     if  $currentNode$  has outgoing edge with label  $currentChar$  then
7:        $currentNode \leftarrow$  end node of this edge
8:     else
9:       add new node  $newNode$  to  $Trie$ 
10:      connect  $currentNode$  and  $newNode$  with edge labeled with  $currentChar$ 
11:       $currentNode \leftarrow newNode$ 
12:     end if
13:   end for
14: end for

```

---



and, starting from the corresponding node of that longest prefix, adding the remaining characters of  $s$  below the identified node.

For example, when adding the string *potter* next, the prefix *pot* is already in the trie. Therefore, we add the remaining letters *ter* as a subtree below the node corresponding to *pot*. The remaining strings are added to the trie in the same fashion.

## 2.4 Integer Linear Programming

Later in this thesis, we formulate an integer linear program (ILP) to design a peptide vaccine. The following explanation of ILPs is based on the definition by Bökler et al. [4]. A linear program (LP) is a mathematical optimization problem defined by an objective function and a set of linear constraints that restrict the solution space to a polyhedron  $\mathcal{P} \subseteq \mathbb{R}^n$ . The objective function  $c^T x$  consists of a vector  $c \in \mathbb{R}^n$  and is maximized by finding an optimal point  $x \in \mathcal{P}$ . The constraints are composed of a matrix  $A \in \mathbb{R}^{m \times n}$  and a vector  $b \in \mathbb{R}^m$ , and often given in the form of  $Ax \leq b$ . From this, we define an LP as

$$\max\{c^T x \mid Ax \leq b, x \in \mathbb{R}^n\}. \quad (1)$$

Constraints not given in standard form can always be transformed to fit our definition. In some cases, specifying the constraints in non-standard form leads to better readability.

Linear programs can be solved in polynomial time. In contrast, integer linear programs are generally NP-hard [31] and differ from linear programs in that the decision variables must be integers, i.e.,  $x \in \mathbb{Z}^n$ . If only some of the variables must be integers, we speak of mixed-integer programs and if they are binary, we refer to 0-1 integer linear programs. The latter is proven to be NP-complete [33].

Besides the complexity of ILPs, modern solvers such as Gurobi are efficient in finding optimal solutions. They often use a branch-and-bound approach, where the solver finds an initial solution by removing the integrality restrictions on the decision variables [23]. Thus, it solves an LP called the *relaxation* of the ILP. If the initial solution only contains integer variables, the solving process is finished and an optimal solution is found. Otherwise, the initial solution presents an upper bound and the solver continues as follows:

First, a variable with non-integer solution is chosen, for example  $x_1 = 2.8$ . The solver initializes a search tree and adds a new constraint to each branch, for this example  $x_1 \leq 2$  and  $x_1 \geq 3$  [23]. Then it tries to solve the LP relaxation again. These steps are repeated until a feasible integer solution is found. For more details and further approaches, we refer to the website of Gurobi [23].

### 3 Vaccine Design using Maximal Scoring k-Superstrings

In this section, we present HOGVAX, our combinatorial approach for choosing an optimal set of peptides that maximizes population coverage. We first introduce our data followed by the description of the underlying *Maximal Scoring k-Superstring* (MSKS) problem. We show that the MSKS problem is NP-hard and give two approaches for solving it. Afterward, we present our integer linear program and three evaluation metrics to analyze the population coverage achieved by a vaccine candidate.

#### 3.1 Data and Preprocessing

Most of the data and preprocessing we are using in this thesis derive from the publication by Liu et al. [39]. Our vaccine design method requires three types of input data: peptide sequences, HLA allele frequencies, and binding affinity predictions between pairs of peptides and HLA alleles.

The peptides were created from the SARS-CoV-2 proteome. Liu et al. used the official reference entry Wuhan/IPBCAMS-WH-01/2019 from the GISAID database [15] that provides the nucleotide sequences of the open reading frames (ORFs) and the four structural proteins E, M, N, and S of SARS-CoV-2. To translate the nucleotide sequences into amino acid sequences, Liu et al. used the Nextstrain pipeline [27]. Then, they applied sliding windows of length 8 to 10 and of length 13 to 25 on the protein sequences to create target peptides for MHC class I and class II, respectively. Usually, MHC class I molecules enclose peptides of lengths 8 to 10 in their peptide binding groove [30]. The peptides are elongated along the binding groove and anchored at the groove's borders. However, cases exist where MHC class I molecules have bounded to longer peptide sequences [30]. Still, we stick to the conventional lengths as Liu et al. did. The binding groove of MHC class II molecules is open to the sides. Thus, they can bind to longer peptide sequences that extend the binding groove and have a minimum length of 13 amino acids [30]. Again, we use the same range of lengths as Liu et al. did [39]. With the sliding window approach, Liu et al. obtained 29 403 MHC class I peptides and 125 593 MHC class II peptides.

However, not all of these peptides are adequate candidates for a vaccine. Some peptides might not yield the desired immune response, if triggering a T cell activation at all. Therefore and for comparison of our method to OptiVax-Unlinked and OptiVax-Robust, we apply the same filtering on the peptides as described by Liu et al. This includes the elimination of self-peptides, *glycosylated* peptides, peptides of *cleavage* regions, and peptides that are likely to mutate. Self-peptides were identified by comparing the SARS-CoV-2 peptide sequences to the human proteome [39]. This step is necessary, as T cells usually do not recognize self-antigens. Moreover, vaccination with self-epitopes may cause an

autoimmune dysfunction such that the immune system reacts to the self-peptides which leads to autoimmune diseases [39].

For a T cell to respond to a pMHC, the presented peptide must have a free N-terminal amino group [40]. This means that no polysaccharide, i.e., *glycan*, binds to the amino group of the flanking amino acids of the peptide which would prevent T cells from binding to the pMHC. Therefore, we must filter out glycosylated peptides. The phenomenon of glycans binding to N-terminals is termed *N-linked glycosylation*. Liu et al. used the Net-NGlyc N-glycosylation prediction server [22] to identify such peptides with non-zero glycosylation probability and remove them from the input.

For the SARS-CoV-2 proteome, some positions are known where the protein sequences are cleaved. Due to the sliding window approach, our input set includes peptides that contain such cleavage positions. Cutting the proteins or peptides results in smaller fragments whose binding affinities to MHC molecules or immunological effects might be different from their structure of origin. Since the peptides are cleaved before they are loaded onto the MHC molecules, binding affinity predictions for peptides that contain cleavage regions are questionable [39]. Liu et al. filtered out such peptides based on data from UniProt and additional publications [12, 58, 13].

Lastly, mutation probabilities were computed for all peptides by tracking sequence changes along a set of different virus strains. The calculated probability is the rate of non-reference peptide sequences to the total number of peptide sequences examined [39]. For a more detailed explanation, we are referring to the original publication by Liu et al. [39]. Peptides with a low probability of mutation are likely to be conserved in future variants. Vaccines based on conserved peptides are therefore expected to be more robust to the evolution of new viral variants [39]. We use the computed mutation probabilities to filter out further peptides. The detailed filter criteria are given for each computation in Section 4.

The distribution of HLA alleles among the target population plays a key role when designing a vaccine. Based on the allele frequencies, we can optimize the vaccine to maximize population coverage. A more detailed explanation of this follows in Section 3.2. One approach in vaccine design is to consider the genes of an MHC class independently from each other. This is when we use single allele frequencies obtained from Liu et al. [39], who in turn collected the allele data from the dbMHC database [28] and accessed the frequencies with the IEDB population coverage tool [5]. The data contains frequencies for 2 395 alleles across 239 geographical regions for MHC class I, and frequencies for 275 alleles across 274 geographical regions for MHC class II.

However, we can also assume linkage between the loci of MHC class I and class II,

respectively, and therefore use haplotype or genotype data. In this case, we look at combinations of either three or six alleles. Liu et al. provided haplotype data for three populations self-reporting as having European, African or Asian ancestors [39]. For this, MHC class I and II alleles were determined with high-resolution typing for 2 327 individuals with European ancestry, 2 886 individuals with African ancestry, and 1 653 individuals with Asian ancestry [39]. The data is composed of 2 138 independent MHC I haplotypes with 230 distinct HLA-A, HLA-B, and HLA-C alleles, and 1 711 MHC II haplotypes with 280 different HLA-DP, HLA-DQ, and HLA-DR alleles [39]. We can generate genotype data by calculating the Cartesian product of the haplotypes for each HLA class independently.

All allele frequencies are normalized to sum up to 1 for each locus. In cases, where the raw frequencies do not sum up to one, Liu et al. insert a pseudo-allele to fill up the frequencies. Similarly, the haplotype frequencies for each population are normalized to sum to 1, using pseudo-haplotypes where necessary.

As mentioned before, the immunizing effect of a peptide vaccine that induces T cell activation depends upon MHC class I and II antigen presentation. Thus, it is crucial to know the probability of a specific peptide being presented by an MHC molecule.

In our case, the binding affinity predictions are calculated with NetMHCpan-4.1 and NetMHCIIpan-4.0 for MHC class I and II, respectively [46]. The NNAlign\_MA algorithm forms the core of both NetMHC applications and consists of a semi-supervised artificial neural network. The neural net was trained on mixed input types from i) *in vitro* binding affinity assays measuring the binding affinity (BA) of a single peptide to various MHC molecules, and ii) data from mass spectrometry known as eluted ligands<sup>1</sup> (EL) that include *in vivo* information about the steps in the antigen presentation pathway prior to the peptide-MHC binding, for example, peptide cleavage or transport [2].

Given a set of input peptides, NetMHC computes the binding affinity prediction for each peptide to bind to each MHC molecule known by the network. For a more detailed explanation of NNAlign\_MA, we refer to the paper by Alvarez et al. [2].

The binding affinity between peptide and HLA molecule is measured as  $IC_{50}$  value, this is *the half maximal inhibitory concentration*. It indicates how high the concentration of a substance must be *in vitro* for 50 % of a biological target to be inhibited by the substance. Here, the substance is a peptide, and the biological target is an MHC receptor. The binding affinity is given in nanomolar (nM) or as a score on a [0,1]-interval. The smaller the concentration in nM, or the higher the score, the stronger the binder. The

---

<sup>1</sup>Ligand: here, peptide that binds to MHC molecule; Elute: here, peptide is freed from MHC bond for measurement in mass spectrometry

binding affinities are bounded at a maximal score of 50 000 nM [39]. We can transform the BA concentration into a BA score by applying  $1 - \log_{50000}(\text{BA})$  [39].

Further, we use jupyter notebooks provided by Liu et al. to bring the binding affinity predictions into the shape of a table, where each cell contains the BA score for a peptide-HLA pair. We generate predictions for each combination of 233 HLA class I alleles and 29 402 peptides of length 8 to 10, and for each combination of 283 HLA class II alleles and 125 593 peptides of length 13 to 25.

Similar to Liu et al., we define a threshold of  $\leq 50\text{nM}$  for a peptide to be considered as a binder. This corresponds to a BA score of  $\geq 0.638$ . We are using a more conservative threshold, even though a threshold of 500nM would be sufficient to classify peptides into binders [59]. This is based on the statement by Liu et al. that the lower bound leads to higher confidence in those peptides predicted as binders [39]. For our computations, we need to binarize the predictions such that a 1 corresponds to a binder and a 0 to a non-binding peptide.

### 3.2 Maximal Scoring k-Superstring Problem

In order to computationally design a peptide vaccine, we first need to define the underlying combinatorial optimization problem. Here, we maximize population coverage by choosing peptides such that the fraction of covered alleles becomes maximal. As the distinct alleles occur with different frequencies in the target population, the summed frequencies of covered alleles correspond to the fraction of immunized individuals in the population. Hence, the two maximizations are equivalent. We say that an allele is covered if at least one peptide from the vaccine is predicted to have a reasonably high binding affinity to that allele. Furthermore, the selected peptides must not exceed a given immunological capacity in terms of total amino acid length.

We emphasize that any peptide can either be added completely or not at all to the vaccine but splitting the peptides might lead to unpredictable immunological characteristics. Therefore, the general problem of selecting an optimal set of peptides is very similar to the 0/1 *knapsack problem*. A definition is given below.

**Definition 5** (0/1 Knapsack Problem [48]). Given a set of  $n$  items  $A = \{a_1, \dots, a_n\}$  with a profit  $p_j$  and a weight  $w_j$  for each  $a_j \in A$ . Further, let  $C$  be the capacity of the knapsack. We are searching for a subset  $A' \subseteq A$ , such that the profit of the elements in  $A'$  is maximal while their summed sizes must not exceed the capacity  $C$ .

The items would be the peptides with their lengths as sizes. However, the peptides do not have a profit themselves but rather indirectly contribute to the overall profit, i.e., the population coverage, by binding to MHC molecules. The 0/1 knapsack problem is NP-

hard [33]. In the following, when speaking of the knapsack problem, we are referring to the 0/1 knapsack problem.

As described earlier, we aim to develop a contiguous vaccine sequence with a maximal length of  $k$ . It is no surprise that this limit is satisfied quickly when concatenating the amino acid sequences of the chosen peptides. We assume that the population coverage improves with more peptides included in the set since the probability increases that rare alleles are covered as well. So to push the given limit further, we want to exploit overlaps between peptides to save space. More precisely, we will calculate a superstring of the chosen peptides with a length of at most  $k$ . This in turn allows us to add even more peptides until the limit is satisfied. Now that we accounted for all relevant aspects, we can formally define the underlying problem which we call the *maximal scoring k-superstring* (MSKS) problem. We store the allele frequencies of  $m$  alleles in a vector  $f$ , where  $f_a$  is the frequency of allele  $a$ . The binding affinities of all peptide-allele pairs are transformed into a binary matrix  $B$  with  $B_{as} = 1$  if allele  $a$  and peptide  $s$  bind. The input set  $S$  of  $n$  peptide sequences, i.e., strings, derives from dissected protein sequences of some pathogen, e.g., the spike protein of SARS-CoV-2.

**Definition 6** (MSKS Problem). Given a set of strings  $S = \{s_1, s_2, \dots, s_n\}$  over a finite alphabet  $\Sigma$ , a  $m \times n$  binary matrix  $B$ , a frequency vector  $f = (f_1 \ f_2 \ \dots \ f_m)^T$  with  $m \in \mathbb{N}$  and  $f_a \in [0, 1]$  for  $a = 1, \dots, m$ , and a limit  $k \in \mathbb{N}$ . We are searching for  $P \subseteq S$  and a superstring  $\bar{s}$  of  $P$ , such that

1. the length of  $\bar{s}$  is at most  $k$ , and
2.  $\sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$  is maximal, where

$$h_a := \begin{cases} 1 & \exists s \in P : B_{as} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

We call  $\bar{s}$  with maximum length  $k$  a *maximal scoring k-superstring* of set  $S$ .

**Example 3.** Given the set  $S$  with  $s_1 = SEET$ ,  $s_2 = VSEE$ ,  $s_3 = EETG$ , and  $s_4 = VSEET$ , a limit of  $k = 5$ , the binary matrix  $B$  and the vector  $f$  with  $m = 5$  entries:

$$B = \begin{pmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad f = \begin{pmatrix} 0.11 \\ 0.36 \\ 0.04 \\ 0.41 \\ 0.08 \end{pmatrix}.$$

To naively identify an MSKS of  $S$ , we can, for example, compute a shortest common superstring for each subset of  $S$ , check if their length is within the limit of 5, and choose the subset with the highest score.

$P \subseteq S$	Superstring	Length	Score
$\{s_1, s_2\}, \{s_1, s_4\}, \{s_2, s_4\}, \{s_1, s_2, s_4\}$	VSEET	5	0.59
$\{s_1, s_3\}$	SEETG	5	0.96
$\{s_2, s_3\}, \{s_3, s_4\}, \{s_1, s_3, s_4\},$ $\{s_2, s_3, s_4\}, \{s_1, s_2, s_3, s_4\}$	VSEETG	6	1.00

Only the first two superstrings in the table do not exceed the limit. From those two, *SEETG* has the higher score that is computed as follows:

$$\text{score}(\{s_1, s_3\}) = \sum_{a=1}^5 h_a \cdot f_a = 1 \cdot 0.11 + 1 \cdot 0.36 + 0 \cdot 0.04 + 1 \cdot 0.41 + 1 \cdot 0.08 = 0.96$$

We multiply  $f_3 = 0.04$  by zero as there are only zero-entries in the binary matrix for the strings  $s_1$  and  $s_3$ . *SEETG* is the MSKS of this instance.

**Theorem 1.** The MSKS problem is NP-hard.

To show that the MSKS problem is NP-hard, we use the decision variant of this problem, i.e., *can we find a solution for the MSKS problem with a score of at least  $x$  that does not exceed the limit  $k$ ?* We define the decision form of the MSKS problem, the  *$x$ -scoring  $k$ -superstring* (XSKS) problem, as follows.

**Definition 7** (XSKS Problem). Given a set of strings  $S = \{s_1, s_2, \dots, s_n\}$  over a finite alphabet  $\Sigma$ , a  $m \times n$  binary matrix  $B$ , a frequency vector  $f = (f_1 \ f_2 \ \dots \ f_m)^T$  with  $m \in \mathbb{N}$  and  $f_i \in [0, 1]$  for  $i = 1, \dots, m$ , a limit  $k \in \mathbb{N}$  and a value  $x \in \mathbb{R}$ . We are searching for  $P \subseteq S$  and a superstring  $\bar{s}$  of  $P$ , such that

1. the length of  $\bar{s}$  is at most  $k$ , and
2.  $\sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$  is at least  $x$ , where

$$h_a := \begin{cases} 1 & \exists s \in P : B_{as} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

We will show that the XSKS problem is NP-complete and hence that the MSKS problem is NP-hard. For a given solution set  $P$  of the MSKS problem, no polynomial time algorithm exists that can verify if a given solution for the MSKS problem is optimal.

**Theorem 2.** The XSKS problem is NP-complete.

*Proof.* We begin by showing that the XSKS problem is in NP. Afterward, we prove that the XSKS is NP-hard.

1. Given a candidate solution for an XSKS instance  $I = (S, B, f, k, x)$  that consists of a set of strings  $P \subseteq S$  and a string  $\bar{s}$ , we need to verify in polynomial time whether the solution is a YES-instance or not. First, we examine if  $\bar{s}$  is a superstring of each string in  $P$  and if  $|\bar{s}|$  is at most  $k$ . This is possible in  $\mathcal{O}(|P| \cdot |\bar{s}|)$ . Next, we calculate the score of  $P$  with the objective function given in Definition 7.2 and check if it is at least  $x$ . This is done in  $\mathcal{O}(m)$ . Thereby, we can verify a given solution of the XSKS problem in polynomial time and it follows that  $\text{XSKS} \in \text{NP}$ .
2. To show the completeness of XSKS, we give a polynomial time transformation of the NP-complete decision version of the knapsack problem to the XSKS problem. More precisely, we show how to solve the knapsack decision problem with an instance of the XSKS problem.

Given a set of items  $A = \{a_1, \dots, a_n\}$  as well as a profit  $p_j$  and a weight  $w_j$  for each  $a_j \in A$ . The decision form of the knapsack problem asks for a subset  $A' \subseteq A$  such that the sum over the profits of the elements in  $A'$  is at least  $v$  while the summed weights of those elements do not exceed the capacity  $C$ . The problem was shown to be NP-complete by Richard M. Karp [33].

Given such a knapsack instance, we set the limit  $k$  of the XSKS instance equal to the capacity  $C$ , and the score  $x$  to the value  $v$ . We can construct a set  $S$  of  $|A|$  strings over an alphabet  $\Sigma$  of size  $|A|$ . Each  $a_j \in A$  corresponds to a unique string  $s_j \in S$  with  $|s_j| = w_j$ . Further, we do not allow any overlaps between the strings of  $S$ , and  $S$  must be substring-free. This can easily be achieved by using a unique character for constructing each string. For example, given two items  $a_1$  and  $a_2$  with weights  $w_1 = 2$  and  $w_2 = 4$ , we construct the corresponding strings  $s_1 = aa$  and  $s_2 = bbbb$ . The computation of the strings is done in  $\mathcal{O}(\sum_{a_j \in A} w_j)$ .

Further, we construct an  $n$ -dimensional frequency vector  $f$  for the XSKS instance such that  $f_j = p_j$  for each  $a_j \in A$ . Correspondingly, we define the binary matrix  $B$  for the XSKS instance to be the identity matrix of size  $n \times n$ . Both constructions take polynomial time.

We have shown how to construct an XSKS instance in polynomial time that corresponds to a given knapsack instance. We now show that an instance of the knapsack problem is a YES-instance if the corresponding XSKS instance is a YES-instance.



“ $\Rightarrow$ ” If we transform a given knapsack instance into an XSKS instance as described above, the following holds. If the XSKS instance  $I = (S, B, f, k, x)$  is a YES-instance that has a solution with a superstring of length at most  $k$ , then the knapsack problem has a solution with a total weight of at most  $C = k$ . Let  $P = \{s_1, \dots, s_m\}$  be a solution for the XSKS problem and let  $\bar{s} = s_1 s_2 \dots s_m$  be a superstring of  $P$  with  $|\bar{s}| \leq k$ . Now, if  $s_j \in P$  then  $a_j \in A'$  and  $\sum_{a_j \in A'} w_j = \sum_{s_j \in P} |s_j| = |\bar{s}|$ . By construction, the strings in  $S$  have no overlaps and thus  $|\bar{s}|$  is equal to the length of the concatenation of the strings in  $P$ . Therefore, the length of  $\bar{s}$  corresponds to the sum of weights of the items corresponding to the chosen strings. Neither the length of the superstring nor the sum over the item weights will exceed the limit  $k$ , i.e., the capacity  $C$ .

Now, if the solution  $P$  has a score  $x$ , then  $\sum_{a_j \in A'} p_j = v = x$ . The score of  $P$  is computed as follows. Whenever we add a string  $s_j$  to  $P$ , there is exactly one non-zero entry  $B_{jj}$  for that string along the diagonal of the binary matrix  $B$ . Thus, the corresponding variable  $h_j$  is 1 according to Definition 7.2 and the frequency  $f_j$  is added to the overall score. Remember, the frequency  $f_j$  corresponds to the profit  $p_j$ . Therefore, the score of  $P$  matches the sum over the profits of the items corresponding to the chosen strings. We have thus shown that from a YES-instance of the XSKS problem also follows a YES-instance of the knapsack problem.

“ $\Leftarrow$ ” Considering the opposite direction, where given a YES-instance for the knapsack problem the XSKS has a corresponding solution. Let  $A' \subseteq A$  be a solution for the knapsack problem with a total weight of at most  $C$  and profit of at least  $v$ . Then for each  $a_j \in A'$ , we select the corresponding string  $s_j \in S$  with length  $|s_j| = w_j$  from the XSKS problem and add it to  $P$ . By concatenating the strings in  $P$  we get a superstring of length at most  $\sum_{s_j \in P} |s_j| = k = C$ . We can calculate the score as described before, where each string  $s_j$  has a corresponding entry  $f_j$  in the frequency vector that is added to the total score. Then,  $\sum_{j \in \{1, \dots, n\}} h_j \cdot f_j = x = v$  where  $h_j = 1$  if  $s_j \in P$  and 0 otherwise. We have thus shown that a YES-instance of the XSKS problem with a score of at least  $x = v$  exists if the knapsack problem has a valid solution with a score of at least  $v$ .

□

Since the decision variant of the MSKS problem is NP-complete, the MSKS problem itself is NP-hard. In the following sections, we discuss two graph based approaches to solving the MSKS problem.

### 3.3 Solving the MSKS Problem using an Overlap Cost Graph

Instead of sampling all possible subsets and calculating the corresponding superstrings, we can solve this problem on a modified version of the overlap graph, which we call *overlap cost graph*. For this, we insert two additional nodes, a source, and a sink, and search for a path from source to sink that corresponds to an MSKS. The MSKS overlap cost graph (OCG) problem is defined as follows.

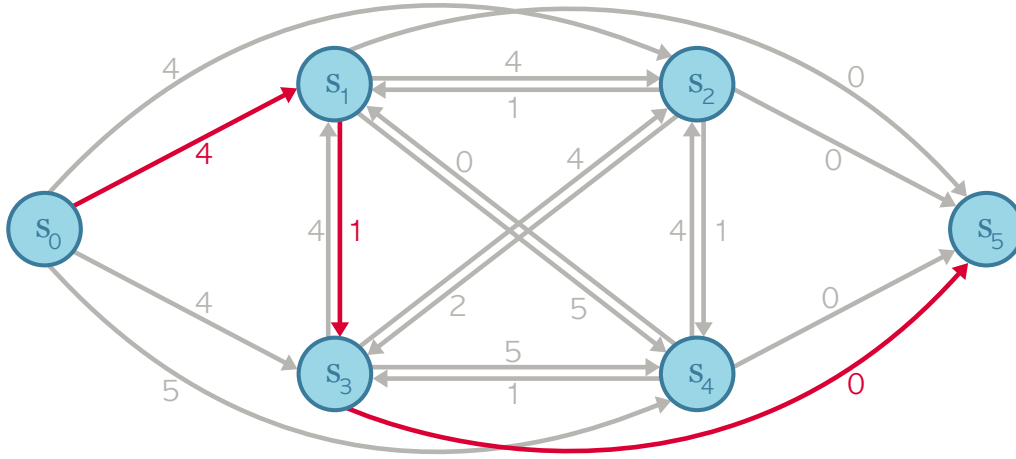
**Definition 8** (MSKS OCG Problem). Given an MSKS instance  $(S, B, f, k)$  with  $n$  strings. The overlap cost graph  $G = (V, E)$  is a directed graph with  $V = \{v_0, \dots, v_{n+1}\}$  where  $v_0$  and  $v_{n+1}$  are source and sink node, respectively, and each node  $v_i \in V$ ,  $1 \leq i \leq n$ , represents a string  $s_i \in S$ . The edge set is defined as  $E = \{(v_i, v_j) \mid v_i, v_j \in V, 0 \leq i \leq n, 1 \leq j \leq n+1, i \neq j\} \setminus (v_0, v_{n+1})$ . Thus, the source, or start node, only has outgoing edges while the sink, or end node, only has incoming edges. Further, we label the edges with the following costs: outgoing edges  $(v_0, v_i)$  from the source are labeled with  $l_{0i} = |s_i|$  and incoming edges  $(v_i, v_{n+1})$  to the sink are labeled with  $l_{in+1} = 0$ . All other edges  $(v_i, v_j)$  are labeled with  $l_{ij} = |s_j| - |ov^*(s_i, s_j)|$  where  $ov^*(s_i, s_j)$  is the longest overlap from  $s_i$  to  $s_j$ . We are searching for a path  $P$  in  $G$  from source  $v_0$  to sink  $v_{n+1}$  with

1.  $\sum_{(v_i, v_j) \in P} l_{ij} \leq k$ , and
2.  $\sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$  is maximal, where

$$h_a := \begin{cases} 1 & \exists v_i \in P : B_{as_i} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Each path from source to sink corresponds to a superstring of the traversed nodes and a potential vaccine sequence. Visited nodes are added to the vaccine and contribute to the overall score. Whenever we traverse an edge, we have to pay the labeled costs, i.e., add this amount of characters to the total length of our superstring. However, overlaps reduce these costs: the longer the overlap, the fewer the costs. Hence, the name of this graph structure. Figure 6 shows the overlap cost graph corresponding to Example 3. The highlighted path shows the MSKS *SEETG* with summed edge costs of 5.

We can solve the MSKS OCG problem using an ILP. The formulation is given in Appendix A.1. However, the OCG is a fully connected graph, except for the source and sink node. To construct it, we have to solve the *all-pair suffix-prefix* problem like for the overlap graph. Therefore, the complexity is in  $\mathcal{O}(|S| + n^2)$  which presents a computational challenge for our set of approximately 150,000 peptides. In contrast, the hierarchical overlap graph offers a significantly more efficient approach and is discussed in the next section.



**Figure 6:** Overlap Cost Graph (OCG) for the input  $s_1 = SEET$ ,  $s_2 = VSEE$ ,  $s_3 = EETG$ ,  $s_4 = VSEET$ , and a limit of  $k = 5$ . Edges are labeled with the number of extended characters. For example, when traversing the edge from source node  $s_0$  to  $s_1$ , we must add four characters to the total sum of characters. Now, when moving from  $s_1$  to  $s_3$ , we only have to add one further character as  $SEET$  and  $EETG$  share the overlap  $EET$ . So, we extend the sequence by the character  $G$ .

### 3.4 Solving the MSKS Problem using a Hierarchical Overlap Graph

The *hierarchical overlap graph* (HOG) was first proposed by Cazaux and Rivals and presents an alternative to the overlap graph [7]. Instead of a fully connected graph, the HOG is a tree-like structure that represents overlaps as inner nodes. It emerges from the Aho-Corasick Trie that was used by Park et al. to realize the linear time and space construction of the HOG [45]. Originally, Alfred V. Aho and Margaret J. Corasick presented the Aho-Corasick trie as a finite state pattern matching machine with a *goto*, *failure*, and *output* function [1]. To understand the HOG structure, we define the Aho-Corasick trie first and show the linear time algorithm to construct the HOG afterward. Thereafter, we present our model HOGVAX to computationally optimize peptide vaccines using a HOG.

**Definition 9** (Aho-Corasick Trie [25]). Given a set  $S$  of strings and the corresponding keyword trie  $K = (V_k, E_k)$  as defined in Definition 4. Let  $L(v)$  be the string corresponding to node  $v$ . We denote the length of the longest proper suffix of  $L(v)$  that is also a prefix of some other string  $s_i \in S$  as  $lp(v)$ . Let  $n_v$  be the unique node in  $K$  labeled with the suffix of  $L(v)$  of length  $lp(v)$ . If  $lp(v) = 0$  then  $n_v$  is the root node. In case  $v$  is the root then  $n_v$  is undefined. The Aho-Corasick trie  $A = (V_a, E_a)$  is a keyword trie with  $V_a = V_k$  and  $E_a = E_k \cup \{(v, n_v) \mid v \in V_a\}$ . We call an edge  $(v, n_v)$  a *failure* or *suffix link*, and the edges of  $E_k$  *tree edges*.

The Aho-Corasick trie is constructed in linear time by first creating the underlying keyword trie as described in Section 2.3, and then adding the suffix links by performing a breadth-first-search [1]. As defined, the root node does not have a suffix link. Nodes of

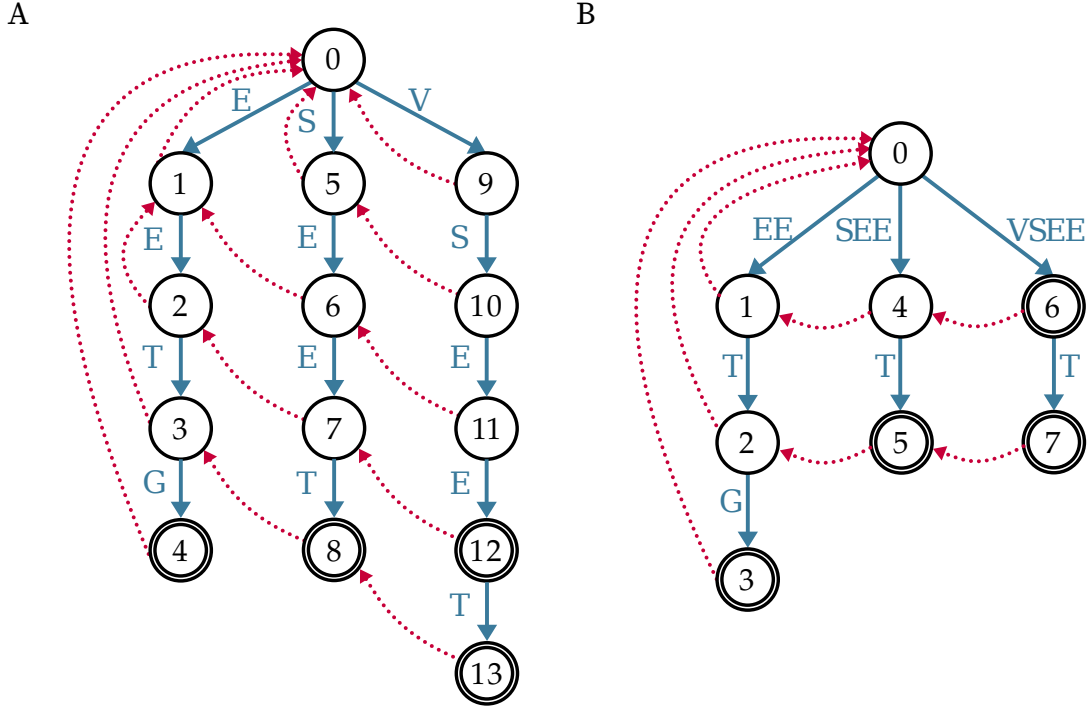
depth one correspond to single character strings. Therefore, their longest proper suffix is the empty string  $\epsilon$  that is represented by the root. Thus, their failure links point to the root node by default. For every other node  $v$  of depth  $d$ , Aho and Corasick described the construction of the failure links as follows. First, an edge  $(u, v) \in E_k$  with label  $c$  can be seen as *goto* function  $g(u, c) = v$ . Similarly, if a node  $u$  has no outgoing edge with label  $c$ , that is, we fail matching character  $c$  at node  $u$ , we define  $g(u, c) = \text{fail}$ . Secondly,  $f(u)$  is the node to which the failure link of  $u$  points, i.e., the *fail node* of  $u$ . Now, starting from each node  $u$  of depth  $d - 1$ , we proceed with one of the following actions:

1. If  $u$  has no outgoing edges, i.e.,  $g(u, c) = \text{fail}$  for all characters of the alphabet, do nothing;
2. Otherwise for each edge  $(u, v) \in E_k$  with label  $c$ , we define  $\text{next} = f(u)$ , and while  $g(\text{next}, c) = \text{fail}$ , we set  $\text{next} = f(\text{next})$ , until
  - (a) we reach a node, such that  $g(\text{next}, c) \neq \text{fail}$  and we define  $f(v) = g(\text{next}, c)$ ;  
or
  - (b) we reach the root  $r$  and  $g(r, c) = \text{fail}$  and we define  $f(v) = r$  [1].

In other words: For every node  $v$  of depth  $> 1$  with incoming edge character  $c$ , one must follow the suffix links starting from the parent node until one finds a tree edge with matching character  $c$ , or, if such an edge does not exist, the suffix link of  $v$  points to the root. The following example will clarify the construction process.

**Example 4.** Consider the set of strings  $S = \{EETG, SEET, VSEE, VSEET\}$ . To construct the Aho-Corasick trie, we first construct the keyword trie of  $S$  that is represented by the blue edges of the graph in Figure 7 A. We proceed with adding the failure links of nodes 1, 5, and 9, which all point towards the root node 0. Next, we add the suffix links for nodes 2, 6, and 10 of depths two. For node 2 and 6, starting from their parent nodes 1 and 5, respectively, we set  $\text{next} = f(1) = f(5) = 0$ . As  $g(0, E) = 1 \neq \text{fail}$ , the suffix links of nodes 2 and 6 direct to node 1. The failure link of node 10 points towards node 5 as  $g(0, S) = 5$ . We continue with the nodes of the next depth. For example, for node 3 we first set  $\text{next} = f(2) = 1$ , where  $g(1, T) = \text{fail}$ , and we must set  $\text{next} = f(1) = 0$ . At the root, we again fail to match the character, such that  $f(3) = 0$ . All other failure links are calculated in the same fashion. Figure 7 A shows the full Aho-Corasick trie for this example with the suffix links as dotted red arcs.

As defined for the keyword trie in 4, a node represents the string given by the concatenation of tree edge labels to that node. We emphasize that all nodes that are visited by trailing tree edges correspond to prefixes of their successive nodes. In contrast, by following the failure links from a certain node, all proper suffixes of that node are traversed.



**Figure 7:** The Aho-Corasick trie **A** and HOG **B** for the string set  $S = \{EETG, SEET, VSEE, VSEET\}$  with tree edges colored in blue and failure links in red. Double-encircled nodes correspond to whole strings of the input set. **B** Here, we did not use the labeling of Definition 10 to make the similarity between Aho-Corasick trie and HOG clearer. One can easily see that by compressing the edges of the Aho-Corasick trie, the HOG is created.

Now, we present the hierarchical overlap graph, which is a compressed version of the Aho-Corasick trie. A node of the HOG either represents a string from the input or the longest overlap between two strings. Additionally, just like the Aho-Corasick trie, the HOG has two types of edges. Cazaux and Rivals explicitly defined the HOG for a substring-free set of strings  $S$ , in other words, there is no string  $s_i$  that is a substring of any  $s_j$  for all  $s_i, s_j \in S$  [9]. However, we do not apply such restrictions to the input string set, and hence define a generalized version of the HOG based on the definition given in [45].

**Definition 10** (Generalized HOG). Given a set of strings  $S$ , we build a generalized *hierarchical overlap graph*  $H = (V, E)$  as follows.  $V = \{S \cup O(S) \cup \epsilon\}$  is the vertex set where  $O(S)$  is the set of longest overlaps from  $s_i$  to  $s_j$  for any two strings  $s_i, s_j \in S$ .

We define the set of directed edges as  $E = E_1 \cup E_2$ , where

- $E_1 = \{(x, y) \in V \times V \mid x \text{ is the longest proper prefix of } y\}$
- $E_2 = \{(x, y) \in V \times V \mid y \text{ is the longest proper suffix of } x\}$

Additionally, for an edge  $(x, y) \in E_1$  we use the labeling  $E_1 \rightarrow \mathbb{N}_0$ :  $(x, y) \mapsto |y| - |x|$ , otherwise the edge is a suffix link and we use labeling  $E_2 \rightarrow 0$ :  $(x, y) \mapsto 0$ .

When speaking of the HOG in this thesis, we are referring to the generalized version defined above. Figure 7 B shows the HOG for  $S = \{EETG, SEET, VSEE, VSEET\}$  that is the condensed version of the Aho-Corasick trie of Figure 7 A.

Assuming multiple strings have the same overlap, then the HOG needs a single node to represent it. For example, the overlap from the strings  $SEET$  and  $VSEET$  to  $EETG$  is  $EET$  and given by node 2 in Figure 7 B. In the overlap graph or OCG, however, the same overlap would be represented multiple times by the edges between the distinct node pairs. Further, the overlap graph and OCG only encode the length of the overlap, but not the overlap itself. Thus, one cannot easily see if an overlap occurs multiple times. Moreover, the HOG is built in linear time and space  $\mathcal{O}(|S|)$  [45]. In contrast, the overlap graph and OCG require  $\mathcal{O}(|S| + n^2)$  space for a set of  $n$  strings [45]. Overall, the HOG is a more informative and efficient data structure compared to the overlap or overlap cost graph. Therefore, it is the graph structure of choice and eponym of our vaccine design approach.

To construct the HOG from an Aho-Corasick trie of a string set  $S$ , an efficient way of deciding which nodes to keep is essential. The linear time algorithm by Park et al. employs the following three additional data structures for this purpose:

1. ***lps-Array***. The *lps*-array is a known data structure from the Knuth-Morris-Pratt algorithm [34]. It is computed for each string by the *lps*-function  $lps_s(i)$  that maps each index  $i$  of a string  $s$  to the length of the longest proper prefix of  $s$  that is also a proper suffix of  $s[1..i]$ . This can also be the empty string of length zero for which the corresponding *lps*-entry would be 0. A single character string, i.e.,  $s[1]$ , can only have the empty string as a prefix or suffix. An example is given below.

**Example 5.** Given the string  $s = CCSCC$ . The first entry of the *lps*-array is always 0. The prefix  $CC$  has a proper suffix that is also a proper prefix, namely  $C$  and  $lps_s(2) = 1$ . The full string also has a proper prefix that is a suffix, that is  $CC$ . Thus, the last entry in the *lps*-array is 2, the length of  $CC$ . Further examples are given below.

String	<i>lps</i> -Array
CCSCC	(0, 1, 0, 1, 2)
CCECT	(0, 1, 0, 1, 0)
ECTG	(0, 0, 0, 0)
SCC	(0, 0, 0)

2. ***Ancestor up(u)***. Park et al. introduced a new function  $up(u)$  for a node  $u$  in an Aho-Corasick trie that is similar to the *lps*-array [45]. Given a set of strings  $S$ , by definition does the corresponding Aho-Corasick trie contain a node for each prefix  $s[1..i]$ ,  $1 \leq i \leq |s|$ , of a string  $s \in S$ . Park et al. defined  $pnode_s(i)$  to be the node representing  $s[1..i]$ . Along the

path of tree edges from the root to the node representing  $s$ , we traverse all longest proper prefixes of  $s$  that are also suffixes of  $s[1..i]$ . Therefore, if we compute the  $lps$ -array of  $s$ , we can find each prefix of length  $lps_s(i)$  of string  $s[1..i]$  along that path for all  $i = 1 \dots |s|$ , where an  $lps$ -entry of 0 corresponds to the root node. The ancestor  $up(u)$  of a node  $u$  is defined as the node representing the longest prefix that is also a suffix of the string corresponding to  $u$  [45]. Formulated as an equation, this means:

$$up(pnode_s(i)) = pnode_s(lps_s(i)) \quad (3)$$

for each  $s \in S$  and  $1 \leq i \leq |s|$ , where  $pnode_s(0)$  is the root node [45].

**3. Index set  $R(u)$ .** Given the Aho-Corasick trie of a string set  $S$  with  $n$  strings, and a node  $u$ . The set  $R(u)$  is defined as

$$R(u) = \{i \in \{1, \dots, n\} \mid L(u) \text{ is a proper prefix of } s_i \in S\}. \quad (4)$$

Thus, it is the set of string indices whose leaf nodes occur in the subtree rooted at node  $u$  [45]. If  $u$  is a leaf itself, then  $R(u)$  is an empty set [45].

To construct  $O(S)$ , we need to identify the longest overlaps from each string  $s_i \in S$  to every other string  $s_j \in S$ . For this, we follow the suffix link chain  $(v_0, v_1, \dots, v_r)$  for each string  $s_i$ . This is, starting from the node  $v_0$  that represents  $s_i$ , we follow the suffix links until we reach the root node  $v_r$ . If we pass a node  $v_k$  for which there is an index  $j \in R(v_k)$ , then  $L(v_k)$  is an overlap from  $s_i$  to  $s_j$  [45]. Moreover, if  $L(v_k)$  is the longest overlap between  $s_i$  and  $s_j$ , we need to add  $v_k$  to the HOG. However, this is only the case for  $v_k$  with  $0 < k \leq r$  if there is no node  $v_m$  with  $0 \leq m < k$  and  $j \in R(v_m)$  [45]. Otherwise,  $v_m$  corresponds to a longer overlap between  $s_i$  and  $s_j$  than  $v_k$ . From this, Park et al. gave the following lemma.

**Lemma 1.**  $L(v_k)$  is the longest overlap from  $s_i$  to  $s_j$  if and only if  $j \in R(v_k) - R(v_{k-1}) - \dots - R(v_0)$  [45].

From this it follows that  $v_k \in O(S)$  if  $|R(v_k) - R(v_{k-1}) - \dots - R(v_0)| > 0$  [45]. To identify all nodes in  $O(S)$ , we must compute  $|R(v_k) - R(v_{k-1}) - \dots - R(v_0)|$  for all  $0 < k \leq r$  [45]. However, some of the index sets do not intersect  $R(v_k)$  as the nodes do not necessarily occur in the same subtree and therefore do not share the same leaf nodes. Thus, they are uninvolved in the computation of Lemma 1. In contrast, the ancestor of a node lays on the same subtree as the node itself. We can reformulate the condition of Lemma 1 so that

**Algorithm 2** Linear Time Construction of a HOG [45]**Require:** Input set of strings  $S$ **Ensure:** HOG of  $S$ 


---

```

1: Construct Aho-Corasick trie of  $S$ 
2: Compute  $lps$ -array for each  $s_i \in S$ 
3: Compute  $up(u)$  for each node  $u$ 
4: Compute  $|R(u)|$  for each node  $u$ 
5: Mark  $root$  as included in HOG
6: For each node  $u$ , initialize an empty set  $Child(u)$ 
7: for  $s_i$  in  $S$  do
8:    $u \leftarrow$  node representing  $s_i$  in Aho-Corasick trie
9:   Mark  $u$  as included in HOG
10:  while  $u \neq root$  do
11:     $r \leftarrow |R(u)|$ 
12:    for all  $v \in Child(u)$  do
13:       $r \leftarrow r - |R(v)|$ 
14:    end for
15:    if  $r > 0$  then
16:      Mark  $u$  as included in HOG
17:    end if
18:     $Child(u) \leftarrow$  empty set
19:    Add  $u$  to  $Child(up(u))$ 
20:     $u \leftarrow$  failure link of  $u$ 
21:  end while
22: end for
23: Call Algorithm 3 to construct HOG from marked nodes

```

---

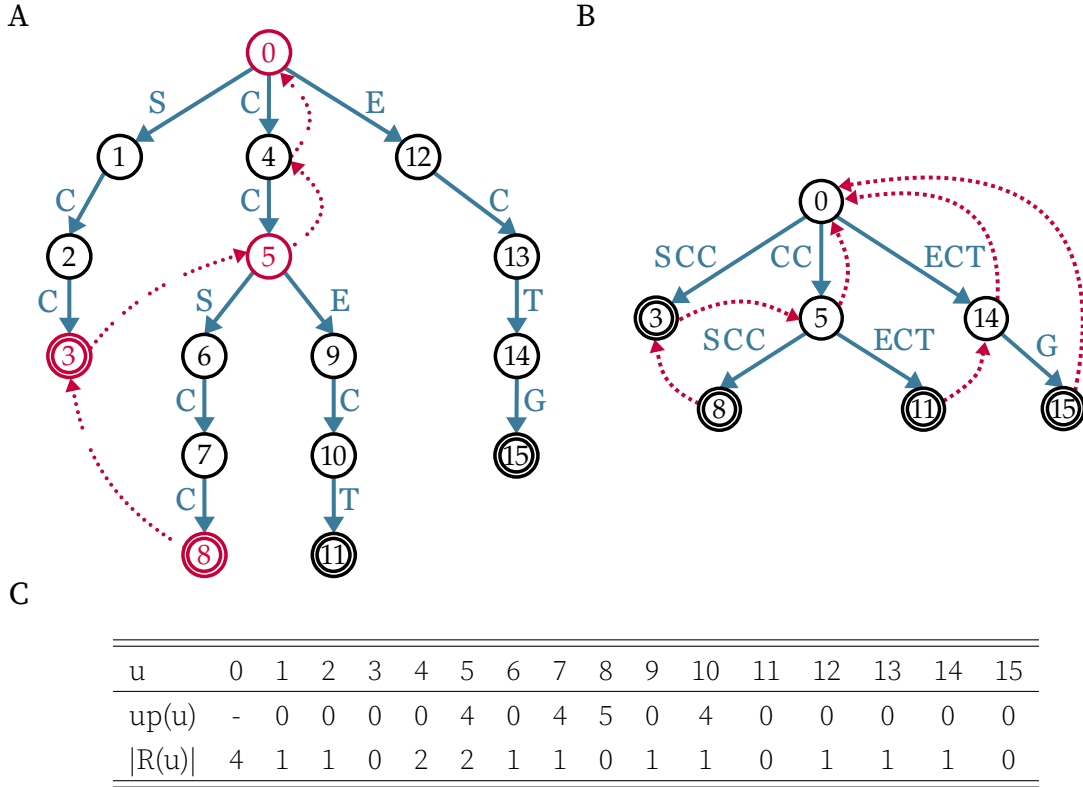
for every  $0 < k \leq r$ , the node  $v_k$  is in  $O(S)$  if

$$|R(v_k)| - \sum_{v_m} |R(v_m)| > 0 \quad (5)$$

with  $up(v_m) = v_k$  and  $0 \leq m < k$  [45].

Now we present the linear time algorithm to construct the HOG of a set  $S$  given in Algorithm 2. We start by constructing the Aho-Corasick trie of  $S$  and compute the  $lps$ -array for each input string. Further, we compute the ancestor using Equation 4 and the index set for each node. The root node and each node corresponding to a full string of the input set are always marked as included in the HOG. We start iterating over the nodes representing the strings in  $S$ , line 7 ff., and follow their suffix link chains until we reach the root node, line 10 ff. While bottom-up traversing the trie, we store each visited node  $u$  as a child of their ancestor  $up(u)$  using the set  $Child(up(u))$ , see Algorithm 2 line 19. We use the child-array to do the computation given in Equation 5, lines 10 to 17. An example of identifying the nodes of the Aho-Corasick trie that must be included in the HOG is given below.





**Figure 8:** Aho-Corasick trie **A** and HOG **B** for the string set  $S = \{SCC, CCSCC, CCECT, ECTG\}$ . Double-encircled nodes indicate input strings. We omit suffix links other than the highlighted ones in **A** to set the focus on this path. Nodes marked in red are also included in the HOG **B**. For the HOG, we again use string labels for an easier understanding. Table **C** shows the ancestor and size of the index set for each node of the Aho-Corasick trie.

**Example 6.** We consider the set of strings  $S = \{CCSCC, CCECT, ECTG, SCC\}$  for which the  $lps$ -array is computed in Example 5. The Aho-Corasick trie and the ancestor of each node  $u$  as well as  $|R(u)|$  are given in Figure 8 A and C. We examine the suffix link chain starting from node 8 that corresponds to the string  $CCSCC$ . Next, we visit node 3 for which by definition  $|R(3)| = 0$ . Thus it would not be marked in line 16 of the algorithm, but as it is a leaf node corresponding to the input string  $SCC$ , the node is marked in another iteration of the for-loop in line 7. We continue with node 5 which is the ancestor of node 8. Thus, in line 11 to 14, we compute  $|R(5)| - |R(8)| = 2 - 0 = 2$  which is larger than zero such that in line 16 we mark node 5 as included in the HOG. Next, node 4 is not marked because it is the ancestor of node 5 and thus  $|R(4)| - |R(5)| = 2 - 2 = 0 \not> 0$  which violates the condition of line 15 of the algorithm, i.e., Equation 5. The next node is the root node which is marked by default. We repeat the procedure for all remaining strings.

After identifying the nodes that must be included in the HOG, we need to compress the Aho-Corasick trie by removing the unmarked nodes and updating the edge set. We

**Algorithm 3** Compress Aho-Corasick trie**Require:** Aho-Corasick trie and marked nodes**Ensure:** HOG from marked nodes

---

```

1: Compute set of unmarked nodes
2: for  $u$  in unmarked nodes do
3:    $p \leftarrow$  predecessor of  $u$  along the trie edges
4:   for  $s$  in successors of  $u$  along the trie edges do
5:     Add edge from  $p$  to  $s$ 
6:   end for
7:    $Fail(u) \leftarrow$  set of nodes whose suffix links direct to  $u$ 
8:   for all  $v \in Fail(u)$  do
9:     Update suffix link from  $v$  to point to fail node of  $u$ 
10:  end for
11:  Remove  $u$ 
12: end for

```

---

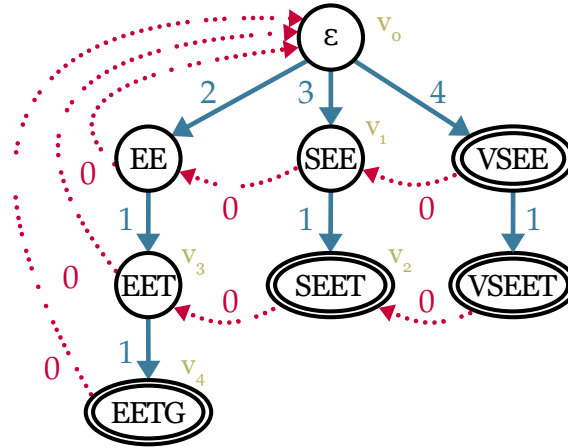
iterate over the unmarked nodes and connect the predecessor of an unmarked node with the successors of that unmarked node. Thereby, we guarantee that the resulting graph is still connected. Further, for all nodes whose suffix links point to an unmarked node, we need to update the fail node to the fail node of the unmarked node. For example, consider the unmarked node 4 of the Aho-Corasick trie in Figure 8 A. First, we connect its predecessor 0 with its successor node 5. Afterward, we set the suffix link of 5 to point to the root node as this is the fail node of node 4. The resulting HOG is presented in Figure 8 B. The compression of the Aho-Corasick trie is given by Algorithm 3.

The core of HOGVAX presents an ILP that solves the MSKS problem using a HOG. We adjust the problem formulation to search for a circuit through the HOG that starts and ends at the root node. In contrast to a cycle, a circuit might traverse nodes multiple times, while edges are passed at most once. Following the terminology in [8] of Cazaux and Rivals, we also call such a circuit a red-blue path, where red edges correspond to suffix links and blue edges refer to tree edges.

**Definition 11** (MSKS HOG Problem). Given an MSKS instance  $(S, B, f, k)$  and the corresponding HOG  $H = (V, E)$  for the string set  $S$ . Starting from the root node, we are searching for a circuit  $P$  in  $H$  such that

1. the sum of the edge weights of  $P$  is at most  $k$ , and
2.  $\sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$  is maximal, where

$$h_a := \begin{cases} 1 & \exists v \in P \text{ and } v \hat{=} s \in S : B_{as} = 1 \\ 0 & \text{otherwise.} \end{cases}$$



**Figure 9:** HOG according to Definition 10 for the string set  $S = \{EETG, SEET, VSEE, VSEET\}$ . The root node represents the empty string  $\epsilon$ . Following the vertex sequence  $(v_0, v_1, v_2, v_3, v_4, v_0)$ , we create the string  $SEETG$ , whose length is determined by the sum over the edge weights of the traversed circuit.

In the setting of vaccine design, we are also interested in the string described by the circuit  $P$ . To construct it, we follow  $P$  through the HOG. The edge weights tell us the number of characters by which the vaccine string is extended. Thus, we can easily see if our limit of  $k$  is exceeded. Here, we denote the label of an edge  $(u, v)$  by  $l_{uv}$  and the string corresponding to  $v$  by  $L(v)$ . Whenever we pass an edge  $(u, v)$ , we must add the suffix of  $L(v)$  with length  $l_{uv}$  to the output string. This is always possible as we either use a failure link and therefore do not extend the string, or we follow a tree edge  $(u, v)$  where  $L(v)$  is longer than  $L(u)$  by exactly  $l_{uv}$  characters.

**Example 7.** Consider the HOG given in Figure 9 that represents the string set of Example 3. The MSKS  $SEETG$  of Example 3 corresponds to the circuit described by the vertex sequence  $(v_0, v_1, v_2, v_3, v_4, v_0)$ . By traversing the edge  $(v_0, v_1)$ , we must add 3 characters to the string, as determined by the edge weight. Thus, we get the string  $SEE$  that is the label of  $v_1$ . Next, we follow the edge  $(v_1, v_2)$  and extend the string  $SEE$  by the suffix of  $L(v_2)$  of length  $l_{v_1 v_2} = 1$  which is  $T$  and gives us the string  $SEET$ . Then, we follow the failure link to node  $v_3$  without appending any characters. Lastly, we must add  $G$  to the string when we follow the tree edge to node  $v_4$ . We use the suffix link to get back to the root node and finished constructing the MSKS  $SEETG$ .

We reformulate the MSKS HOG problem as an integer linear program (ILP). Each edge  $(i, j)$  of the HOG  $H = (V, E)$  is assigned to a binary decision variable  $x_{ij}$ , where  $x_{ij} = 1$  if the edge is included in the circuit and  $x_{ij} = 0$  otherwise. We omit further variables for the vertices to reduce the number of possible variable assignments and the number of constraints. Thus, in the solution of our ILP, the chosen circuit is described by edges only. Our goal is to maximize population coverage by choosing peptides that cover the alleles

of a target population. Each allele  $a \in \{1, \dots, m\}$  is associated with a binary variable  $h_a$ . An allele is covered if at least one peptide from the vaccine set binds to the allele. By Definition 11, this is the case if the circuit contains at least one node of the HOG allocated to one of the input peptides for which the entry in the binding affinity matrix  $B$  is 1 and therefore  $h_a = 1$ . Otherwise the allele is uncovered and  $h_a = 0$ . A node is included in the circuit if an edge entering the node and an edge leaving the node are traversed by the circuit.

$$\text{maximize } \sum_{a \in \{1, \dots, m\}} h_a \cdot f_a \quad (1)$$

$$\text{subject to } \sum_{(0,j) \in E} x_{0j} \geq 1 \quad (2) \text{ outgoing edge from root}$$

$$\sum_{(i,j) \in E} l_{ij} \cdot x_{ij} \leq k \quad (3) \text{ sum of edge weights } \leq k$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} = \sum_{(j,g) \in \delta^+(j)} x_{jg} \quad \forall j \in V \quad (4) \text{ flow conservation}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} \leq \sum_{(h,g) \in \delta^-(W)} x_{hg} \quad \forall W \subset V, j \in W \quad (5) \text{ subtour elimination}$$

$$h_a \leq \sum_{\substack{j \in V: \\ s_j \in S}} \sum_{\substack{i: (i,j) \\ \in E}} x_{ij} \cdot B_{aj} \quad \forall a \in \{1, \dots, m\} \quad (6) \text{ hit if } a \text{ and } j \text{ bind}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E \quad (7)$$

$$h_a \in \{0, 1\} \quad \forall a \in \{1, \dots, m\} \quad (8)$$

Maximizing the population coverage is equal to maximizing the allele frequency of covered alleles, as described earlier in this thesis. We use the maximization term given by Definition 11 as the objective function of HOGVAX (1).

As we search for a circuit starting from the root node, we require at least one outgoing edge from the root 0 to start the circuit, i.e.,  $x_{0j} = 1$  (2).

Whenever we traverse an edge  $(i, j)$ , i.e.,  $x_{ij} = 1$ , we must add the edge label  $l_{ij}$  to the total length of the vaccine sequence constructed from the traversal. The sum of edge weights must not exceed the given limit of  $k$  (3). Further, to build a circuit, we must assure that for each visit of a node, we enter the node and leave it again. Thus, we apply a flow conservation constraint for each node ensuring that the number of incoming edges

equals the number of outgoing edges (4). We denote the set of incoming edges of a node  $j$  by  $\delta^-(j)$  and the set of edges leaving  $j$  by  $\delta^+(j)$ .

We intend to construct a single vaccine sequence described by a connected circuit. Thus, we use a subtour elimination constraint to avoid additional cycles that are not reachable from the root node. We reformulate the generalized subtour elimination constraint given in [4] as follows. For each subset of nodes  $W \subset V$ , we require that for each incoming edge to a node  $j \in W$  there must be an incoming edge to  $W$  (5). More precisely, if an edge  $(i, j)$  for  $j \in W$  is in the circuit, then there must be an edge  $(h, g) \in \delta^-(W)$  with  $x_{hg} = 1$  where we define  $\delta^-(W) = \{(h, g) \in E \mid g \in W, h \notin W\}$ .

Finally, we define the constraints on our  $h_a$  variables (6). Given a node  $j \in V$  that represents the peptide sequence  $s_j \in S$ . The node is part of the circuit, if an incoming edge  $(i, j)$  is in the circuit, thus if any  $x_{ij} = 1$ . An allele  $a$  is covered, i.e.,  $h_a = 1$ , if an  $x_{ij} = 1$  exists for which  $B_{aj} = 1$  as well. We must sum over all nodes representing peptides, and in a second sum, go over all incoming edges to address each  $x_{ij}$  which is multiplied with the corresponding matrix entry  $B_{aj}$ . We add such a constraint for each allele of our input frequencies.

So far, we only considered single allele frequencies and binding predictions between alleles and peptides. However, we can easily adjust HOGVAX to run on haplotype or genotype data. The ILP remains the same, but we change the meaning of  $h_a$  by simply changing the input data. Instead of referring to a single allele,  $a$  refers to a haplotype or genotype. Similarly, the binding affinity matrix must contain predictions for haplotype-peptide pairs or genotype-peptide pairs. We create the haplotype BA data from the allele BA data as follows. For each peptide, we pick the maximum BA score from the three alleles that compose the haplotype and use it as the BA score for the haplotype-peptide pair. We assume that a haplotype is covered if at least one peptide from the vaccine set binds to at least one of the three haplotype alleles. We proceed in the same way for the genotype data. Here, we must select the maximum out of six allele BA scores. Therefore, a genotype is covered if at least one peptide binds to one of the genotype alleles.

Furthermore, to design a vaccine for a set  $E$  of multiple populations whose allele frequencies differ heavily, we can adjust the objective function as follows:

$$\text{maximize} \quad \sum_{e \in E} \sum_{a \in \{1, \dots, m\}} h_a \cdot f_a^e$$

with  $f_a^e$  referring to the allele frequency of allele  $a$  in population  $e$ . Note, that the allele set must be the same for different populations. For alleles that do not occur in a target population, we can simply set the corresponding frequency entry in the vector to zero.

Instead of requiring that only at least one peptide binds to an allele, we can specify that at least  $c$  many peptides must bind to an allele to be covered by a vaccine. This is, we require  $c$  many peptide-HLA hits and change the constraint (6) of the ILP to

$$h_a \cdot c \leq \sum_{\substack{j \in V: \\ s_j \in S}} \sum_{\substack{i:(i,j) \\ \in E}} x_{ij} \cdot B_{aj} \quad \forall a \in \{1, \dots, m\}.$$

We can apply this constraint for haplotype and genotype data as well.

### 3.5 Implementation

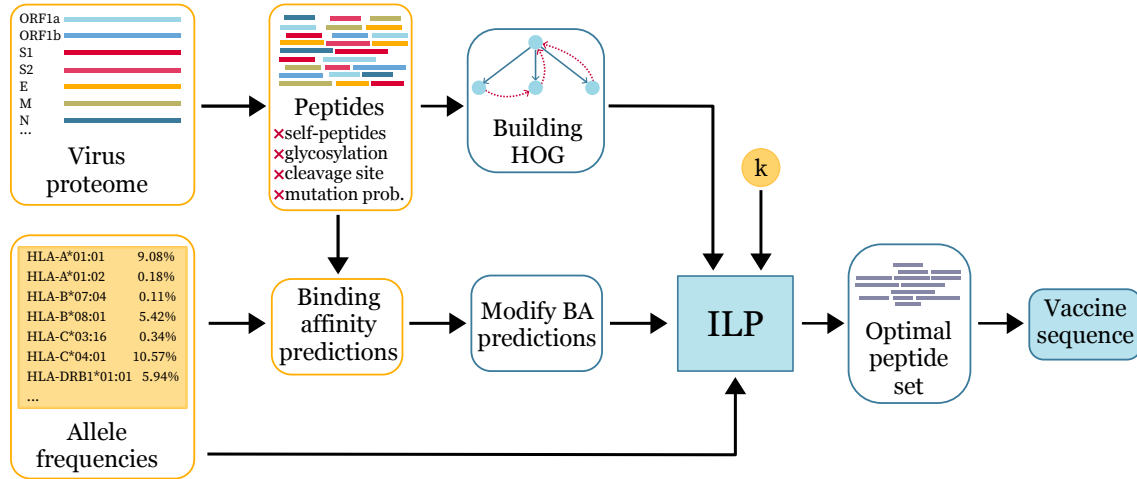
In this section, we present some details on the implementation of HOGVAX. We implemented a pipeline using the bioinformatics workflow management system Snakemake for an easy and reproducible execution of HOGVAX [42]. Snakemake is based on the Python language and we use it with the software management system Conda such that any software requirements are automatically deployed in the workflow. The Snakemake pipeline consists of rules that define how output files are generated from specified input files. When executing the pipeline, Snakemake generates a directed acyclic graph that determines the execution order of the rules. Our workflow is given in Figure 10.

As shown, the allele frequencies and the construction of the peptides as well as the peptide filtering are not yet part of the pipeline as we use the data provided by Liu et al. [39]. The binding affinity predictions must be calculated in advance and formatted by a script also given by Liu et al.

The limit  $k$  is also an input to our pipeline. Since we are comparing HOGVAX to OptiVax by Liu et al., we use the selected peptides of OptiVax, concatenate them, and use the length of the resulting string as our input  $k$ .

The pipeline starts with generating the HOG of the given input peptides. This also includes the construction of the Aho-Corasick trie. We implemented the linear time algorithm given in Algorithm 2 and used NetworkX as the underlying framework. We saved the ancestor, the size of the index set, and the child-array as attributes of each node. As our input set of peptides is not substring-free due to the sliding windows approach, not only leaf nodes but some inner nodes, too, correspond to full input strings. Thus, we had to keep track of these nodes and store them in a separate file.

The pipeline is supposed to create a superstring of the chosen peptides by using the HOG. If the string set for constructing the HOG is substring-free then the shortest red-blue path corresponds to an SCS [8]. However, to realize this behavior for our set of peptides, we must apply another preprocessing step to the data which is part of our pipeline. We cannot simply remove all substrings from the set as they might fit better into the vaccine sequence compared to their superstrings. Further, the impact of a peptide depends upon



**Figure 10:** HOGVAX workflow. The yellow parts refer to input data of the pipeline that must be generated beforehand. The blue parts correspond to the pipeline steps.

its predicted binding affinities. However, we must avoid selecting both the substring and the corresponding superstring, because the graph has no information about substrings. In such a case, the two strings would be concatenated instead of selecting only the superstring. As this work is based on the assumption that the proteasome cuts out the peptides we assembled to create the vaccine sequence, we expect the substring is cut from the superstring when the larger string is selected. Therefore, we merge the scores of an allele and all peptide sequences in the binding affinity prediction matrix such that a superstring is given the maximum score from itself and its substrings. An optimal circuit does not select both peptides as the substring has no benefits regarding immunization over the superstring, and thus would only add to the total length.

**Example 8.** Here, we consider three peptides  $p_1$ ,  $p_2$  and  $p_3$ , where  $p_1$  is a substring of  $p_2$  which is a substring of  $p_3$ . Given are three alleles and fictional binding affinity scores. The table on the left shows the original values with the maximal values highlighted in red. On the right is the modified table, where each superstring receives the maximum score from itself or its substrings. For example, in the last row,  $p_3$  has the maximum score, so we change nothing for  $p_3$ . However, the score of  $p_1$  is larger than that of  $p_2$ , so we change the score of  $p_2$ .

	$p_1$	$\subseteq$	$p_2$	$\subseteq$	$p_3$		$p_1$	$\subseteq$	$p_2$	$\subseteq$	$p_3$
HLA-A*01:01	0.5		0.1		0.2	$\Rightarrow$	0.5		0.5		0.5
HLA-B*07:02	0.1		0.4		0.3		0.1		0.4		0.4
HLA-C*01:02	0.3		0.2		0.5		0.3		0.3		0.5

The next step in our pipeline is solving the ILP. We decided to use the Gurobi Optimizer [24] that provides a Python API. Thus, we implemented the ILP as described in Section 3.4 in Python. However, instead of implementing the subtour elimination constraints for each subset of nodes directly, we use the Gurobi callback class. This allows for adding constraints during the optimization process. For example, if Gurobi found a solution, we use the callback function to check for subtours. Those are strongly connected components that are not connected to the root. We identify them using NetworkX. Only if any subtours exist, do we add subtour elimination constraints for the specific subtours. Thereby, we can reduce the practical complexity of the ILP.

When we compute a vaccine candidate using single-allele frequencies, we use all three loci of MHC class I or class II simultaneously. As the frequencies are normalized per locus, the maximum objective value would be 3.0 for MHC class I or II. Similarly, when concurrently optimizing on normalized haplotype or genotype frequencies of three populations, the maximum score would be 3.0.

The solver runs until it finds an optimal variable assignment, that corresponds to an optimal set of peptides, or is terminated after ten hours. From the chosen edges, we construct a subgraph of the HOG, which corresponds to the chosen circuit. This circuit is an Eulerian circuit of the subgraph due to our constraints. We traverse it once to create the vaccine sequence calculated by HOGVAX.

### 3.6 Evaluation Metrics

For the comparison between HOGVAX and OpiVax, we evaluated the respective vaccine candidates using the evaluation and optimization functions of both methods. We begin with formulating an evaluation metric based on the objective function of HOGVAX. Afterward, follow descriptions of EvalVax-Unlinked and EvalVax-Robust that are based on the explanation given in [39].

#### HOGVAX Evaluation Metric

We applied the objective function of HOGVAX to determine the fraction of covered alleles per locus. Following the terminology of Liu et al., we distinguish  $M_l$  alleles  $a_1, \dots, a_{M_l}$  per locus  $l$  of overall  $L$  loci. An allele  $a_i$  is covered, if at least one peptide from the vaccine set binds to it. The number of peptide hits of an allele is denoted by  $e(a_i)$  and given by

$$e(a_i) = \sum_{p \in P} e_l^p(a_i) \quad (6)$$

where  $P$  is the set of vaccine peptides and  $e_l^p(a_i)$  is the binary binding affinity prediction of allele  $a_i$  and peptide  $p$ . In our metric, we use a threshold of 0.638 and set all binding



predictions below to zero and all other values to one. The frequency of alleles  $F_l$  of locus  $l$  covered by peptides of  $P$  is calculated following our objective function by

$$F_l(P) = \sum_{i=1}^{M_l} \mathbb{1}\{e(a_i) \geq 1\} \cdot f_l(a_i) \quad (7)$$

where  $f_l(a_i)$  is the frequency with which allele  $a_i$  occurs in a given population. We assume that the frequencies of all alleles from the same locus sum up to one. Intuitively, this metric computes the fraction of individuals covered by locus  $l$ .

### EvalVax-Unlinked

Liu et al. presented EvalVax-Unlinked, an evaluation metric that uses single-allele frequencies and assumes no linkage between alleles from different loci. It computes the probability that an individual displays at least one peptide of the vaccine at at least one locus. The binding affinity prediction of a peptide  $p$  of the vaccine set  $P$  to bind to allele  $a_i$  of locus  $l$  is denoted by  $e_l^p(a_i)$ . Following from this, the probability that an allele  $a_i$  is targeted by at least one peptide from the set is

$$e_l(a_i) = 1 - \prod_{p \in P} (1 - e_l^p(a_i)) \quad (8)$$

Liu et al. set all binding predictions below the threshold of 0.638 to zero, but keep the precise values of the predictions above the threshold. In doing so, they hope to increase the diversity of the selected peptides.

To take diploidy into account, EvalVax-Unlinked uses combined allele frequencies  $F_l(a_i, a_j) = f_l(a_i)f_l(a_j)$ . Again, the frequencies of the same locus sum up to one. Liu et al. state that a homozygous diploid locus only contributes as a single allele, and define the probability of at least one binding peptide at locus  $l$  as follows.

$$B_l(a_i, a_j) = \begin{cases} 1 - (1 - e_l(a_i))(1 - e_l(a_j)), & \text{if } i \neq j \\ e_l(a_i) & \text{otherwise.} \end{cases} \quad (9)$$

From this, they infer that the probability that any allele of locus  $l$  presents at least one peptide of  $P$  is given by

$$F_l(P) = \sum_{i=1}^{M_l} \sum_{j=1}^{M_l} F_l(a_i, a_j) \cdot B_l(a_i, a_j) \quad (10)$$

The likelihood that an individual will be immunized depends on whether at least one peptide of the vaccine binds to at least one allele from one of the MHC loci. Thus, the

population coverage is computed by EvalVax-Unlinked through

$$Cov(P) = 1 - \prod_{l=1}^L (1 - F_l(P)) \quad (11)$$

### EvalVax-Robust

With EvalVax-Robust, Liu et al. introduce an evaluation metric that takes linkage disequilibrium between alleles of distinct loci into account. Based on the assumption that more presented peptides lead to more robust immunity, EvalVax-Robust calculates the probability that each person of a given population has at least  $N$  peptide-HLA hits. There are  $M_a, M_b, M_c$  alleles of locus  $a, b$ , and  $c$ , respectively. The frequency of a haplotype  $a_i b_j c_k$  is denoted by  $f(ijk)$  and all frequencies sum up to one. Liu et al. again take diploidy into account and assume independence between two haplotypes of a genotype. Thus, the frequency of a genotype is calculated by

$$F(ijk, xyz) = f(a_i b_j c_k, a_x b_y c_z) = f(ijk) \cdot f(xyz) \quad (12)$$

The binding affinity predictions are transformed into binary values based on the binding threshold of 0.638. The number of peptides binding to an allele  $a$  is given by  $e(a)$ , i.e., the number of peptide-HLA hits. A genotype consists of three to six unique alleles and Liu et al. assume no benefits from homozygous loci. Thus, the number of genotype hits is calculated from the amount of peptide-HLA hits of the unique alleles that form the genotype:

$$C(ijk, xyz) = C(a_i b_j c_k, a_x b_y c_z) = \sum_{a \in \{a_i b_j c_k\} \cup \{a_x b_y c_z\}} e(a) \quad (13)$$

The probability that each individual has exactly  $k$  peptide-HLA hits is

$$P(n = k) = \sum_{i=1}^{M_a} \sum_{j=1}^{M_b} \sum_{k=1}^{M_c} \sum_{x=1}^{M_a} \sum_{y=1}^{M_b} \sum_{z=1}^{M_c} F(ijk, xyz) \cdot \mathbb{1}\{C(ijk, xyz) = k\} \quad (14)$$

The probability that each individual has at least  $N$  peptide-HLA hits is computed by EvalVax-Robust as follows:

$$P(n \geq N) = \sum_{k=N}^{\infty} P(n = k) \quad (15)$$

Additionally, from the exact peptide-HLA hits of a diploid individual, we calculate the expected number of hits. In Section 4.2, we discuss the outcomes of HOGVAX and OptiVax based on the metrics presented in this section.

## 4 Experimental Results

In this section, we present the performance results of HOGVAX. We begin with showing benchmarking results of the construction time of the HOG and runtimes of the Gurobi solver for three different ILPs. Afterward, we compare HOGVAX to OptiVax-Unlinked and OptiVax-Robust for MHC class I and II separately on the metrics presented in the previous section. We further show the benefits of a combined MHC class I and II vaccine. Lastly, we explain the results of our proteasomal cleavage prediction analysis of vaccines designed by HOGVAX.

All computations were executed on the workstation of the Chair of Algorithmic Bioinformatics at Heinrich Heine University Düsseldorf with the following specification: AMD EPYC 7742 64-Core processor with 128 threads, 1 TiB DDR4 RAM, running Debian Buster with Linux kernel 5.8.

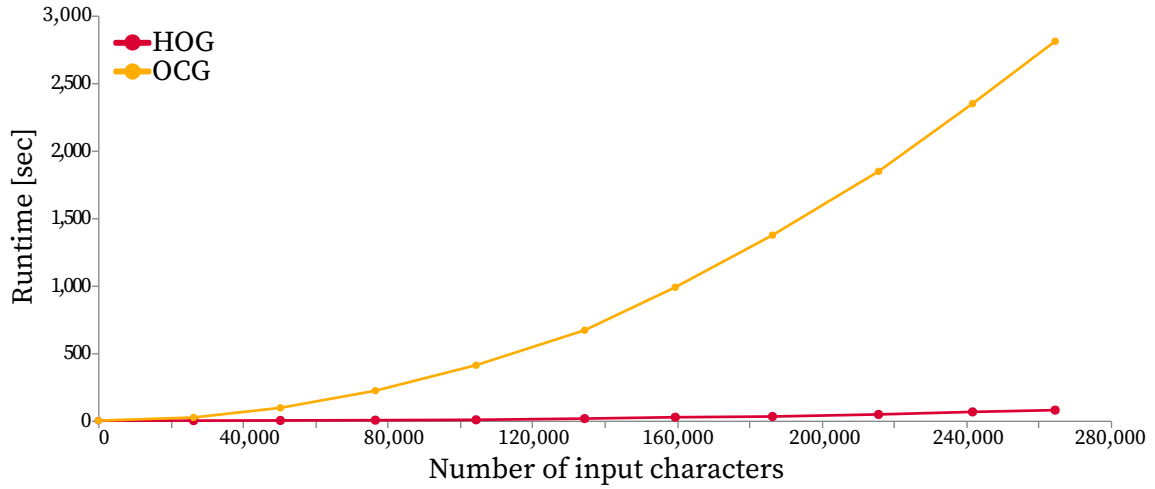
### 4.1 Runtime Comparisons

In this section, we illustrate experimental runtime comparisons of constructing the HOG and the OCG. We also demonstrate the differences in time used by Gurobi to solve the MSKS on a HOG and an OCG. We compare these results to an optimized concatenation approach, which we present later in this section.

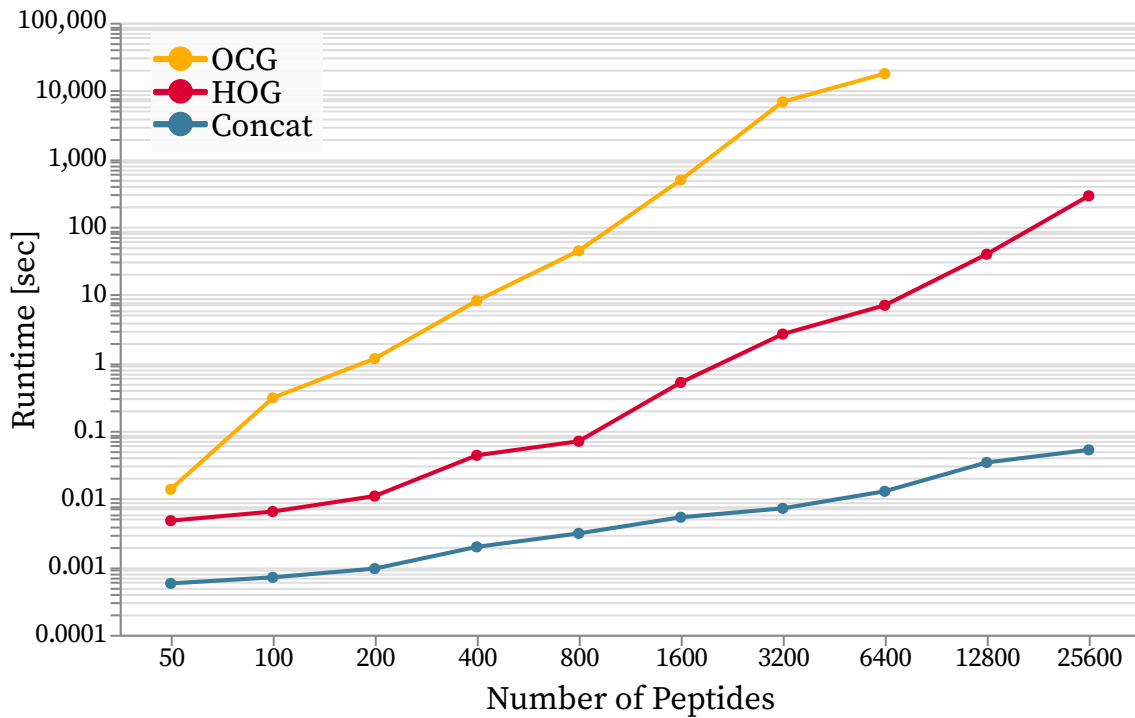
For the construction comparisons, we created instances of different sizes from the 29 403 unfiltered MHC class I peptides. The computations started with five peptides and were iteratively extended by 3000 peptide sequences. The results, given in Figure 11, confirm that the HOG is constructed in linear time, and the OCG requires quadratic runtime.

In Figure 12, we present the runtimes of the Gurobi solver for three ILPs. The first two ILPs solve the MSKS problem on an OCG and a HOG. The third ILP selects an optimal set of peptides based on the optimization function of the MSKS problem, but instead of computing a superstring from overlaps, we compute the concatenation of the selected sequences that must not exceed the length of  $k$ . Thus, we do not need any underlying data structure here. Still, this problem is NP-hard and the proof is similar to the NP-proof for the MSKS problem in Section 3.2. The ILP formulation is given in Appendix A.2.

For our measurements, we used the set of MHC class I peptides without glycosylated, cleaved, and self-peptides. We started with 50 peptides and doubled the number of peptides after each instance. With the increase of the input size, the runtime of each method also increased linearly, where the optimized concatenation approach was the fastest. However, its ILP formulation consists of much fewer constraints and variables compared to the other two combinatorial models. In contrast, the OCG constitutes the



**Figure 11:** Comparison of runtimes to construct HOG and OCG for different instance sizes. We used peptides generated for the MHC class I vaccine design. Each point is the average runtime of five computations. The first point corresponds to graphs constructed from five peptides. Each subsequent point contains additional 3 000 peptides until the last data point corresponds to the full 29 403 input peptides.



**Figure 12:** Runtime comparison of three ILPs. The x-axis gives the number of peptides included in the input data. The runtime on the y-axis is given in seconds on a logarithmic scale. The yellow graph shows the runtimes of the ILP using the OCG to solve the MSKS problem. The red graph corresponds to the ILP solving the MSKS by using a HOG. The blue graph is a simplified version of the MSKS that computes the concatenation of peptides which should not exceed the limit of  $k$ . Further, it does not use any underlying data structure.

most complex data structure and the corresponding ILP has the highest number of variables and constraints, which was noticeable in the runtimes. Moreover, Gurobi could not finish the computations of the last two instances within a time limit of ten hours. We let the instance of 12 800 peptides run again, and the offset between the best objective solution and the best bound was 2.3007 % after 50 hours, whereupon we terminated the process.

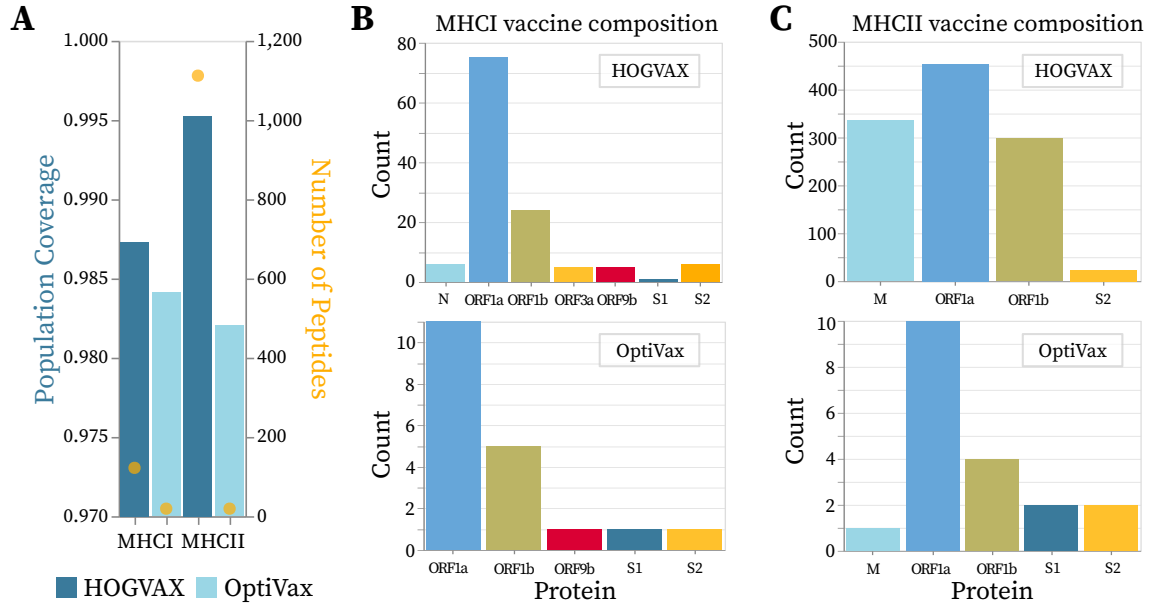
The results make clear that HOGVAX outperforms the OCG approach. For example, considering the instance of size 6 400, HOGVAX was more than 2 500 times faster than the OCG approach. Thus, the HOG is not only faster in construction time, but also when it comes to solving the ILPs.

## 4.2 HOGVAX Significantly Increases Vaccine Robustness

We compared HOGVAX-designed MHC class I and class II vaccines with the sets of vaccine peptides selected by OptiVax. First, we considered single-allele frequencies and compared HOGVAX to OptiVax-Unlinked. Afterward, we used haplotype data as input to HOGVAX and compared it to OptiVax-Robust. We let OptiVax run first and filter the set of peptides, which we then used as input to HOGVAX. For MHC class I, we used peptides of lengths 8 to 10 and filtered out self-peptides, peptides with non-zero mutation or glycosylation probability, and peptides of cleavage regions. For the MHC class II vaccine design, we used peptides of lengths 13 to 25 and removed self-peptides, peptides of cleavage regions, peptides with non-zero glycosylation probability, and peptides with a mutation probability greater than 0.001. To determine the maximum length of our vaccine, we used the output peptides from OptiVax, concatenated them, and calculated the length of the resulting string. This value was then used as input  $k$ . We applied all metrics presented in Section 3.6 for the comparisons, where we assume that the proteasome correctly cuts our vaccine into all peptides chosen by our method.

### 4.2.1 Population Coverage Optimization on Allele Data

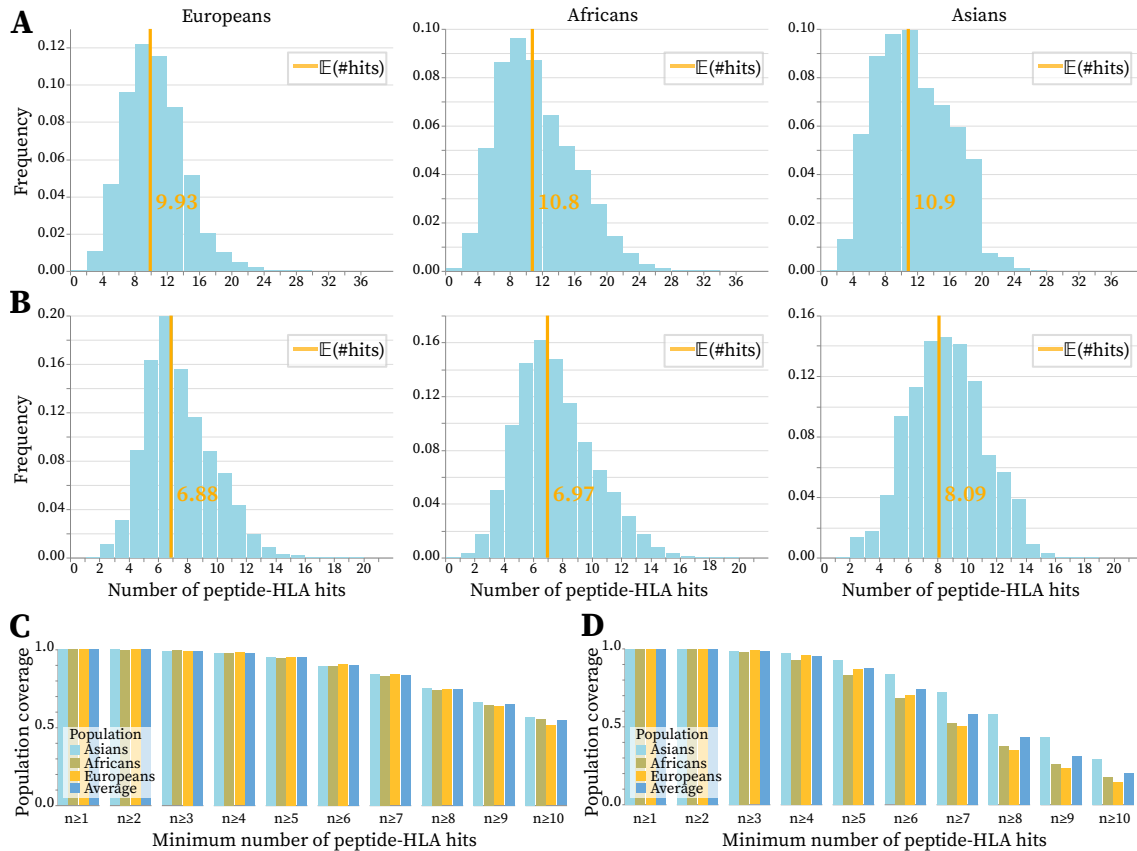
For the comparison of single-allele MHC class I vaccines, we began with computing an optimal set of peptides using OptiVax-Unlinked. After OptiVax-Unlinked filtered the input peptides, a set of 4 512 candidate peptides remained. We used the  $IC_{50}$  predictions by NetMHCpan-4.1 for alleles of HLA-A, -B, and -C over the average world population and transformed them into binary values. As done by Liu et al. [39], we ran OptiVax-Unlinked for 19 iterations resulting in a vaccine candidate of 19 peptides. From their concatenation, we inferred  $k = 170$ . We let HOGVAX create a vaccine candidate, i.e., *hogvaxine*, from the 4 512 filtered peptides. Each substring of a selected peptide, which was also in the input, was added to the final output as well. This is due to our assumption that



**Figure 13:** Comparison of HOGVAX and OptiVax-Unlinked vaccines on single-allele data. **A** Shows the population coverage of the average world population predicted with EvalVax-Unlinked and the sizes of the different vaccines. **B** Peptide viral protein origins of the MHC class I vaccines. **C** Peptide viral protein origins of the MHC class II vaccines.

the proteasome cuts out all possible peptides from our vaccine sequence. The HOGVAX MHC class I candidate consisted of 122 peptides from seven distinct SARS-CoV-2 proteins, which was about six times the number of the OptiVax peptides. In contrast, OptiVax-Unlinked selected peptides from five different proteins. In both methods, peptides from the proteins ORF1a and ORF1b dominated the sets. The exact compositions are shown in Figure 13 B.

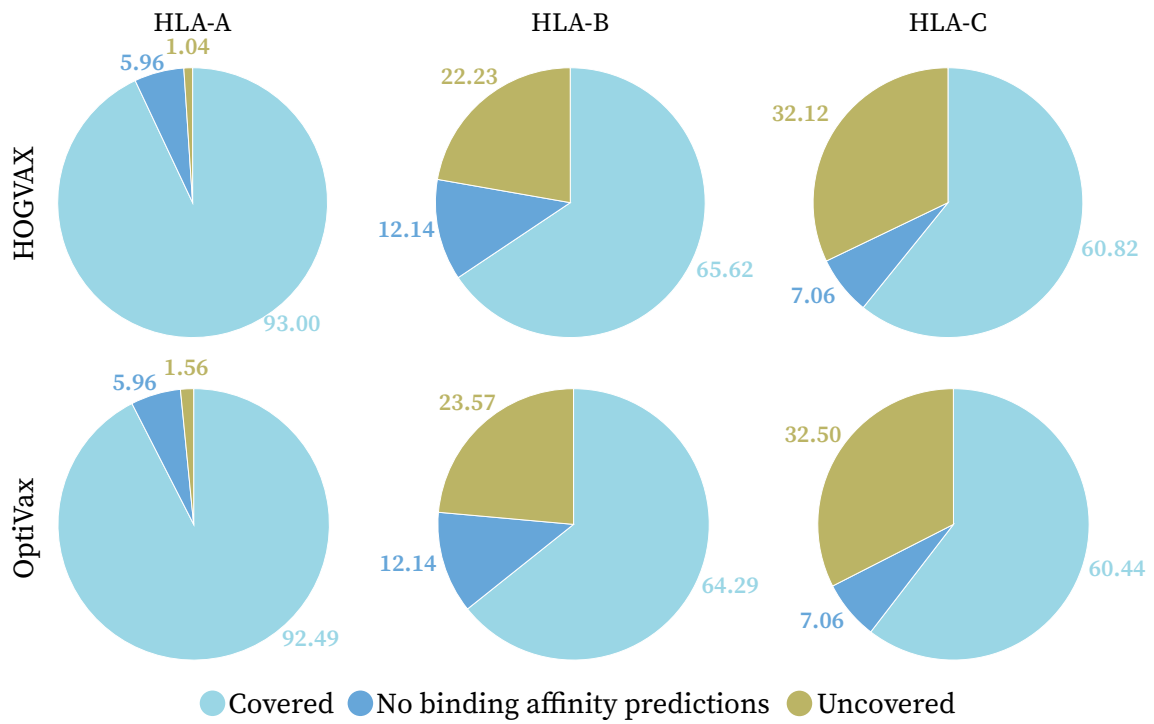
With EvalVax-Unlinked, we achieved a population coverage of 98.41 % for the MHC class I vaccine set of OptiVax-Unlinked and slightly higher coverage of 98.73 % for the hogvaxine, see Figure 13 A. The evaluation with EvalVax-Robust led to further promising results regarding the robustness of the peptide set chosen by HOGVAX compared to the OptiVax-Unlinked vaccine set. Here, we examined the performance of the vaccines on haplotype data as stated in Section 3.6. The probabilities of each number of observed peptide-HLA hits for HOGVAX and OptiVax-Unlinked are presented in Figure 14 A and B, respectively. HOGVAX achieved an overall higher number of expected peptide-HLA hits in each of the three populations of Europeans, Africans, and Asians. Moreover, the maximum value of observed hits is higher by approximately ten hits compared to the OptiVax-Unlinked outcome. This is also evident in the predictions calculated with EvalVax-Robust that are presented in Figure 14 C and D for HOGVAX and OptiVax-Unlinked, respectively. With the HOGVAX-designed vaccine, we obtained 99.998 % coverage for at least one peptide hit, 95.09 % for at least five hits, and 54.44 % for at least ten hits for the average of all



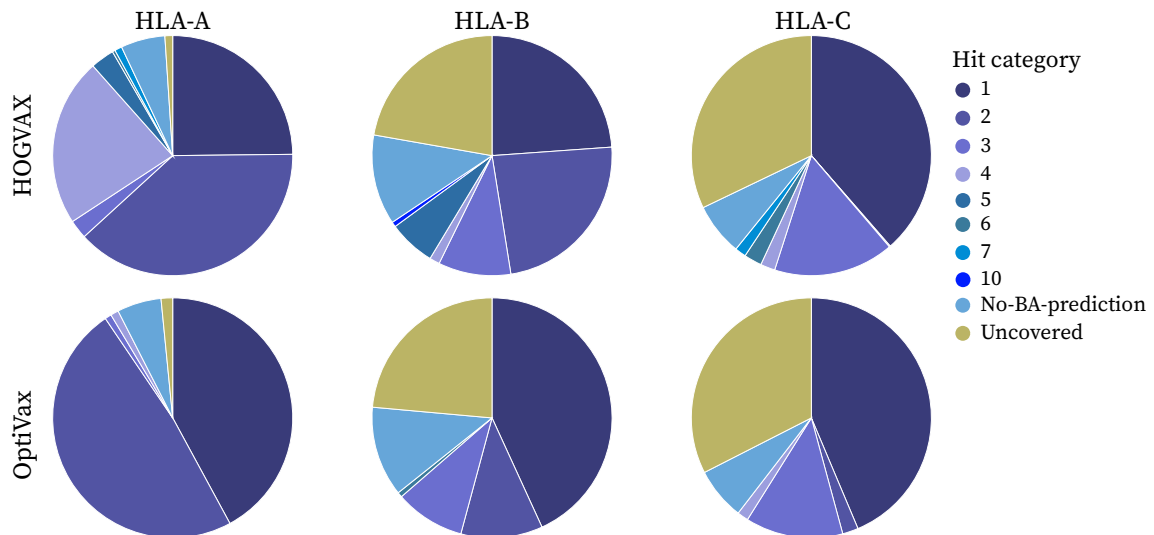
**Figure 14:** Evaluation of population coverage of the MHC class I vaccine candidates optimized on single-allele data for populations self-reporting as having European, African, or Asian ancestry. **A** HOGVAX and **B** OptiVax-Unlinked distribution of the number of peptide-HLA hits and expected number of hits per individual for the three populations. **C** HOGVAX and **D** OptiVax-Unlinked population coverage computed with EvalVax-Robust for the three populations and average values.

three populations. In contrast, we observed a stronger decrease in the coverage achieved by the OptiVax-Unlinked vaccine candidate for higher hit counts. For the average population, the OptiVax-Unlinked peptides were predicted with 99.994 % coverage for at least one peptide-HLA hit, 87.538 % coverage for at least five hits, and 20.377 % coverage for at least ten hits.

Furthermore, we evaluated the population coverage based on covered alleles per locus and their corresponding allele frequencies of the average world population. For this, we used the metric based on the objective function of HOGVAX. The results are given in Figure 15. At each locus, we observed a small increase of alleles covered by the hogvaxine compared to the coverage of the OptiVax-Unlinked candidate. Although the improvements here seem rather negligible, the results of HOGVAX compared to OptiVax-Unlinked are promising when measuring the exact number of peptide-HLA hits per allele, see Figure 16. We computed the number of hits per allele and summed up the frequencies of those alleles covered by the same amount of peptides. Thereby, we gain insights into



**Figure 15:** Coverage of MHC class I loci based on HOGVAX objective function. *Covered* gives the fraction of individuals in the population with at least one peptide-HLA hit. *No binding affinity predictions* gives the summed frequencies of alleles for which we cannot predict the binding behavior of peptides with NetMHCpan-4.1. *Uncovered* is the fraction of alleles to which no peptide of the vaccine sets binds. The top row shows results for HOGVAX, the bottom row for OptiVax-Unlinked. The columns present the loci.



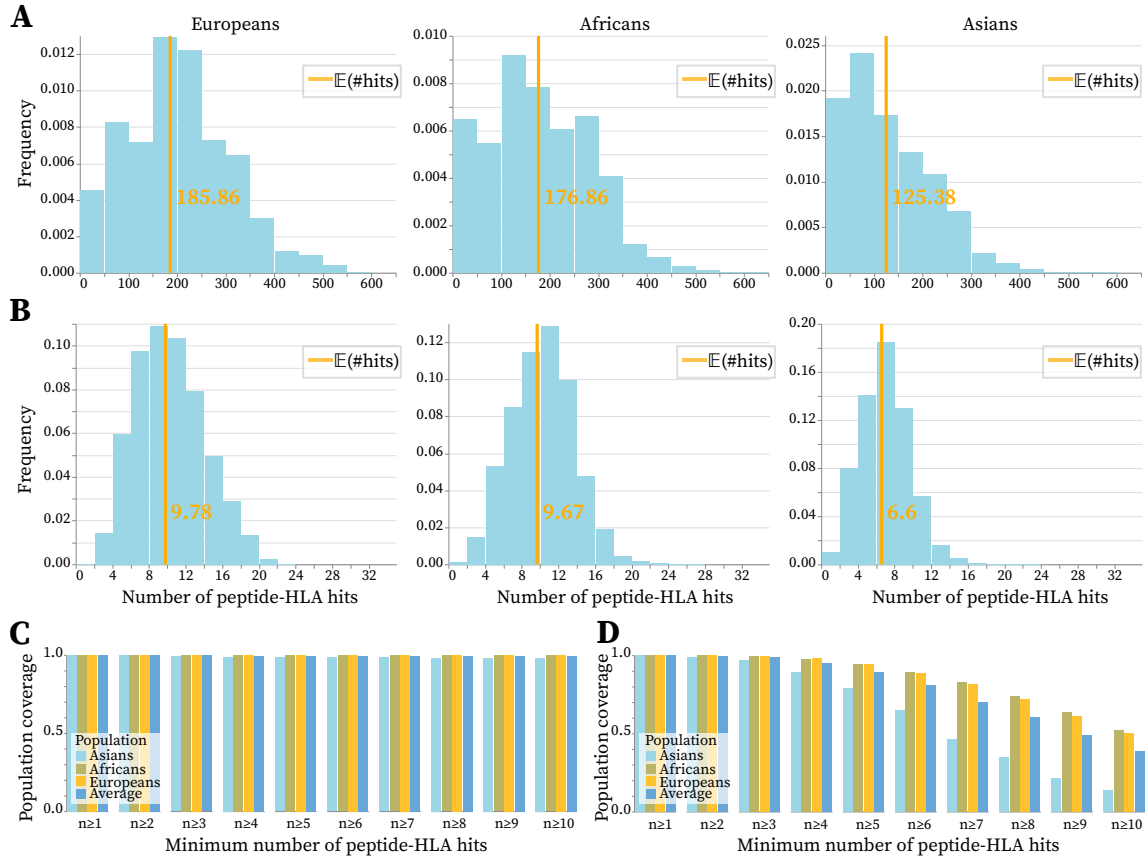
**Figure 16:** Hits of MHC class I loci based on HOGVAX objective function. We counted the exact number of hits induced by the vaccine sets and summed up the frequencies of the alleles with the same number of hits. Thereby we got the fraction of the population covered by  $n$  many hits. Certainly, the fraction of uncovered alleles and alleles without BA predictions is equivalent to the fractions given in Figure 15.



the minimum number of hits a population is covered with for each locus separately. In most cases, the vaccine candidate of OptiVax-Unlinked induced a single peptide-HLA hit, whereas HOGVAX showed a wider range of hit numbers as well as a higher maximum number of hits. For example, 65.62 % of the population is covered by at most ten hits at locus HLA-B by the HOGVAX peptide set, while the OptiVax-Unlinked peptides cover 64.29 % of the population at locus HLA-B with at most six peptide-HLA hits. Additionally, Figures 15 and 16 provide information about the fraction of uncovered alleles and the fraction of alleles for which we do not have binding affinity predictions, as NetMHCpan-4.1 is not able to make any predictions for the corresponding alleles.

For the MHC class II vaccine, a set of 37 435 peptides remained after filtering. We used NetMHCIIpan-4.0 to predict binding affinities for alleles of HLA-DP, -DQ, and -DR and converted them into binary values again. The allele frequencies used are taken from the average world population. We created a set of 19 peptides with OptiVax-Unlinked whose concatenated string has a length of 322 amino acids. Thus, we called HOGVAX with  $k = 322$  and obtained a vaccine set of 1 112 peptides. With EvalVax-Unlinked, we predicted a population coverage of 98.2 % for the OptiVax-Unlinked-designed vaccine, and a slightly higher coverage of 99.52 % for the hogvaxine of MHC class II, see Figure 13 A. Although the amount of peptides in our vaccine candidate is almost 60 times greater than that of OptiVax-Unlinked, fewer different proteins are represented in the hogvaxine. More precisely, HOGVAX renounced peptides of the S1 unit of the spike protein and the distribution of peptides from distinct proteins is also different from the distribution in the OptiVax-Unlinked vaccine. For example, the second largest fraction of peptides in the hogvaxine derives from the M protein, where the second largest number of peptides chosen by OptiVax-Unlinked originates from ORF1b. The exact compositions are given in Figure 13 C.

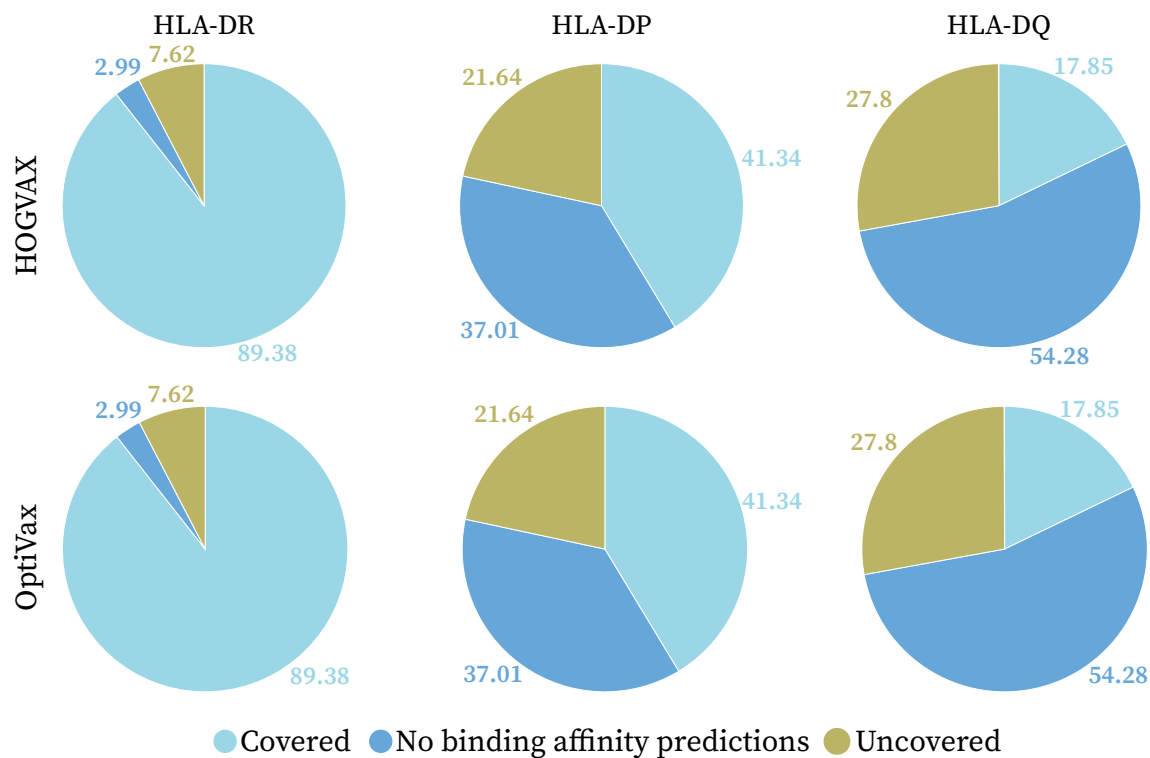
The results computed with EvalVax-Robust over the three ethnic groups of Europeans, Africans, and Asians yielded significantly better results for our method than for OptiVax-Unlinked. Surprisingly, HOGVAX achieved expected numbers of 185.86, 176.86, and 125.38 peptide-HLA hits per individual of the European, African, and Asian population, respectively. These are almost 20 times better than the expected values computed for the vaccine peptides of OptiVax-Unlinked, which are 9.78, 9.67, and 6.6 for Europeans, Africans, and Asians, respectively. Further, the maximum number of observed hits from the hogvaxine was 628, and 31 from the OptiVax-Unlinked peptides. However, the corresponding frequencies, with which these numbers were observed, are too low to be visible in the plot given in Figure 17 A and B for HOGVAX and OptiVax-Unlinked, respectively. We obtained 99.975 % EvalVax-Robust coverage over the three ethnic groups with at least one



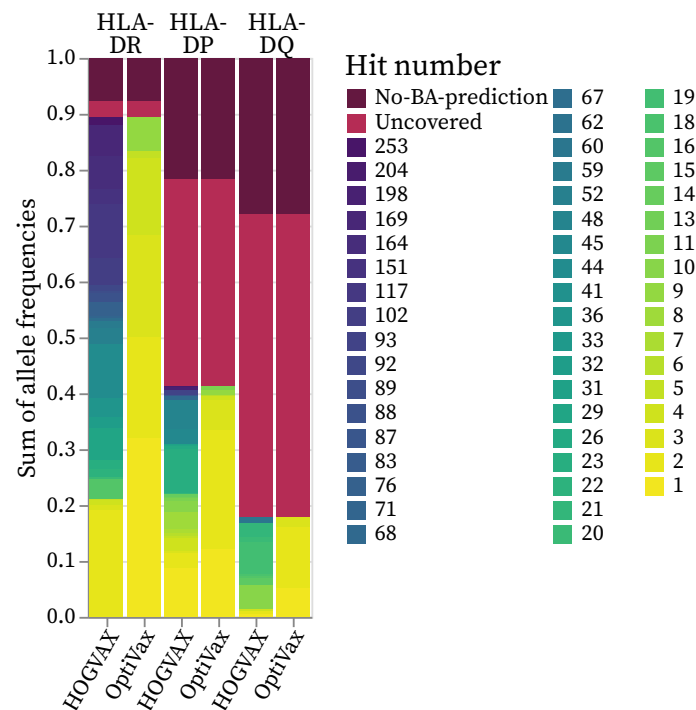
**Figure 17:** Evaluation of population coverage of the MHC class II vaccine candidates for populations self-reporting as having European, African, or Asian ancestry. **A** HOGVAX and **B** OptiVax-Unlinked distribution of the number of per-individual peptide-HLA hits and expected number of hits. **C** HOGVAX and **D** OptiVax-Unlinked population coverage computed with EvalVax-Robust for the three populations and average values.

peptide-HLA hit for our set of HOGVAX-peptides as well as for the OptiVax-Unlinked peptides, see Figure 17 C and D. However, the hogvaxine still provided 99.395 % coverage with at least five peptide-HLA hits per individual, and 99.215 % coverage with at least ten peptide hits. In contrast, the OptiVax-Unlinked vaccine achieved 89.36 % EvalVax-Robust coverage with at least five peptide-HLA hits, and 38.639 % with ten hits. Thus, the coverage by the OptiVax-Unlinked candidate rapidly decreased with an increasing number of required hits, while the vaccine set of HOGVAX maintained a high level of population coverage.

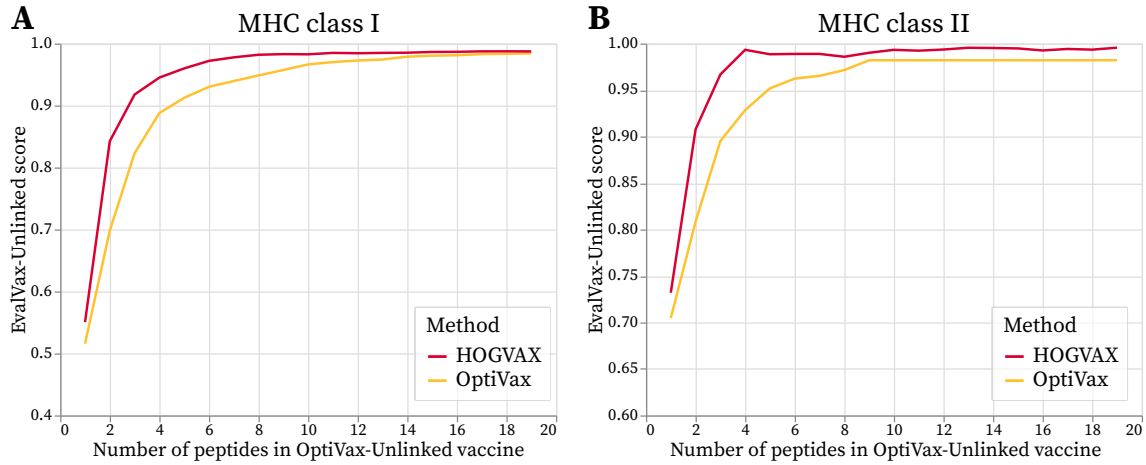
When evaluating the locus-wise allele coverage, HOGVAX and OptiVax-Unlinked obtained the exact same population coverage, see Figure 18. However, calculating the fraction of individuals covered with a specific number of hits at each locus, we observed that HOGVAX induced a significantly greater amount of peptide-HLA hits. This is shown in Figure 19, where darker colors indicate higher hit numbers. We can also observe that large proportions of HLA-DP and -DQ alleles are either uncovered or without any binding affinity predictions.



**Figure 18:** Coverage of MHC class II loci based on HOGVAX objective function. *Covered* is the fraction of individuals covered by at least one peptide of the vaccine at each locus separately. *No binding affinity predictions* give the summed frequencies of alleles with no available predictions from NetMHCIIpan-4.0. *Uncovered* is the fraction of alleles without any peptide hit.



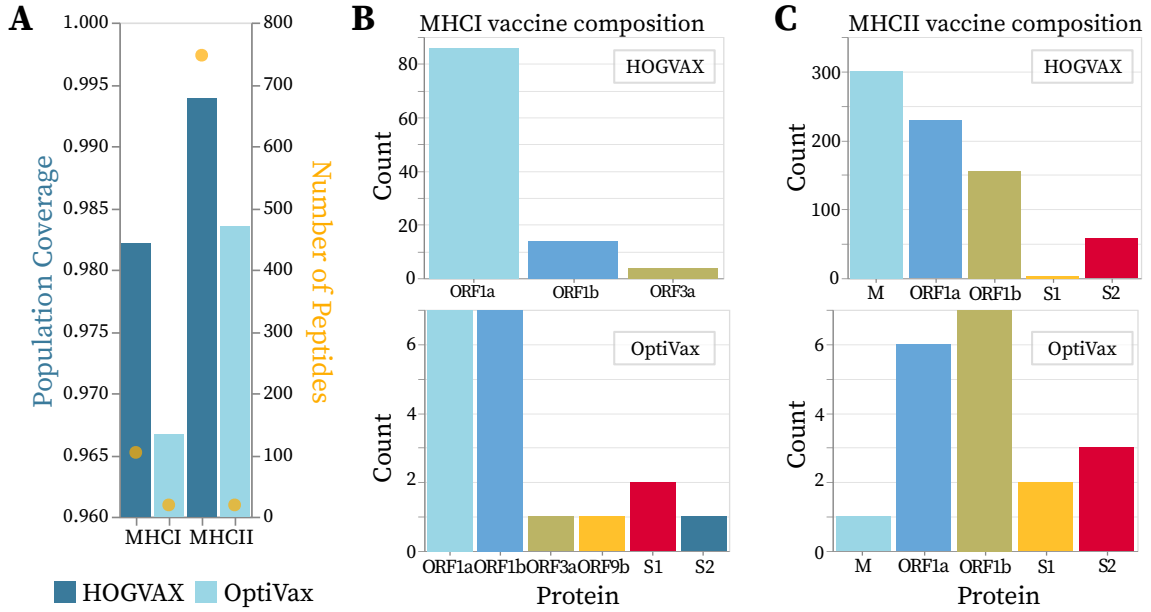
**Figure 19:** Hits of MHC class II loci based on HOGVAX objective function. We counted the exact number of hits per allele and summed up the allele frequencies of those alleles with the same number of peptide-HLA hits induced by the vaccine. We also show the summed frequencies of uncovered alleles and alleles without available BA predictions.



**Figure 20:** Comparison of HOGVAX and OptiVax-Unlinked on MHC class I and class II data to evaluate which tool reached the maximal coverage first. With the same maximal length of the vaccine sequences, HOGVAX reached the maximal population coverage much faster.

We were curious if HOGVAX could reach the maximal population coverage faster than OptiVax-Unlinked. For this, we used OptiVax-Unlinked to create 19 vaccine candidates that differ in the number of peptides included in the set. The smallest set consisted of a single peptide and for each new candidate, we allowed OptiVax-Unlinked to add another peptide. The largest set contained 19 peptides. From each of the vaccines, we used the length of the concatenated peptides as input for HOGVAX. We used EvalVax-Unlinked to predict the population coverage of each vaccine instance. The results for MHC class I as well as MHC class II are given in Figure 20. We observed, that HOGVAX reached higher population coverage more rapidly than OptiVax-Unlinked even though the computed vaccines have the same maximal lengths. The small coverage drop in Figure 20 B can be explained by the fact that HOGVAX was terminated after ten hours and for some instances, Gurobi did not find the optimum in the given time. For the comparison we used the best solution found until termination. Still, HOGVAX performed better than OptiVax.

For both, MHC class I and MHC class II, we observed that the vaccine set designed by HOGVAX led to a significantly higher population coverage and vaccine robustness when optimizing the coverage based on single-allele frequencies. Although the constructed vaccine sequences of HOGVAX and OptiVax-Unlinked had the same maximal length, we outperformed OptiVax-Unlinked with our superstring approach on all metrics. In the next section, we evaluate the performance of HOGVAX and OptiVax-Robust vaccines that were optimized using haplotype frequencies.

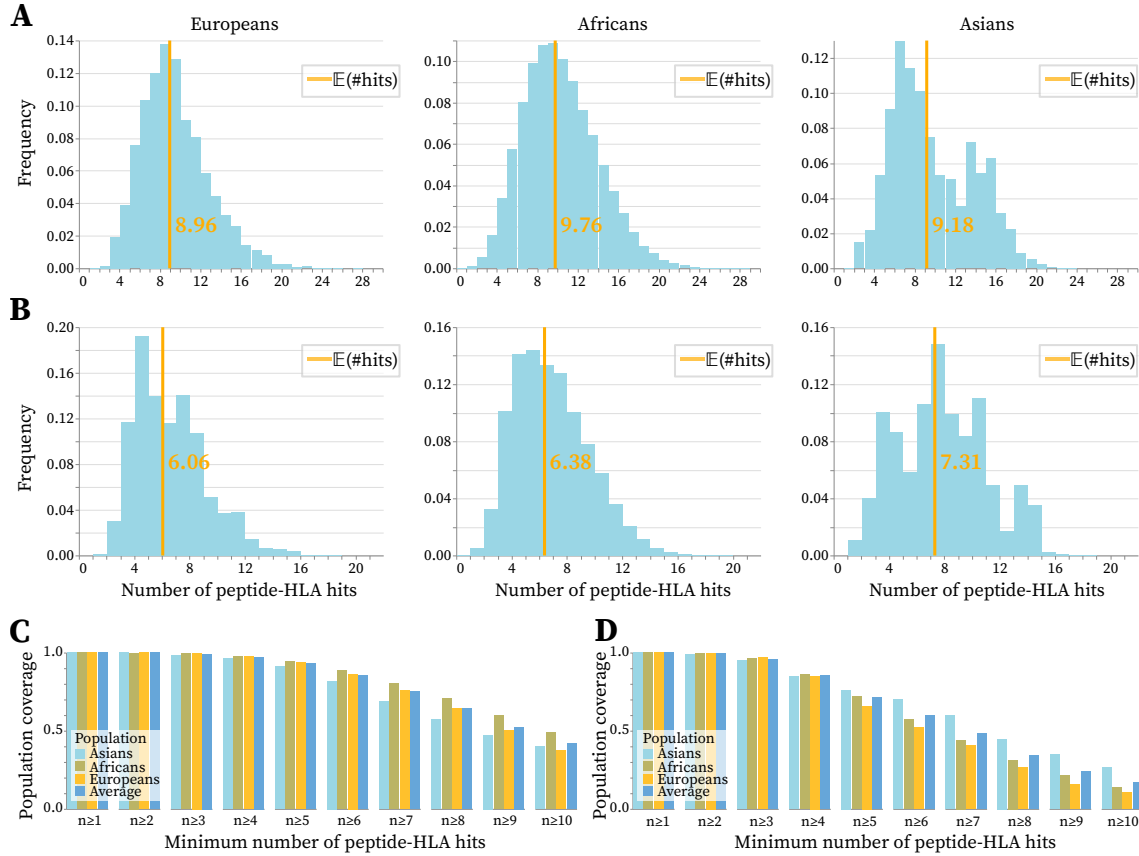


**Figure 21:** Comparison of HOGVAX and OptiVax-Robust haplotype vaccine population coverage and compositions. **A** EvalVax-Unlinked predicted population coverage of an average world population and number of peptides included in each vaccine. **B** Peptide viral protein origins of haplotype MHC class I vaccines. **C** Peptide viral protein origin of haplotype MHC class II vaccines.

#### 4.2.2 Population Coverage Optimization on Haplotype Data

For our comparison to OptiVax-Robust, we used haplotype data as input to HOGVAX and maximized the population coverage based on haplotype frequencies of three ethnic groups, i.e., Europeans, Africans, and Asians. OptiVax-Robust actually requires genotype frequencies as input, although Liu et al. speak of haplotype frequencies in [39]. The genotype frequencies were generated by multiplying the frequencies of all pairwise haplotype combinations. This resulted in more than one million genotypes and the corresponding binding affinity matrix became very large. Gurobi did not start the optimization process after several days of running and we observed an error that was likely caused by exceeding the available memory. Thus, we focused on comparing HOGVAX and OptiVax-Robust using haplotype frequencies. For this, we had to apply small modifications to OptiVax-Robust to properly run on haplotype data that did not include changes to the optimization framework itself. We used the binding affinity predictions of NetMHCpan-4.1 and NetMHCIIpan-4.0 and transformed them as described in Section 3.4 to model haplotype binding affinities. The modified binding affinities were binarized based on the 50 nM threshold.

For the MHC class I vaccine design 4519 peptides remained after OptiVax-Robust applied the filter criteria. With OptiVax-Robust, we designed a set of 19 peptides from whose concatenation we derived our input parameter  $k = 174$ . Within the given limit, HOGVAX created a vaccine sequence of length 172 amino acids that contained 104 of our input pep-



**Figure 22:** Comparison between HOGVAX and OptiVax-Robust MHC class I vaccines optimized on haplotype data for populations self-reporting as having European, African, or Asian ancestry. Distribution and expected number of hits per individual for **A** HOGVAX and **B** OptiVax-Robust. EvalVax-Robust population coverage for three ethnic groups and average values for **C** HOGVAX and **D** OptiVax-Robust.

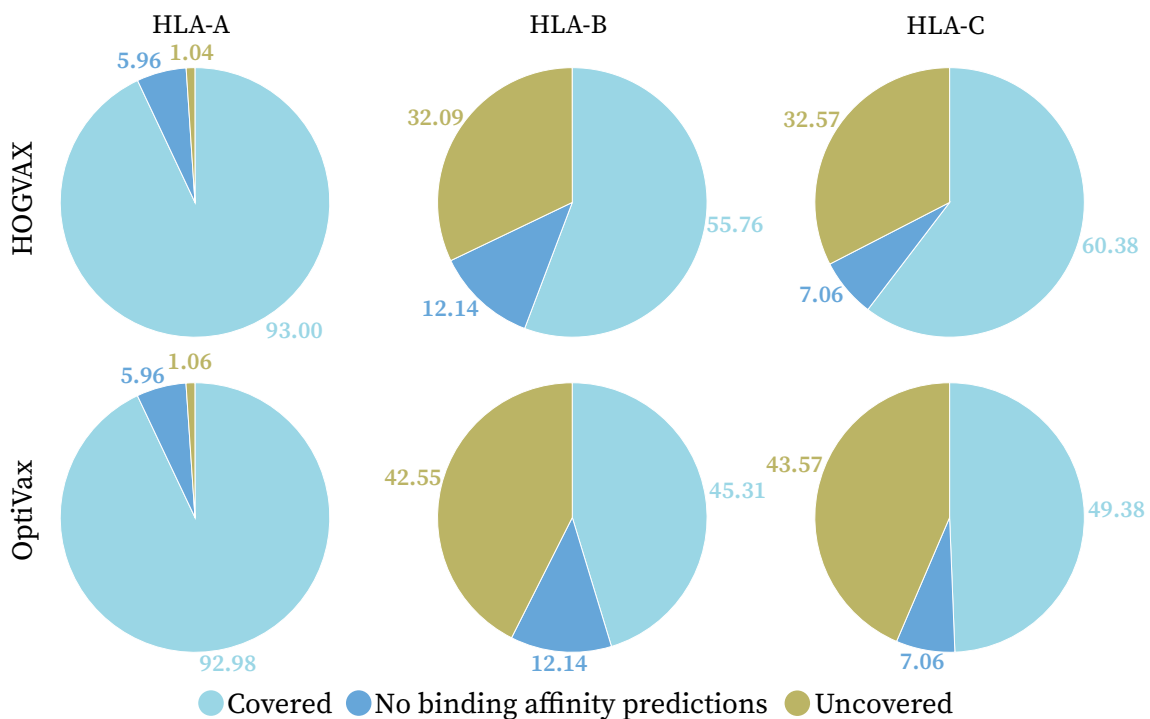
tides, see Figure 21 A. Here, we observed a strong difference in the compositions and less diversity in the HOGVAX-designed peptide set compared to the OptiVax-Robust vaccine candidate. HOGVAX included peptides from three different SARS-CoV-2 proteins, while the peptides chosen by OptiVax-Robust originated from six different regions. Further, the hogvaxine was rather homogeneous with about 80 % of the peptides belonging to ORF1a. The compositions are given in Figure 21 B.

For the average world population, we predicted a coverage of 96.67 % with EvalVax-Unlinked for the MHC class I OptiVax-Robust vaccine candidate. The MHC class I peptides chosen by HOGVAX achieved a higher EvalVax-Unlinked coverage of 98.21 %, see Figure 21 A. The distribution of peptide hits induced by the hogvaxine led to further promising results. Histograms of per-individual peptide-HLA hits for three populations self-reporting as having European, African, or Asian ancestry are given in Figure 22 A and B for HOGVAX and OptiVax-Robust. The expected numbers of HOGVAX vaccine hits were 8.96, 9.76, and 9.18 for Europeans, Africans, and Asians, respectively. These were higher by at least one hit compared to the expected per-individual peptide-HLA hits

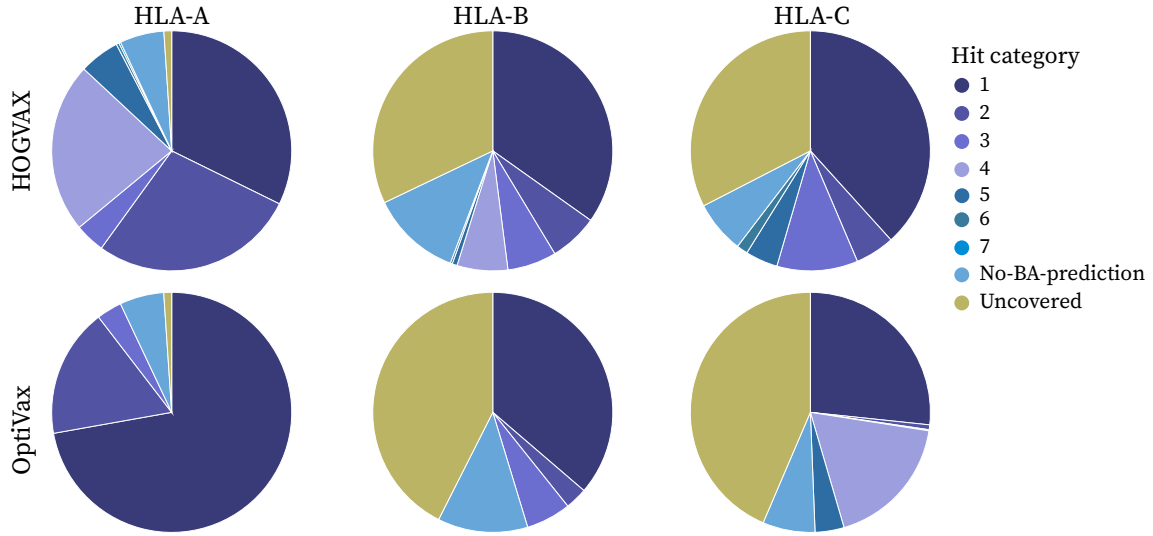
for the OptiVax-Robust candidate of 6.06, 6.38, and 7.31 for Europeans, Africans, and Asians, respectively.

From the evaluation with EvalVax-Robust, we again obtained better results with HOGVAX regarding the robust presentation of vaccine peptides, see Figure 22 C and D. The HOGVAX-designed vaccine set was predicted with 99.998 % EvalVax-Robust coverage over the average ethnic group of Europeans, Africans, and Asians with at least one peptide-HLA hit per individual. Furthermore, the peptides chosen by HOGVAX achieved 93.122 % coverage with at least five peptide-HLA hits, and 42.176 % coverage with at least ten hits. In contrast, OptiVax-Robust showed a stronger decrease in population coverage. The OptiVax-Robust vaccine set provided 99.997 % coverage with at least one peptide-HLA hit, 71.353 % with five hits, and 16.889 % with ten hits per individual on the average population.

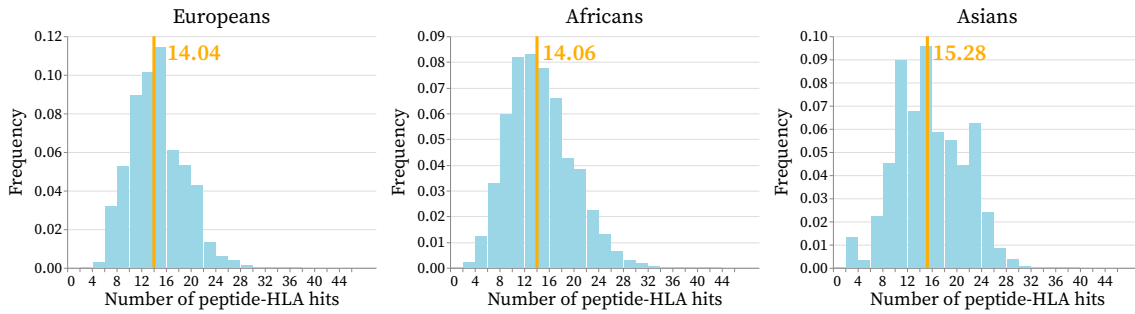
When evaluating the fraction of covered alleles of each MHC class I locus, HOGVAX achieved a higher locus-wise population coverage than OptiVax-Robust, see Figure 23. For example, 60.38 % of the average world population is covered from peptides chosen by HOGVAX that bind to HLA-C alleles, whereas the OptiVax-Robust peptide set only covered 49.38 % of the population through peptides binding to the HLA-C locus. Additionally,



**Figure 23:** coverage of MHC class I loci based on the HOGVAX evaluation metric. *Covered* gives the proportion of individuals in the average world population with at least one peptide-HLA hit. *No binding affinity predictions* gives the summed frequencies of alleles for which NetMHCpan-4.1 could not compute any predictions. *Uncovered* gives the proportion of alleles not covered by any peptides of the vaccine. The columns present the loci.



**Figure 24:** Hits for MHC class I loci based on the HOGVAX evaluation metric. We counted the exact number of hits induced by the vaccine candidates and summed up the frequencies of the alleles with the same number of hits. This gives us the proportion of the average world population covered by  $n$  hits. Certainly, the proportion of uncovered alleles and alleles without BA predictions are equivalent to the fractions given in Figure 23.

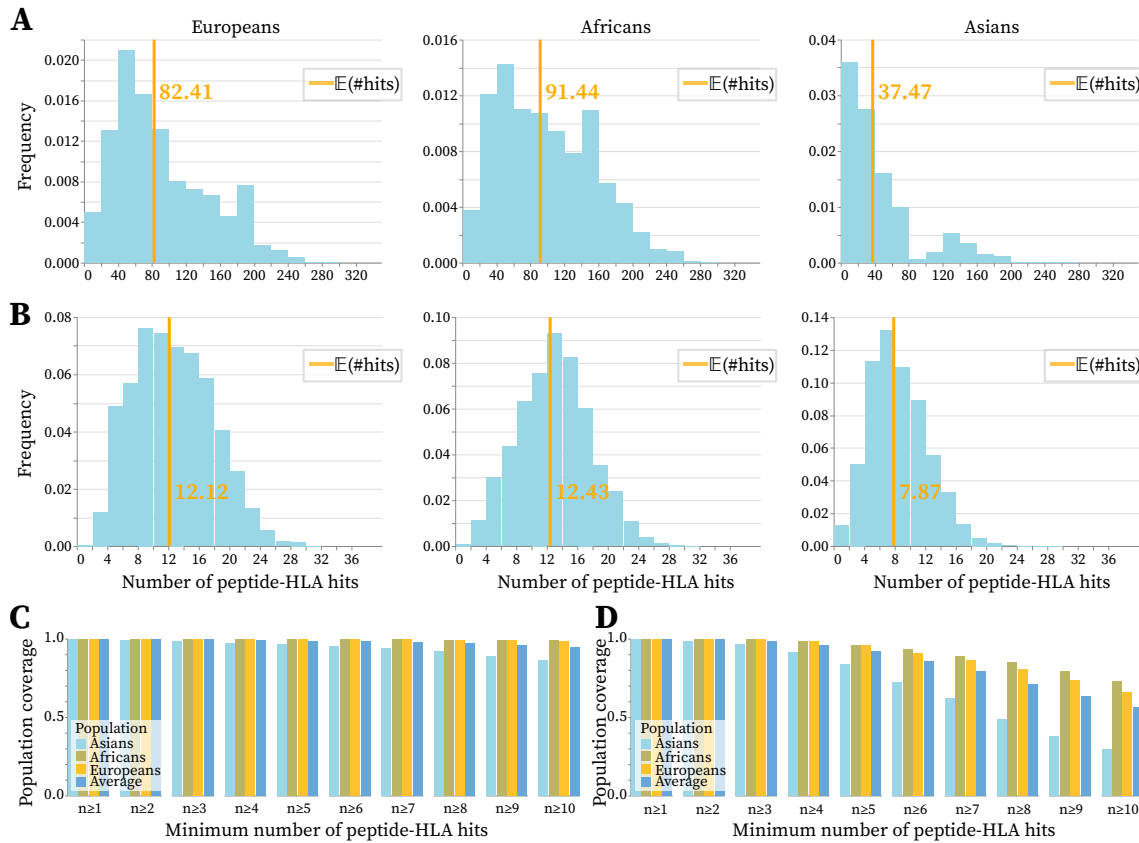


**Figure 25:** Distribution of the per-individual peptide-HLA hits and expected number of hits for the HOGVAX MHC class I vaccine candidate which was designed using a minimum required number of 5 hits to cover a haplotype for populations self-reporting as having European, African, or Asian ancestry.

we measured the fraction of the average world population that is covered by a specific number of hits at each locus. As shown by Figure 24, HOGVAX generally led to higher numbers of peptide-HLA hits with a higher maximum value compared to the OptiVax-Robust vaccine candidate.

We additionally tested the MHC class I computations for HOGVAX with a minimum number of required peptide-HLA hits of 5. Thus, a haplotype is only covered, if at least five peptides from the vaccine set are predicted to bind to the haplotype. This led to a substantial increase in the expected number of per-individual peptide-HLA hits beyond the five required hits in all three populations, as shown in Figure 25. With EvalVax-Robust, we achieved a predicted coverage of 85.529 % with at least ten peptide-HLA hits per individual of the average population. Further results are given in Appendix B.1.





**Figure 26:** Comparison between HOGVAX and OptiVax-Robust MHC class II vaccine candidates designed using haplotype data for populations self-reporting as having European, African, or Asian ancestry. **A** HOGVAX and **B** OptiVax-Robust distribution of per-individual peptide-HLA hits and expected number of hits. **C** HOGVAX and **D** OptiVax-Robust population coverage computed with EvalVax-Robust.

For the MHC class II vaccine computations, we obtained a set of 37 435 peptides after filtering with OptiVax-Robust. We used the binding affinities predicted by NetMHCIIpan-4.0 and binarized them such that all values smaller than 0.638 were set to zero and all others to one. With OptiVax-Robust, we created a set of 19 peptides from which we inferred an input limit of  $k = 324$ . HOGVAX computed a set of 747 peptides that originated from the same five proteins as the peptides chosen by OptiVax-Robust, see Figure 21. Interestingly, the majority of the hogvaxine consisted of peptides of the membrane protein, which was represented by only a single peptide in the OptiVax-Robust vaccine. In contrast, peptides of ORF1b accounted for the greatest part of the OptiVax-Robust vaccine.

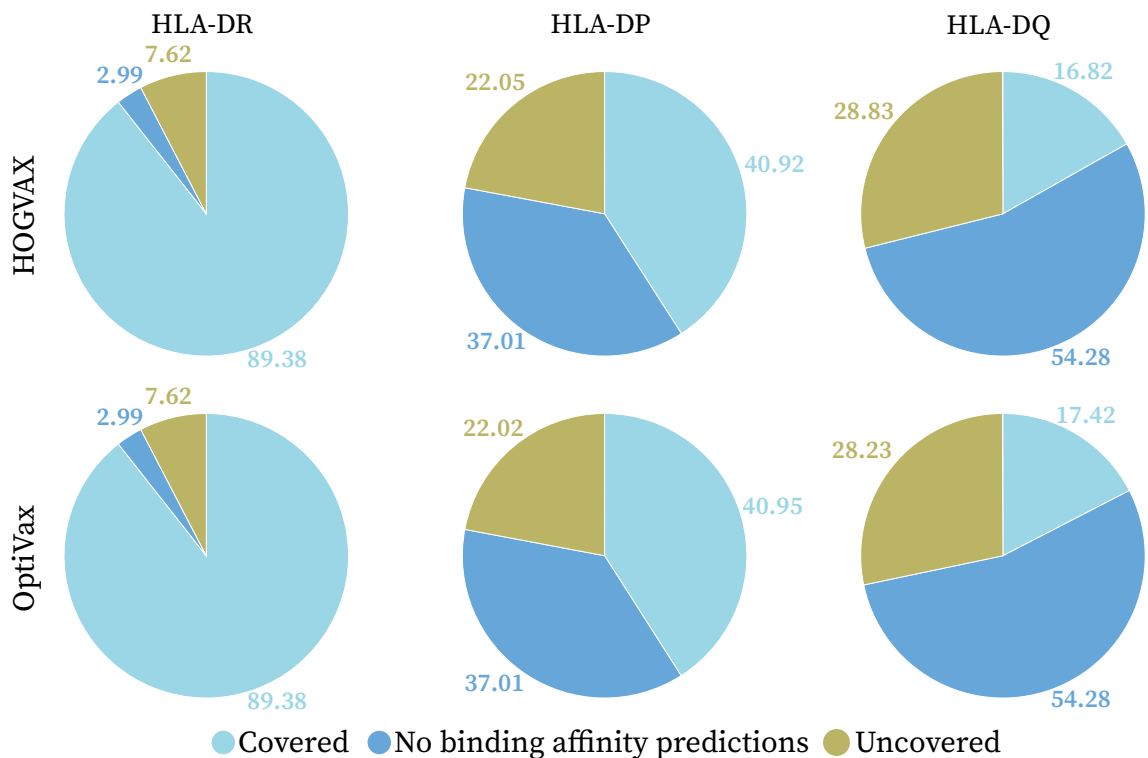
As presented in Figure 21 A, the OptiVax-Robust vaccine achieved 98.35 % coverage of an average world population with EvalVax-Unlinked, whereas the HOGVAX peptides were predicted with a slightly higher coverage of 99.38 %. HOGVAX again achieved promising results, when evaluating the vaccine based on the distribution of peptide-HLA hits per individual, see Figure 26 A, compared to OptiVax-Robust, see Figure 26 B. We observed a generally higher number of hits with a maximum of 302 observed hits compared to

the OptiVax-Robust-designed vaccine with a maximum observed hit count of 35. The expected peptide-HLA hits of the OptiVax-Robust peptides were 12.12, 12.43, and 7.87 in the European, African, and Asian population, respectively. In contrast, for the HOGVAX peptides, we calculated expected hits of 82.41, 91.44, and 37.47 for the European, African, and Asian population, respectively. What is conspicuous here are the low values for the Asian population of both methods. Compared to the African and European population, in our input data, there were fewer peptides predicted with less than 50 nM binding affinity for the MHC class II alleles in the Asian population. The evaluation of hits per population is given in Appendix B.2 Figure 38.

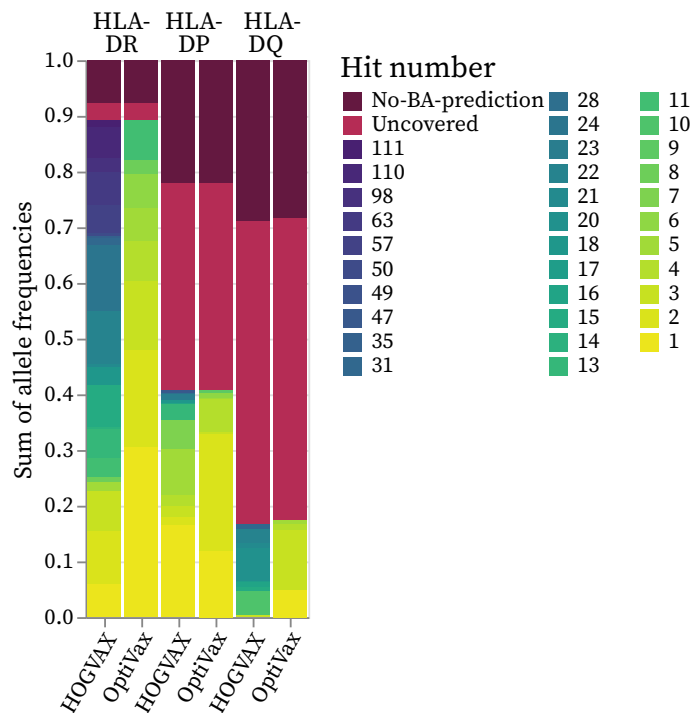
Just as with the expected peptide-HLA hits, HOGVAX outperformed OptiVax-Robust on the EvalVax-Robust metric. The results are shown in Figure 26 C and D for HOGVAX and OptiVax-Robust, respectively. Again, OptiVax-Robust showed a stronger decrease with an increasing number of minimum hits. EvalVax-Robust predicted a coverage of 99.975 % with at least one peptide-HLA hit, 91.888 % with at least five hits, and 56.142 % with at least ten hits for the OptiVax-Robust candidate over the three ethnic groups. As the expected numbers of the HOGVAX vaccine were already higher compared to the OptiVax-Robust peptide set, the hogvaxine preserved a higher average population coverage of 99.975 % with at least one peptide-HLA hit, 98.656 % with at least five hits, and 94.685 % with at least ten hits. What stood out here was the significantly lower population coverage of the Asian population with OptiVax-Robust, which had only 29.352 % EvalVax-Robust coverage with at least 10 hits.

When evaluating both methods using the HOGVAX metric, we observed a decrease in coverage by the HOGVAX vaccine set for loci HLA-DP and HLA-DQ. This is the first time, where OptiVax-Robust performed slightly better. For HOGVAX, we computed a coverage of 89.38 % of an average world population for locus HLA-DR, 40.92 % coverage for locus HLA-DP, and 16.82 % for locus HLA-DQ. The vaccine candidate of OptiVax-Robust achieved 89.38 %, 40.95 %, and 17.42 % for HLA-DR, -DP, and -DQ, respectively. The details are displayed in Figure 27. Furthermore, we calculated the fraction of individuals covered with a specific number of hits from each locus. Once again, we observed superior results for our method that induced significantly more hits per locus compared to the OptiVax-Robust vaccine, see Figure 28.

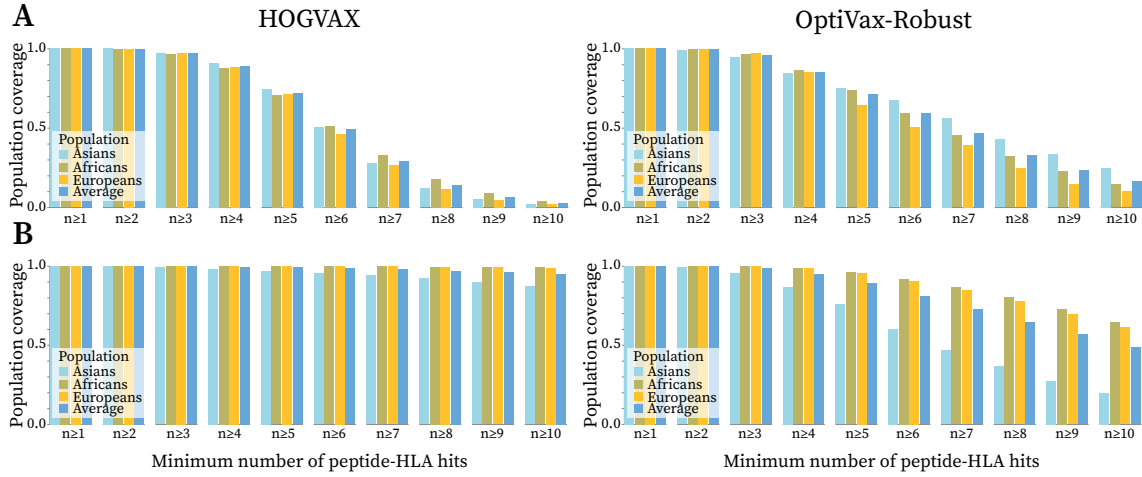
Despite that we could not run HOGVAX on the full genotype data, we wanted to compare OptiVax-Robust and HOGVAX using genotype frequencies. Thus, we designed a vaccine candidate with OptiVax-Robust that was optimized for the original genotype data of the African, Asian, and European population provided by Liu et al. [39]. For MHC class I, the chosen peptides resulted in a concatenated string of length 174, and for MHC



**Figure 27:** Coverage of MHC class II loci evaluated with the HOGVAX objective function. *Covered* is the proportion of individuals covered by at least one peptide at the respective locus. *No binding affinity predictions* gives the fraction of alleles for which NetMHCIIpan-4.0 could not predict any binding affinities. *Uncovered* gives the fraction of alleles without any peptide hit from the vaccine candidates.



**Figure 28:** Hits of MHC class II loci based on the HOGVAX evaluation metric. We counted the exact number of hits per allele and summed up the allele frequencies of those alleles with the same number of peptide-HLA hits induced by the vaccine. We also show the summed frequencies of uncovered alleles and alleles without available BA predictions.



**Figure 29:** EvalVax-Robust population coverage evaluation of the HOGVAX (first column) and the OptiVax-Robust (second column) vaccine candidates which were designed based on genotype data. Row **A** shows the results of the MHC class I candidates, row **B** the predictions for the MHC class II candidate.

class II, we obtained a value of 318. For HOGVAX to perform the calculations, we restricted the input data to the 100 000 genotypes with the highest average frequencies in the three populations. The MHC class I averaged genotype frequencies summed up to 72.70 % for the African population, 94.77 % for the Asian population, and 86.87 % for the European population. With the MHC class II averaged genotype frequencies, we covered 83.77 %, 94.88 %, and 94.48 % of the African, Asian, and European population, respectively. For the evaluation, we used all available haplotype combinations. Regarding the EvalVax-Robust predictions, we observed a sudden decrease in the MHC class I coverage achieved by HOGVAX for at least six peptide-HLA hits. HOGVAX was predicted with a coverage of 99.991 % with at least one peptide-HLA hit, 72.068 % with at least five hits, and 2.482 % with at least ten hits, where OptiVax-Robust achieved 99.997 %, 70.95 %, and 16.329 % population coverage with at least one, five, and ten hits, respectively, see Figure 29 A. Whereas, for MHC class II, the peptides chosen by HOGVAX achieved a higher EvalVax-Robust coverage compared to the OptiVax-Robust-designed vaccine set. The results are shown in Figure 29 B for MHC class II. We predicted 99.975 %, 98.817 %, and 94.875 % EvalVax-Robust coverage for HOGVAX with at least one, five, and ten hits, respectively. OptiVax-Robust, in contrast, achieved 99.975 %, 88.988 %, and 48.481 % for at least one, five, and ten hits, respectively. For HOGVAX, we observed a reduced number of expected peptide-HLA hits of 62.89, 65.84, and 28.87 for Europeans, Africans, and Asians, respectively. These and further results are given in Appendix B.2.

Except for the MHC class I genotype computations, HOGVAX was predicted with higher population coverage and expected number of hits compared to OptiVax-Robust, even

though that OptiVax-Robust optimizes the EvalVax-Robust metric. Here, the advantages of the superstring-vaccine design of HOGVAX became particularly clear and we were able to select significantly higher numbers of peptides for a maximal vaccine length equal to the summed lengths of the OptiVax-Robust-chosen peptides. In the following sections, we again focus on single-allele data for our experiments.

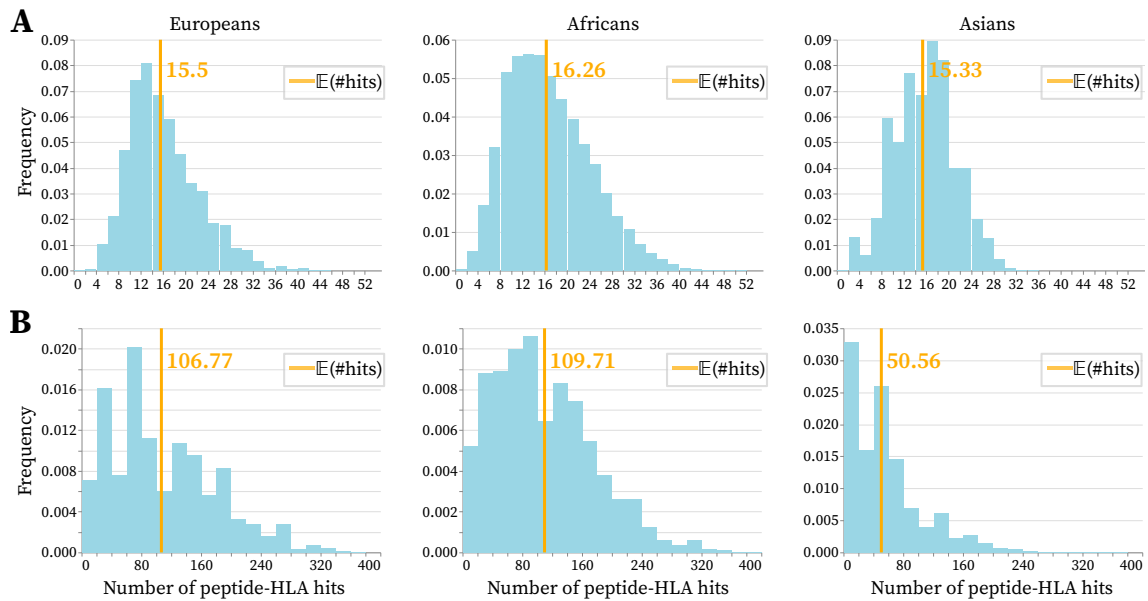
### 4.3 Designing Vaccines from the SARS-CoV-2 Spike Protein

Most of the currently available COVID-19 vaccines are based on the spike protein to induce immunity [29]. Furthermore, *in vivo* experiments showed that peptides from the S protein of SARS-CoV-2 activated T cell-mediated immune responses [35]. By filtering the input peptides for peptides with specific protein origins, HOGVAX can construct a vaccine that directs the immune response to these same proteins. OptiVax provides a similar option. We compared vaccine candidates of OptiVax-Unlinked and HOGVAX that were designed solely from peptides originating from the spike protein. For this, we used the input peptides from the comparison of HOGVAX and OptiVax-Unlinked in Section 4.2.1, giving us a set of 639 and 5 064 input peptides for MHC class I and class II, respectively. With OptiVax-Unlinked, we designed two vaccine candidates consisting of 19 peptides each for MHC class I and II. From the concatenations of the peptides, we obtained a limit of  $k = 176$  for MHC class I, and a limit of  $k = 322$  for MHC class II. The MHC class I hogvaxine included 149 peptides and achieved an EvalVax-Unlinked population coverage of 88.81 %. The OptiVax-Unlinked candidate was predicted with a coverage of 88.01 %. Thus, with the significantly higher number of peptides, we achieved higher population coverage. HOGVAX computed a set of 1180 peptides for MHC class II that obtained a population coverage of 96.35 % with EvalVax-Unlinked. Again, we observed a lower value of 93.00 % for OptiVax-Unlinked. For EvalVax-Robust, the results behaved similarly to the results presented in the previous sections. The vaccine computed by OptiVax-Unlinked showed a faster decrease of the EvalVax-Robust population coverage with an increasing number of minimal hits than the vaccine candidate of HOGVAX. We further observed that HOGVAX achieved higher expected numbers of peptide-HLA hits in the European, African, and Asian population for MHC class I and class II. Regarding our HOGVAX metric, we achieved the same locus-wise coverage with both methods on both MHC classes. The details are presented in Appendix B.3.

### 4.4 Combined Vaccine for MHC Class I and II

With HOGVAX, we can easily design a combined vaccine for MHC class I and II. Therefore, we had to adjust the input data, such that the binding affinity matrix and the frequency vector contained data for both MHC classes. We designed a combined vaccine

based on single-allele data. For this purpose, we used the filtered peptides of MHC class I and class II from OptiVax-Unlinked, which yielded a total input of 41 947 peptides. As the limit for our vaccine length, we used the summed limits of the separate vaccines, which gave us the parameter value  $k = 492$ . Thus, in theory, HOGVAX could create a vaccine equal to the concatenation of the separate class vaccines. However, HOGVAX chose a set of 1095 peptides, which are fewer than the summed number of peptides from the separate MHC class I and class II vaccines. Nevertheless, the combined vaccine achieved 98.93 % EvalVax-Unlinked coverage measured for MHC class I, and 99.11 % EvalVax-Unlinked coverage on MHC class II data. The population coverage for MHC class I was thereby slightly higher than that of the single MHC class I vaccine, while for MHC class II it was slightly lower. For the distribution of per-individual peptide-HLA hits, we observed a similar trend. The number of expected hits for MHC class I increased significantly to 15.5, 16.26, and 15.33 for Europeans, Africans, and Asians, respectively. For MHC class II, the number of expected hits decreased compared to the single MHC class II vaccine, and we predicted 106.77, 109.71, and 50.56 per-individual peptide-HLA hits for the European, African, and Asian population, respectively. This is shown in Figure 30 A and B. Similarly, the EvalVax-Robust predicted population coverage increased for MHC class I and decreased for MHC class II. We achieved 84.403 % population coverage with at least ten peptide-HLA hits per individual of an average population for MHC class I. For MHC class II, we observed 94.397 % coverage with at least ten hits in an average population, see Appendix B.4.



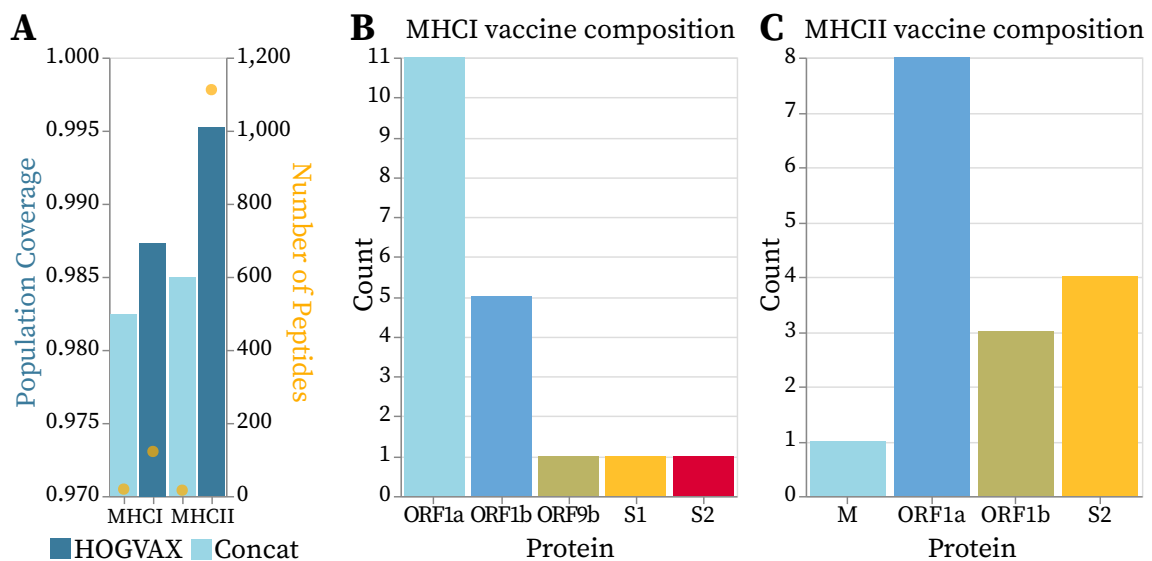
**Figure 30:** Evaluation of the combined MHC class I and II vaccine designed by HOGVAX. Distribution of per-individual peptide-HLA hits and expected hits for three populations self-reporting as having European, African, or Asian ancestry measured for **A** MHC class I and **B** MHC class II separately.

#### 4.5 Increased Population Coverage by Leveraging Overlaps

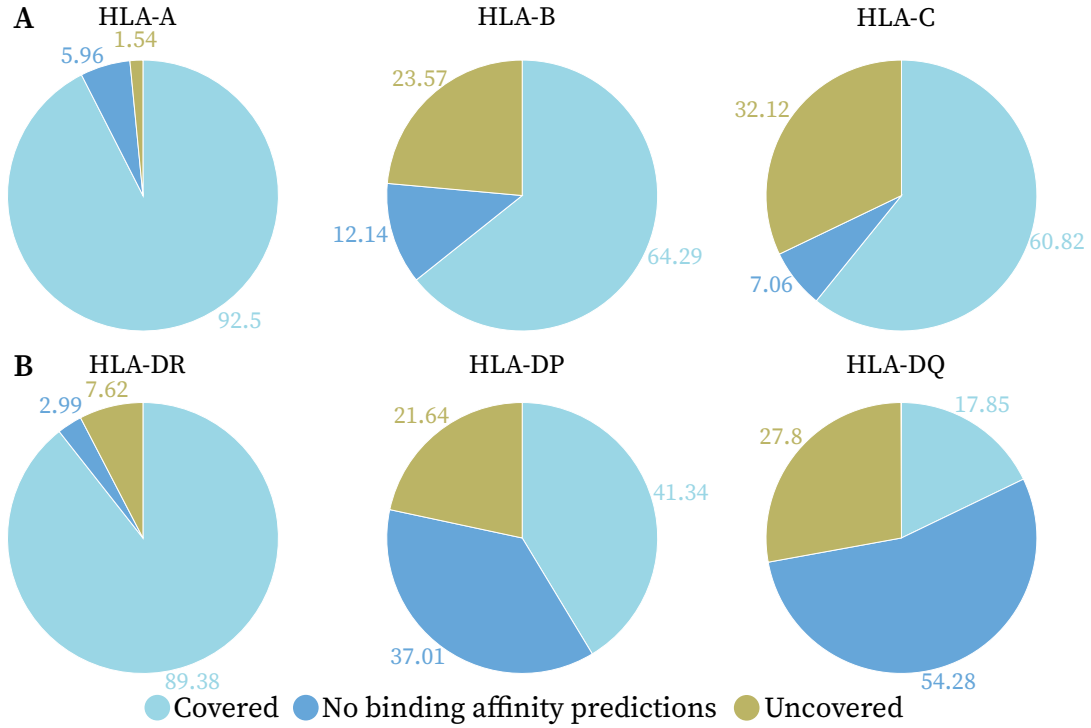
We also compared the vaccine efficacy of our HOGVAX-designed vaccines for MHC class I and II with the outcomes of the concatenation approach described in Section 4.1. Both models optimize the same objective function. For the computations presented here, we used the same data as for our comparison to OptiVax-Unlinked. For MHC class I, the concatenation method chose 19 peptides, and for MHC class II 16 peptides, as shown in Figure 31 A. It is no surprise that the composition of the MHC class I vaccine is very similar to the OptiVax-Unlinked MHC class I candidate given that we concatenated the OptiVax-Unlinked peptides to create our limit of  $k = 170$ . The MHC class II concatenated vaccine also consisted of peptides from similar origins as the OptiVax-Unlinked peptides, though the concatenation method did not choose peptides from the S1 unit of the spike protein. The compositions are presented in Figure 31 B and C.

Based on the EvalVax-Unlinked metric, the concatenation approach achieved lower population coverage than HOGVAX. For MHC class I, EvalVax-Unlinked predicted a coverage of 98.24 % for the average world population, and for MHC class II, the predicted coverage was 98.49 %. Interestingly, the latter coverage was slightly higher than that predicted for the OptiVax-Unlinked MHC class II candidate, which was optimized based on the EvalVax-Unlinked metric.

With the concatenation vaccine for MHC class I, we did not achieve the same locus-wise coverage as with the HOGVAX MHC class I candidate. This is what we expected for a smaller number of chosen peptides. However, the fraction of covered individuals per locus was slightly higher compared to the OptiVax-Unlinked candidate. For MHC



**Figure 31:** A Comparison of EvalVax-Unlinked population coverage predictions for HOGVAX and the concatenation approach. Compositions of the concatenation **B** MHC class I and **C** MHC class II vaccine candidates.



**Figure 32:** Coverage of **A** MHC class I and **B** MHC class II concatenation vaccine candidate evaluated with HOGVAX objective function. *Covered* gives the proportion of individuals covered by at least one peptide at the respective locus. *No binding affinity predictions* gives the fraction of alleles for which NetMHCIIpan-4.1 and NetMHCIIpan-4.0 could not predict any binding affinities. *Uncovered* is the proportion of alleles without any peptide hit from the concatenation vaccine.

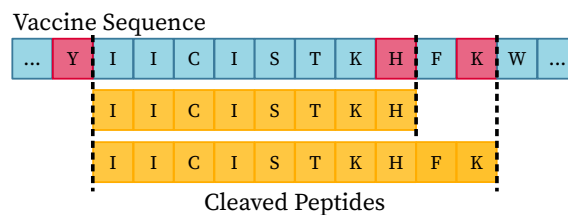
class II, the peptides chosen by the concatenation ILP covered each locus equally good as HOGVAX and OptiVax-Unlinked. The results are given in Figure 32.

The improvements by HOGVAX were particularly evident when we measured the number of per-individual peptide-HLA hits. For the concatenated MHC class I vaccine, we obtained 6.65, 6.93, and 7.69 expected hits per individual of the European, African, and Asian population, respectively, which is lower by at least 3.21 hits compared to the MHC class I HOGVAX candidate. For MHC class II, the difference in expected hits became even larger between HOGVAX and the concatenation approach. The latter achieved an expected number of hits of 13.18, 12.98, and 8.5 for the European, African, and Asian population, respectively, which is on average lower by 151.15 hits than the expected values for the MHC class II hogvaxine. However, these MHC class II results of the concatenation method are significantly better compared to the MHC class II expected hits achieved by OptiVax-Unlinked. These results are presented in Appendix B.5.

#### 4.6 Regional Context Embedding

Our method relies greatly on the assumption that the peptides are correctly cut from the generated superstring vaccine. However, this depends in particular on the cutting pro-





**Figure 33:** Example of cleavage sites (red) on a vaccine sequence (blue) with corresponding cleaved peptides (yellow). The dashed lines represent the cut. A peptide is cut at the C-terminus of the amino acid, thus the cleavage site is included in the cut fragment.

cess of the proteasome. So far, little is known about the exact cleavage process, though it is anticipated that preferred cleavage motives exist [60]. To be able to make predictions nevertheless, neural networks such as NetChop-3.1 [43] and pepsickle [60] were developed. NetChop-3.1 is a state-of-the-art prediction tool trained on data from *in vitro* digestion experiments with the human proteasome and experimentally analyzed MHC class I ligands [43]. Pepsickle is a recent tool by Weeder et al. [60] from 2021 that was trained on *in vitro* protein digestion and *in vivo* MHC class I presented ligand data from various databases, and has similar performance to NetChop-3.1.

For the proteasomal cleavage experiments, we used single-allele data and removed self-peptides, peptides of cleavage regions, and peptides with non-zero glycosylation probability from our input peptides. Since our vaccine approach targets the cross-presentation pathway, we consider both MHC class I and MHC class II vaccine candidates here. With HOGVAX, we created vaccine sequences for MHC class I and II with maximal lengths of  $k = 170$  and  $k = 322$ , respectively. Additionally, we computed the concatenations of the peptides chosen by HOGVAX.

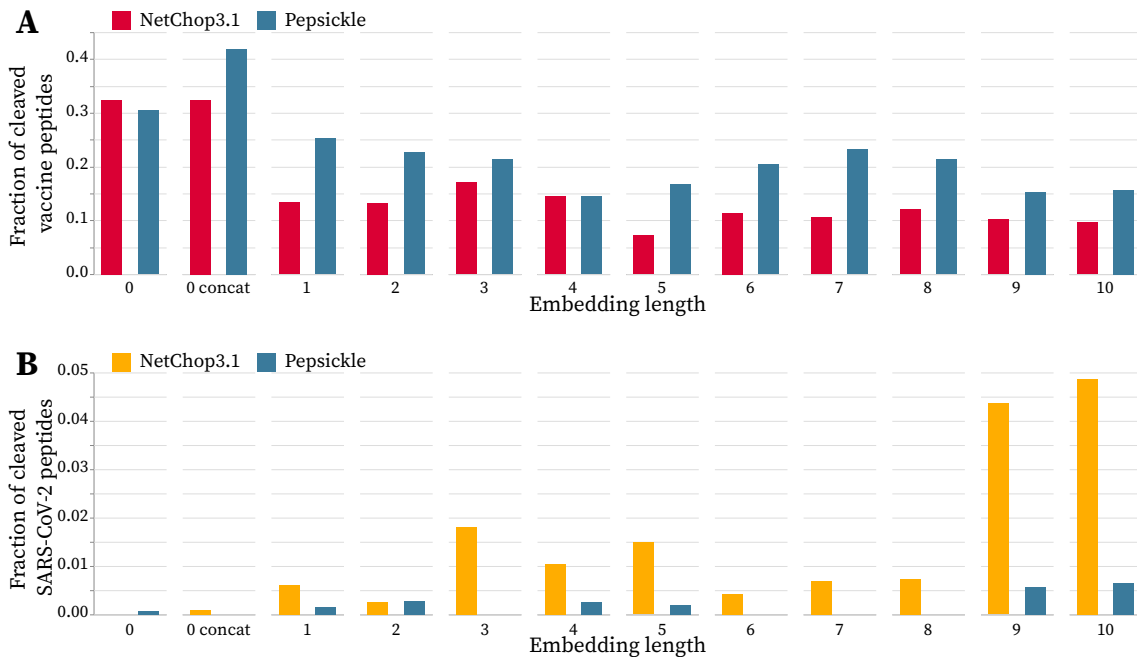
We wondered if embedding the peptides in their regional context, which means adding additional flanking amino acids to either side of a peptide sequence from their origin, might lead to a better cleavage prediction. For us, this means that more of the peptides selected by HOGVAX are properly cut from the vaccine sequence. We considered embedding lengths of one to ten amino acids that were added to the left and the right of each input peptide. In this process, we excluded peptides from the start and end of the SARS-CoV-2 proteins for which we could not add the requested amount of flanking sequences to either side to prevent duplicates in our input data.

We used NetChop-3.1 and pepsickle to get an impression of how the HOGVAX-designed vaccines are processed in the cell by the proteasome. For each position  $i$  of the respective vaccine sequence, NetChop-3.1 and pepsickle computed a probability that the sequence is cut directly after position  $i$ . We considered each peptide from one cutting position to all further cutting positions within the range of range 8 to 10 for MHC class I and of range 13 to 25 for MHC class II. For clarification see Figure 33.

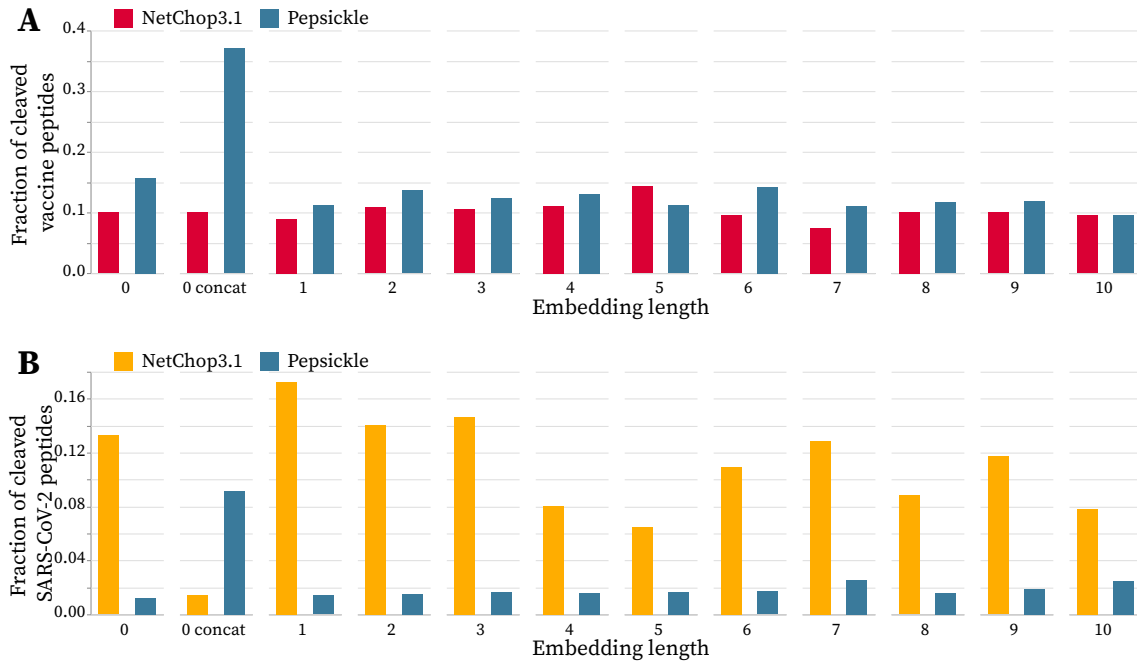
For each instance of MHC class I, we calculated the fraction of chosen peptides that occurred in the cleavage prediction output. The results are presented in Figure 34 A, where it becomes clear that the vaccines created from non-embedded peptides performed best. Interestingly, pepsickle always predicted more chosen peptides to be cut from a respective vaccine sequence than NetChop-3.1, except for our non-embedded hogvaxine. However, the frequencies of correctly cut peptides were rather low, where the concatenation sequence achieved the maximum value of slightly over 40 %.

Furthermore, we compared the peptides obtained from the NetChop-3.1 and pepsickle predictions with peptides that were predicted to be cut from the SARS-CoV-2 proteome. For each instance, we computed the fraction of peptides that matched the cleaved virus peptides. The results for MHC class I are shown in Figure 34 B. In contrast to our first evaluation, we observed that the two largest embedding lengths were predicted with NetChop-3.1 to have the highest fractions of matching peptides. However, these predictions were within a range of 4 % to 5 %. All other predictions, especially those calculated with pepsickle, resulted in a significantly lower proportion of peptides cut in the same way. In most cases, we did not even observe a single equally cleaved peptide predicted by either method.

For MHC class II, the cleavage results are given in Figure 35. The probability for a chosen peptide to be correctly cut by the proteasome is generally lower than for the MHC class I peptides, see Figure 35 A. Again, the vaccine sequence of concatenated peptides achieved the maximum value of about 37 % predicted by pepsickle. In contrast, we observed generally larger fractions of cleaved peptides from the MHC class II vaccine that were equally cut from the SARS-CoV-2 proteome, than for the MHC class I candidates. The results are shown in Figure 35 B. Here, NetChop-3.1 predicted higher numbers of matching peptides than pepsickle, with a maximum value of approximately 17 % for the concatenated sequence. However, the concatenated sequence is longer than all the other sequences and thus also has more cleavage positions. Interestingly, with NetChop-3.1 we did not observe any beneficial behavior of the concatenation vaccine. Actually, for the SARS-CoV-2 cleavage predictions, it was predicted to have the lowest of all NetChop-3.1 MHC class II scores.



**Figure 34:** Proteasomal cleavage predictions for MHC class I candidates. **A** Proportion of peptides we added to the vaccine that was predicted to be cut by the proteasome. **B** Proportion of peptides we added to the vaccine that is equivalently cut as the peptides from the SARS-CoV-2 proteome.



**Figure 35:** Proteasomal cleavage predictions for MHC class II candidates. **A** Proportion of peptides we added to the vaccine that was predicted to be cut by the proteasome. **B** Proportion of peptides we added to the vaccine that is equivalently cut as the peptides from the SARS-CoV-2 proteome.

## 5 Discussion and Conclusions

The immunogenicity of a vaccine targeting the cell-mediated immune response essentially relies on the distribution of HLA alleles in the target population. T cells can only recognize antigens that are presented by MHC molecules. Since the HLA complex is the most polymorphic region in human DNA [47], vaccine design is a challenging task in a genetically heterogeneous population. In the current COVID-19 pandemic, with over 6.5 million deaths worldwide, it is more important than ever to develop a vaccine that immunizes as many individuals as possible. Thus, a vaccine must also target HLA alleles occurring at a low frequency.

In this study, we present HOGVAX, a novel combinatorial approach for MHC class I and II peptide vaccine design that solves the *Maximal Scoring  $k$ -Superstring* problem. As the underlying data structure to select an optimal set of peptides, we use the hierarchical overlap graph. With HOGVAX, we intend to fill the gap of uncovered rare alleles, maximize population coverage based on allele frequencies of the target population, and induce robust immunity. Due to our overlap vaccine design, we can fit a significantly larger amount of peptides within the size of a regular vaccine constructed from concatenated peptides. In this work, we compare HOGVAX to OptiVax, a combinatorial machine learning approach invented by Liu et al. [39]. Even though the MSKS problem is NP-hard, by solving our ILP, we create a provably optimal vaccine candidate. In contrast, OptiVax is based on a heuristic search that does not necessarily return the optimal peptide vaccine set.

Note that OptiVax chooses a specific number of peptides and Liu et al. did not assume any cutting of the peptides. Our sequence, however, must be cut into peptides. Therefore, we have more peptides in our vaccine than in the OptiVax vaccine. As reflected by our results, the larger number of peptides chosen by HOGVAX generally leads to higher predicted population coverage and per-individual peptide-HLA hits compared to the OptiVax-designed vaccines. To not waste any space, Gurobi likely chooses peptides that cover multiple loci or haplotypes at once. This, and the high number of peptides would explain the tremendous increase of expected hits. We assume that a higher number of peptide-HLA hits per genotype correlates with higher efficacy of the vaccine [39], for example by inducing immunity against various parts of the pathogen. Moreover, a large set of peptides leads to a more conserved immunity to viral mutations, as the probability of simultaneous mutations in the viral peptides exponentially decreases with an increasing number of considered peptides [57].

However, we often observe a lower number of expected hits for MHC class II of the Asian population for both methods. There are generally fewer peptides in our input pre-

dicted to bind to diverse MHC class II alleles of the Asian population. It remains unclear, if this is due to the input alone, or if there is a systematic error in predictions of NetMHCIIpan-4.0 for alleles of the Asian population. Further, regarding the NetMHCIIpan-4.0 predictions, the lack of binding affinity predictions for HLA-DP and -DQ is a reason to explore additional prediction tools in the future.

In our haplotype computations, we observed a slight decrease in locus-wise coverage. This is because our method does not optimize for single alleles at this point, but for haplotype frequencies of three distinct populations. As a haplotype is covered, if at least one peptide binds to one of the three alleles, the population coverage scales with the maximally covered loci. Regarding the genotype computations, the performance of HOGVAX for MHC class I is rather poor. This is likely due to the reduced input data which was necessary to get the ILP running. Here, our method could be improved further, for example by preprocessing the genotype data by merging genotypes that are covered by the same set of peptides. If we do so, we would sum the corresponding frequencies together and create a smaller set of  $h_a$  variables, which also results in fewer constraints.

In contrast, for MHC class II, HOGVAX achieved higher EvalVax-Robust coverage compared to OptiVax-Robust, even though it used about 10 % of the genotype data. This is presumably because HOGVAX generally achieved higher peptide-HLA hit numbers for MHC class II. Apart from that, the 10 % still covered more than 90 % of the average population for which HOGVAX could optimize the vaccine peptides.

The diverse compositions of the HOGVAX and OptiVax vaccines are worth mentioning. HOGVAX often chooses more peptides from the ORF1a and the membrane protein compared to OptiVax. Grifoni et al. showed that the largest proportion of peptides recognized by CD4<sup>+</sup> and CD8<sup>+</sup> T cells in individuals unexposed to SARS-CoV-2 derived from ORF1a and ORF1b [20]. The second largest fraction of recognized peptides originated from the spike protein, followed by peptides from the membrane protein [20]. However, T cells from individuals infected with COVID-19 mostly targeted the spike protein, followed by similar fractions of the peptides from the membrane protein and ORF1a and b [20]. It seems like, the vaccine compositions model the T cell activation of unexposed individuals. From the SARS-CoV outbreak in 2002, we know that we need to vaccinate against the spike protein to prevent infections [14]. However, the amount of S1 and S2 peptides in the vaccine candidates computed with HOGVAX and OptiVax is rather low, which is likely the cause of the specific predictions of NetMHCpan-4.1 and NetMHCIIpan-4.0. Nevertheless, by targeting further proteins, a vaccine can be more robust against viral mutations that otherwise likely lead to immune escapes as in the case of the SARS-CoV-2 Omicron variant and the available spike protein vaccines [61].

Combining peptides for MHC class I and II in a single vaccine likely increases the acti-

vation of CD8<sup>+</sup> T cells and thereby leads to an improved immunity [37]. In our combined vaccine, the decrease of the peptide-HLA hits for MHC class II was rather unexpected, although this is not the focus of our objective function. We chose our limit of  $k$  in such a way that HOGVAX could theoretically have selected the same peptides as for the individual vaccine designs. However, the number of chosen peptides was lower compared to the separate designs if the candidate sets were taken together. Here, we realized that multiple solutions exist that maximize our objective function. Furthermore, when recomputing the results for the same input, the output changed although Gurobi is deterministic. While analyzing this issue, we observed non-deterministic behavior by NetworkX. Thus, when computing the HOG twice for the same input peptides, the two graph objects were slightly different. For example, NetworkX lists the successors of each node in random order. Though the changes were minimal, Gurobi changed its optimization process. Unfortunately, neither setting seeds, nor sorting the nodes and edges had an effect. Thus, the task of fixing the randomness is part of our future work. Note, that the changes of the vaccine peptides did not affect the superior performance of HOGVAX over OptiVax. Some of the optimal solutions might also be permutations of other solutions, for example, if the same edges are traversed but in a different order. This is known as *symmetry* and might have a negative effect on the branch-and-bound approach of Gurobi [41]. By adding further constraints it is possible to remove symmetric solutions. We will investigate such strategies in the future to eventually increase the performance of HOGVAX.

So far, little is known about the protein degradation by the proteasome, such that there is uncertainty if the peptides of our overlap-vaccine sequence will be present in the end. The predictions with NetChop-3.1 and pepsickle were not promising, but machine learning prediction tools are limited in their ability to model real world behavior, especially when the underlying mechanism is poorly understood [17, 43]. Thus, the cleavage predictions should be taken with a grain of salt. Furthermore, the presentation pathway of MHC class II usually does not depend on the protein degradation of the proteasome, rather than on the endocytic pathway which is also not fully understood [21]. However, evidence was found that MHC class II molecules can present endogenous antigens derived from the MHC class I presentation pathway [36, 38]. It should be noted, that NetChop-3.1 and pepsickle were only trained on data for MHC class I, so the predictions for MHC class II are even less meaningful. Thus, only experimental investigations will show if our approach stands a chance against competing methods, or if it will fail due to unpredictable proteasomal cleavage.

Further limitations of our method derive from the use of binding affinity prediction tools. As for neural networks in general, we can argue that predictions might not be completely reliable. For SARS-CoV-2, however, thanks to the intensive research in the last

two years, binding predictions might be validated from publicly available experimental results.

The fact that multiple optimal solutions exist, offers new opportunities to further improve our method. For example, we could maximize the number of peptides in the output set in a second optimization, where we apply the already optimized population coverage as a constraint. Another possibility would be to also optimize the cleavage probability of peptides. We could also include peptides that were shown to be good vaccine candidates by setting the corresponding variables to 1. Then we would build a vaccine sequence around those pre-selected peptides. Furthermore, HOGVAX is not restricted to the application to SARS-CoV-2, but could, for example, be used to design cancer vaccines. Peptide vaccines are considered a promising approach in oncology to prevent cancerous diseases [40].

To our knowledge, our combinatorial approach is the first method that designs peptide vaccines by exploiting overlaps between the peptide sequences. Thereby, we significantly increase the number of peptides included in a vaccine and conquer the challenge of high HLA diversity. With our vaccine formulations, we maximize population coverage and induce robust immunity against SARS-CoV-2. Our approach is flexible and can easily be adjusted for other purposes, like vaccine design against novel SARS-CoV-2 variants and other diseases like cancer.

## 6 Code

We provide the Snakemake pipeline we used for our computations, and necessary input files, as well as results, computed vaccine sequences, preprocessing scripts, and evaluation scripts on gitlab <https://gitlab.cs.uni-duesseldorf.de/schulte/hogvax>. Additionally, the repository includes HOGVAX as stand-alone program.





## References

- [1] A. V. Aho and M. J. Corasick. “Efficient String Matching: An Aid to Bibliographic Search.” In: **Communications of the ACM** 18.6 (1975), pp. 333–340. URL: <https://dl.acm.org/doi/abs/10.1145/360825.360855>.
- [2] B. Alvarez et al. “NNAlign\_MA; MHC Peptidome Deconvolution for Accurate MHC Binding Motif Characterization and Improved T-cell Epitope Predictions.” In: **Molecular & Cellular Proteomics** 18.12 (2019), pp. 2459–2477. URL: [https://www.mcponline.org/article/S1535-9476\(20\)31649-2/fulltext](https://www.mcponline.org/article/S1535-9476(20)31649-2/fulltext).
- [3] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis. “Linear Approximation of Shortest Superstrings.” In: **Journal of the ACM (JACM)** 41.4 (1994), pp. 630–647. URL: <https://dl.acm.org/doi/abs/10.1145/179812.179818>.
- [4] F. Bökler, M. Chimani, M. H. Wagner, and T. Wiedera. “An Experimental Study of ILP Formulations for the Longest Induced Path Problem.” In: **Combinatorial Optimization**. Ed. by M. Baïou, B. Gendron, O. Günlük, and A. R. Mahjoub. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 89–101. ISBN: 978-3-030-53262-8.
- [5] H.-H. Bui et al. “Predicting Population Coverage of T-cell Epitope-Based Diagnostics and Vaccines.” In: **BMC bioinformatics** 7.1 (2006), pp. 1–5. URL: <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-7-153>.
- [6] G. Cafri et al. “mRNA Vaccine-Induced Neoantigen-Specific T Cell Immunity in Patients with Gastrointestinal Cancer.” In: **The Journal of clinical investigation** 130.11 (2020), pp. 5976–5988. URL: <https://www.jci.org/articles/view/134915>.
- [7] B. Cazaux, R. Cánovas, and E. Rivals. “Shortest DNA Cyclic Cover in Compressed Space.” In: **2016 Data Compression Conference (DCC)**. IEEE. 2016, pp. 536–545. URL: <https://ieeexplore.ieee.org/abstract/document/7786198>.
- [8] B. Cazaux and E. Rivals. “A Linear Time Algorithm for Shortest Cyclic Cover of Strings.” In: **Journal of Discrete Algorithms**. 2015 London Stringology Days and London Algorithmic Workshop (LSD & LAW) 37 (Mar. 2016), pp. 56–67. ISSN: 1570-8667. URL: <https://www.sciencedirect.com/science/article/pii/S1570866716300065> (visited on 09/04/2022).
- [9] B. Cazaux and E. Rivals. “Hierarchical Overlap Graph.” In: **Information Processing Letters** 155 (2020), p. 105862. URL: <https://www.sciencedirect.com/science/article/abs/pii/S0020019019301450>.
- [10] D. D. Chaplin. “Overview of the Immune Response.” In: **The Journal of allergy and clinical immunology** 125.2 Suppl 2 (Feb. 2010), S3–23. ISSN: 0091-6749. URL: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2923430/> (visited on 09/30/2022).

- [11] P. Compeau and P. Pevzner. **Bioinformatics Algorithms: An Active Learning Approach**. Second. Vol. 2. Active Learning Publishers La Jolla, California, 2015. Chap. 9, pp. 123–124.
- [12] U. Consortium. “UniProt: A Worldwide Hub of Protein Knowledge.” In: **Nucleic acids research** 47.D1 (2019), pp. D506–D515. URL: <https://academic.oup.com/nar/article/47/D1/D506/5160987>.
- [13] B. Coutard et al. “The Spike Glycoprotein of the New Coronavirus 2019-nCoV Contains a Furin-like Cleavage Site Absent in CoV of the Same Clade.” In: **Antiviral Research** 176 (2020), p. 104742. ISSN: 0166-3542. URL: <https://www.sciencedirect.com/science/article/pii/S0166354220300528>.
- [14] L. Du et al. “The Spike Protein of SARS-CoV — a Target for Vaccine and Therapeutic Development.” In: **Nature Reviews Microbiology** 7.3 (Mar. 2009), pp. 226–236. ISSN: 1740-1534. URL: <https://www.nature.com/articles/nrmicro2090> (visited on 09/29/2022).
- [15] S. Elbe and G. Buckland-Merrett. “Data, Disease and Diplomacy: GISAID’s Innovative Contribution to Global Health.” In: **Global Challenges** 1.1 (2017), pp. 33–46. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/gch2.1018>.
- [16] M. Englert, N. Matsakis, and P. Vesely. “Improved Approximation Guarantees for Shortest Superstrings Using Cycle Classification by Overlap to Length Ratios.” In: **Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing**. 2022, pp. 317–330. URL: <https://dl.acm.org/doi/abs/10.1145/3519935.3520001>.
- [17] P. Fort, A. V. Kajava, F. Delsuc, and O. Coux. “Evolution of Proteasome Regulators in Eukaryotes.” In: **Genome Biology and Evolution** 7.5 (May 2015), pp. 1363–1379. ISSN: 1759-6653. URL: <https://doi.org/10.1093/gbe/evw068> (visited on 09/29/2022).
- [18] J. Gallant, D. Maier, and J. Astorer. “On Finding Minimal Length Superstrings.” In: **Journal of Computer and System Sciences** 20.1 (1980), pp. 50–58. ISSN: 0022-0000. URL: <https://www.sciencedirect.com/science/article/pii/0022000080900045>.
- [19] T. Gao et al. “Highly Pathogenic Coronavirus N Protein Aggravates Lung Injury by MASP-2-mediated Complement over-Activation.” In: **medRxiv : the preprint server for health sciences** (2020).
- [20] A. Grifoni et al. “Targets of T Cell Responses to SARS-CoV-2 Coronavirus in Humans with COVID-19 Disease and Unexposed Individuals.” In: **Cell** 181.7 (June 2020), 1489–1501.e15. ISSN: 0092-8674. URL: <https://www.sciencedirect.com/science/article/pii/S092867420306103> (visited on 09/29/2022).
- [21] J. Gruenberg. “The Endocytic Pathway: A Mosaic of Domains.” In: **Nature Reviews Molecular Cell Biology** 2.10 (Oct. 2001), pp. 721–730. ISSN: 1471-0080. URL: <https://www.nature.com/articles/35096054> (visited on 09/29/2022).

- [22] R Gupta, E Jung, and S. Brunak. “Prediction of N-glycosylation Sites in Human Proteins.” In: (2004). URL: <http://www.cbs.dtu.dk/services/NetNGlyc/>.
- [23] Gurobi. **Mixed-Integer Programming (MIP) - A Primer on the Basics**. URL: <https://www.gurobi.com/resource/mip-basics/> (visited on 09/08/2022).
- [24] **Gurobi - The Fastest Solver - Gurobi**. 2022. URL: <https://www.gurobi.com/> (visited on 09/10/2022).
- [25] D. Gusfield. **Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology**. Cambridge: Cambridge University Press, 1997.
- [26] D. Gusfield, G. M. Landau, and B. Schieber. “An Efficient Algorithm for the All Pairs Suffix-Prefix Problem.” In: **Information Processing Letters** 41.4 (1992), pp. 181–185. ISSN: 0020-0190. URL: <https://www.sciencedirect.com/science/article/pii/002001909290176V>.
- [27] J. Hadfield et al. “Nextstrain: Real-Time Tracking of Pathogen Evolution.” In: **Bioinformatics (Oxford, England)** 34.23 (May 2018), pp. 4121–4123. ISSN: 1367-4803. URL: <https://doi.org/10.1093/bioinformatics/bty407>.
- [28] W. Helmberg, R. Dunivin, and M. Feolo. “The Sequencing-Based Typing Tool of dbMHC: Typing Highly Polymorphic Gene Sequences.” In: **Nucleic acids research** 32.suppl\_2 (2004), W173–W175. URL: [https://academic.oup.com/nar/article/32/suppl\\_2/W173/1040628](https://academic.oup.com/nar/article/32/suppl_2/W173/1040628).
- [29] **Herstellung von Covid-19-Impfstoffen**. URL: <https://www.vfa.de/de/arzneimittel-forschung/coronavirus/corona-impfstoffe-herstellung> (visited on 09/26/2022).
- [30] C. Janeway, P. Travers, M. Walport, and M. Shlomchik. **Immunobiology: The Immune System in Health and Disease**. Fifth. New York: Garland Science, 2001. URL: <https://www.ncbi.nlm.nih.gov/books/NBK27098/>.
- [31] R. Kannan and C. L. Monma. “On the Computational Complexity of Integer Programming Problems.” In: **Optimization and Operations Research**. Ed. by R. Henn, B. Korte, and W. Oettli. Lecture Notes in Economics and Mathematical Systems. Berlin, Heidelberg: Springer, 1978, pp. 161–172. ISBN: 978-3-642-95322-4.
- [32] H. Kaplan and N. Shafrir. “The Greedy Algorithm for Shortest Superstrings.” In: **Information Processing Letters** 93.1 (2005), pp. 13–17. ISSN: 0020-0190. URL: <https://www.sciencedirect.com/science/article/pii/S0020019004002698>.
- [33] R. M. Karp. “Reducibility among Combinatorial Problems.” In: **Complexity of Computer Computations**. Ed. by R. E. Miller, J. W. Thatcher, and J. D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. URL: [https://doi.org/10.1007/978-1-4684-2001-2\\_9](https://doi.org/10.1007/978-1-4684-2001-2_9).

- [34] D. E. Knuth, J. H. Morris Jr, and V. R. Pratt. “Fast Pattern Matching in Strings.” In: **SIAM journal on computing** 6.2 (1977), pp. 323–350. URL: <https://epubs.siam.org/doi/abs/10.1137/0206024>.
- [35] Y.-S. Lee et al. “Peptides Derived From S and N Proteins of Severe Acute Respiratory Syndrome Coronavirus 2 Induce T Cell Responses: A Proof of Concept for T Cell Vaccines.” In: **Frontiers in Microbiology** 12 (2021). ISSN: 1664-302X. URL: <https://www.frontiersin.org/articles/10.3389/fmicb.2021.732450> (visited on 09/26/2022).
- [36] C. S. K. Leung. “Endogenous Antigen Presentation of MHC Class II Epitopes through Non-Autophagic Pathways.” In: **Frontiers in Immunology** 6 (2015). ISSN: 1664-3224. URL: <https://www.frontiersin.org/articles/10.3389/fimmu.2015.00464> (visited on 09/24/2022).
- [37] W. Li, M. D. Joshi, S. Singhanian, K. H. Ramsey, and A. K. Murthy. “Peptide Vaccine: Progress and Challenges.” In: **Vaccines (cold Spring Harbor Laboratory Press)** 2.3 (2014), pp. 515–536. URL: <https://www.mdpi.com/2076-393X/2/3/515>.
- [38] J. D. Lich, J. F. Elliott, and J. S. Blum. “Cytoplasmic Processing Is a Prerequisite for Presentation of an Endogenous Antigen by Major Histocompatibility Complex Class II Proteins.” In: **Journal of Experimental Medicine** 191.9 (May 2000), pp. 1513–1524. ISSN: 0022-1007. URL: <https://doi.org/10.1084/jem.191.9.1513> (visited on 09/24/2022).
- [39] G. Liu et al. “Computationally Optimized SARS-CoV-2 MHC Class I and II Vaccine Formulations Predicted to Target Human Haplotype Distributions.” In: **Cell systems** 11.2 (2020), pp. 131–144. URL: <https://www.sciencedirect.com/science/article/pii/S2405471220302386>.
- [40] R. J. Malonis, J. R. Lai, and O. Vergnolle. “Peptide-Based Vaccines: Current Progress and Future Challenges.” In: **Chemical reviews** 120.6 (2019), pp. 3210–3229. URL: <https://pubs.acs.org/doi/full/10.1021/acs.chemrev.9b00472>.
- [41] F. Margot. “Symmetry in Integer Linear Programming.” In: **50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art**. Ed. by M. Jünger et al. Berlin, Heidelberg: Springer, 2010, pp. 647–686. ISBN: 978-3-540-68279-0. URL: [https://doi.org/10.1007/978-3-540-68279-0\\_17](https://doi.org/10.1007/978-3-540-68279-0_17) (visited on 10/02/2022).
- [42] F. Mölder et al. **Sustainable Data Analysis with Snakemake**. Apr. 2021. URL: <https://f1000research.com/articles/10-33> (visited on 09/10/2022).
- [43] M. Nielsen, C. Lundegaard, O. Lund, and C. Keşmir. “The Role of the Proteasome in Generating Cytotoxic T-cell Epitopes: Insights Obtained from Improved Predictions of Proteasomal Cleavage.” In: **Immunogenetics** 57.1 (Apr. 2005), pp. 33–41. ISSN: 1432-1211. URL: <https://doi.org/10.1007/s00251-005-0781-7> (visited on 09/24/2022).

- [44] P. Oyarzun and B. Kobe. “Computer-Aided Design of T-cell Epitope-Based Vaccines: Addressing Population Coverage.” In: **International journal of immunogenetics** 42.5 (2015), pp. 313–321. URL: <https://onlinelibrary.wiley.com/doi/full/10.1111/iji.12214>.
- [45] S. Park, S. G. Park, B. Cazaux, K. Park, and E. Rivals. “A Linear Time Algorithm for Constructing Hierarchical Overlap Graphs.” In: **arXiv preprint arXiv:2102.12824** (2021). URL: <https://arxiv.org/abs/2102.12824>.
- [46] B. Reynisson, B. Alvarez, S. Paul, B. Peters, and M. Nielsen. “NetMHCpan-4.1 and NetMHCIIpan-4.0: Improved Predictions of MHC Antigen Presentation by Concurrent Motif Deconvolution and Integration of MS MHC Eluted Ligand Data.” In: **Nucleic Acids Research** 48.W1 (May 2020), W449–W454. ISSN: 0305-1048. URL: <https://academic.oup.com/nar/article/48/W1/W449/5837056>.
- [47] J. Robinson, M. J. Waller, P. Parham, J. G. Bodmer, and S. G. E. Marsh. “IMGT/HLA Database—a Sequence Database for the Human Major Histocompatibility Complex.” In: **Nucleic Acids Research** 29.1 (Jan. 2001), pp. 210–213. ISSN: 0305-1048. URL: <https://doi.org/10.1093/nar/29.1.210> (visited on 09/28/2022).
- [48] H. M. Salkin and C. A. De Kluyver. “The Knapsack Problem: A Survey.” In: **Naval Research Logistics Quarterly** 22.1 (1975), pp. 127–144. ISSN: 1931-9193. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800220110> (visited on 10/03/2022).
- [49] M. Z. Salleh, M. N. Norazmi, and Z. Z. Deris. “Immunogenicity Mechanism of mRNA Vaccines and Their Limitations in Promoting Adaptive Protection against SARS-CoV-2.” In: **PeerJ** 10 (Mar. 2022), e13083. ISSN: 2167-8359. URL: <https://peerj.com/articles/13083> (visited on 09/30/2022).
- [50] B. Schubert and O. Kohlbacher. “Designing String-of-Beads Vaccines with Optimal Spacers.” In: **Genome Medicine** 8.1 (Jan. 2016), p. 9. ISSN: 1756-994X. URL: <https://doi.org/10.1186/s13073-016-0263-6> (visited on 10/01/2022).
- [51] M. Skwarczynski and I. Toth. “Peptide-Based Synthetic Vaccines.” In: **Chemical science** 7.2 (2016), pp. 842–854. URL: <https://pubs.rsc.org/en/content/articlehtml/2016/sc/c5sc03892h>.
- [52] J. Tarhio and E. Ukkonen. “A Greedy Approximation Algorithm for Constructing Shortest Common Superstrings.” In: **Theoretical Computer Science** 57 pp. 131-145 (1988). ISSN: 0304-3975. URL: <https://www.sciencedirect.com/science/article/pii/0304397588901673>.
- [53] M. Thura et al. “Targeting Intra-Viral Conserved Nucleocapsid (N) Proteins as Novel Vaccines against SARS-CoVs.” In: **Bioscience reports** 41.9 (2021). URL: <https://portlandpress.com/bioscirep/article/41/9/BSR20211491/229773/Targeting-intra-viral-conserved-nucleocapsid-N>.

- [54] N. C. Toussaint, P. Dönnes, and O. Kohlbacher. “A Mathematical Framework for the Selection of an Optimal Set of Peptides for Epitope-Based Vaccines.” In: **PLOS Computational Biology** 4.12 (Dec. 2008), e1000246. ISSN: 1553-7358. URL: <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000246> (visited on 10/01/2022).
- [55] D. Turner. “ES07.02 The Human Leucocyte Antigen (HLA) System.” In: **Vox Sanguinis** 87.s1 (2004), pp. 87–90. ISSN: 1423-0410. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1741-6892.2004.00438.x> (visited on 09/30/2022).
- [56] E. Ukkonen. “A Linear-Time Algorithm for Finding Approximate Shortest Common Superstrings.” In: **Algorithmica. An International Journal in Computer Science** 5.1 (1990), pp. 313–323. URL: <https://link.springer.com/article/10.1007/BF01840391>.
- [57] T. Vider-Shalit, S. Raffaeli, and Y. Louzoun. “Virus-Epitope Vaccine Design: Informatic Matching the HLA-I Polymorphism to the Virus Genome.” In: **Molecular Immunology** 44.6 (Feb. 2007), pp. 1253–1261. ISSN: 0161-5890. URL: <https://www.sciencedirect.com/science/article/pii/S0161589006002094> (visited on 09/28/2022).
- [58] Q. Wang et al. “A Unique Protease Cleavage Site Predicted in the Spike Protein of the Novel Pneumonia Coronavirus (2019-nCoV) Potentially Related to Viral Transmissibility.” In: **Virologica Sinica** 35.3 (2020), pp. 337–339. URL: <https://link.springer.com/article/10.1007/s12250-020-00212-7>.
- [59] Ward Fleri. **Selecting Thresholds (Cut-Offs) for MHC Class I and II Binding Predictions**. 2020. URL: <https://help.iedb.org/hc/en-us/articles/114094151811-Selecting-thresholds-cut-offs-for-MHC-class-I-and-II-binding-predictions>.
- [60] B. R. Weeder, M. A. Wood, E. Li, A. Nellore, and R. F. Thompson. “Pepsickle Rapidly and Accurately Predicts Proteasomal Cleavage Sites for Improved Neoantigen Identification.” In: **bioRxiv : the preprint server for biology** (2021). URL: <https://academic.oup.com/bioinformatics/article/37/21/3723/6363787>.
- [61] B. J. Willett et al. “SARS-CoV-2 Omicron Is an Immune Escape Variant with an Altered Cell Entry Pathway.” In: **Nature Microbiology** 7.8 (Aug. 2022), pp. 1161–1179. ISSN: 2058-5276. URL: <https://www.nature.com/articles/s41564-022-01143-7> (visited on 09/29/2022).

## A Appendix ILP Formulations

### A.1 MSKS OCG Problem ILP Formulation

$$\text{maximize } \sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$$

$$\text{subject to } \sum_{(0,j) \in E} x_{0j} = 1 \quad // \text{ exactly one outgoing edge from source } v_0$$

$$\sum_{(j,n+1) \in E} x_{jn+1} = 1 \quad // \text{ exactly one incoming edge to sink } v_{n+1}$$

$$\sum_{(i,j) \in E} l_{ij} \cdot x_{ij} \leq k \quad // \text{ sum of edge weights } \leq k$$

$$\sum_{j: (i,j) \in E} x_{ij} \leq 1 \quad \forall i \in \{1, \dots, n\} \quad // \text{ each node has } \leq 1 \text{ outgoing edges}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} = \sum_{(j,g) \in \delta^+(j)} x_{jg} \quad \forall j \in V \quad // \text{ flow conservation}$$

$$\sum_{(i,j) \in \delta^-(j)} x_{ij} \leq \sum_{(h,g) \in \delta^-(W)} x_{hg} \quad \forall W \subset V, j \in W \quad // \text{ subtour elimination}$$

$$h_a \leq \sum_{\substack{j \in V: (i,j) \in E \\ s_j \in S}} x_{ij} \cdot B_{as_j} \quad \forall a \in \{1, \dots, m\} \quad // \text{ hit if } a \text{ and } j \text{ bind}$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E$$

$$h_a \in \{0, 1\} \quad \forall a \in \{1, \dots, m\}$$

## A.2 Concatenation ILP Formulation

$$\text{maximize } \sum_{a \in \{1, \dots, m\}} h_a \cdot f_a$$

$$\text{subject to } \sum_{s \in S} x_s \cdot |s| \leq k \quad // \text{ sum of string lengths } \leq k$$

$$h_a \leq \sum_{s \in S} x_s \cdot B_{as} \quad \forall a \in \{1, \dots, m\} \quad // \text{ hit if a and s bind}$$

$$x_s \in \{0, 1\} \quad \forall s \in S$$

$$h_a \in \{0, 1\} \quad \forall a \in \{1, \dots, m\}$$



## B Appendix Results

### B.1 Minimum Number of Required Hits

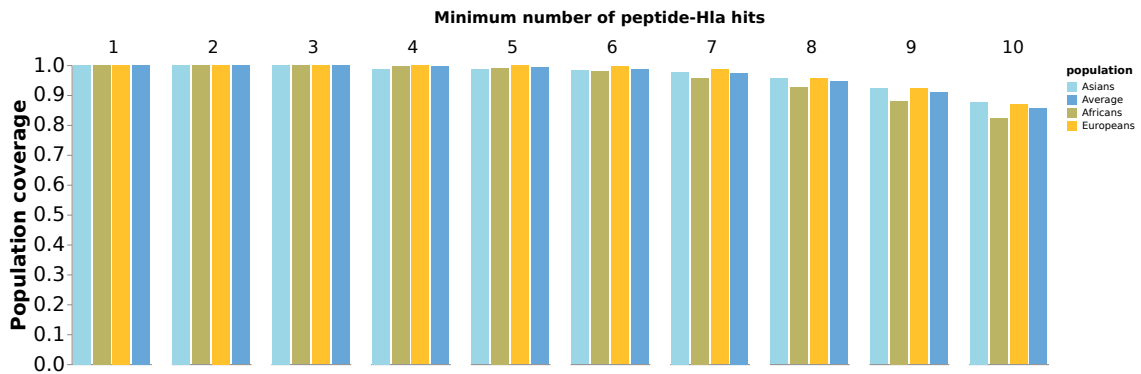


Figure 36: MHC class I evaluation with EvalVax-Robust for haplotype vaccine with a minimum number of required hits of 5.

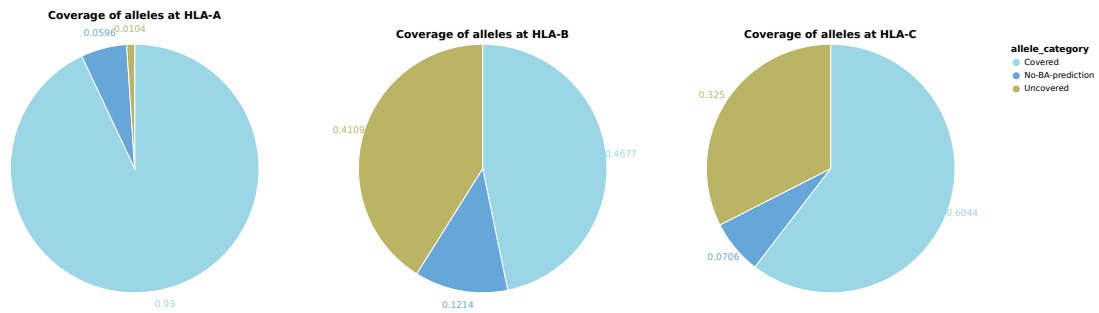


Figure 37: MHC class I evaluation with HOGVAX metric for haplotype vaccine with a minimum number of required hits of 5. As the coverage of HLA-A is so high, likely all haplotypes are covered.

### B.2 Comparison of HOGVAX and OptiVax-Robust on Genotype Data

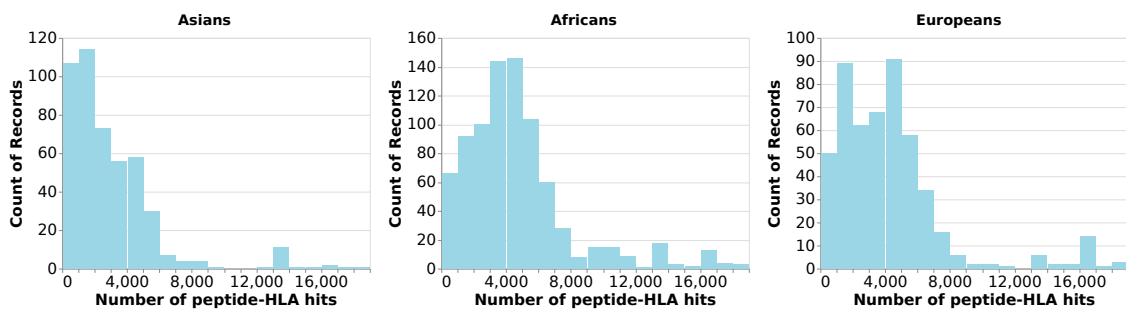


Figure 38: MHC class II hits per haplotype of the three populations for all MHC II input peptides

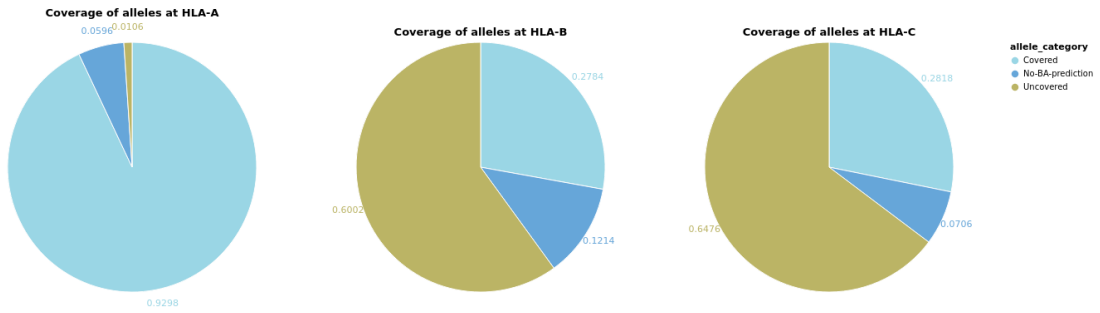


Figure 39: HOGVAX MHC class I genotype vaccine candidate evaluated with HOGVAX metric.

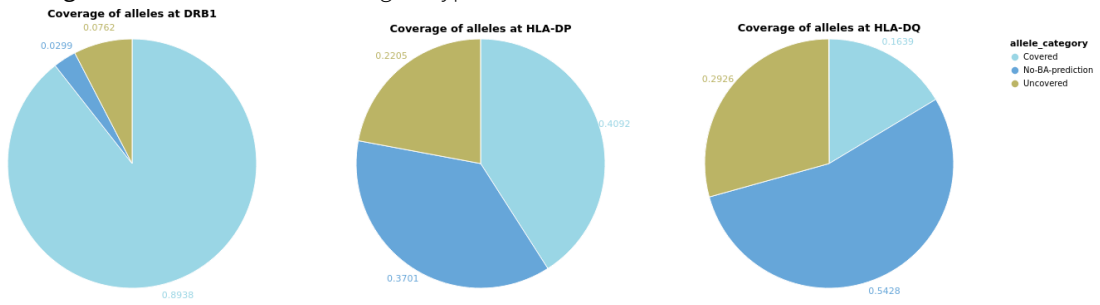


Figure 40: HOGVAX MHC class II genotype vaccine candidate evaluated with HOGVAX metric.

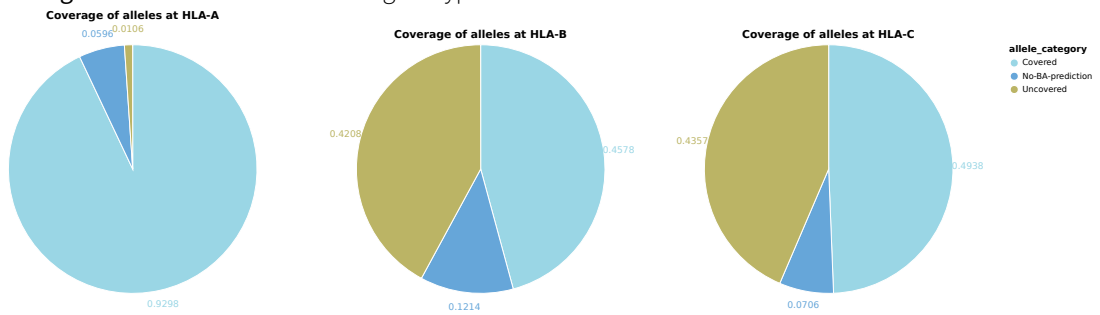


Figure 41: OptiVax-Robust MHC class I genotype vaccine candidate evaluated with HOGVAX metric.

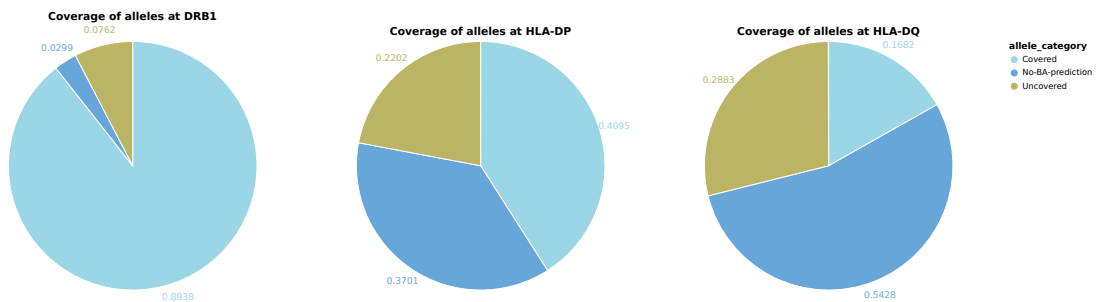


Figure 42: OptiVax-Robust MHC class II genotype vaccine candidate evaluated with HOGVAX metric.

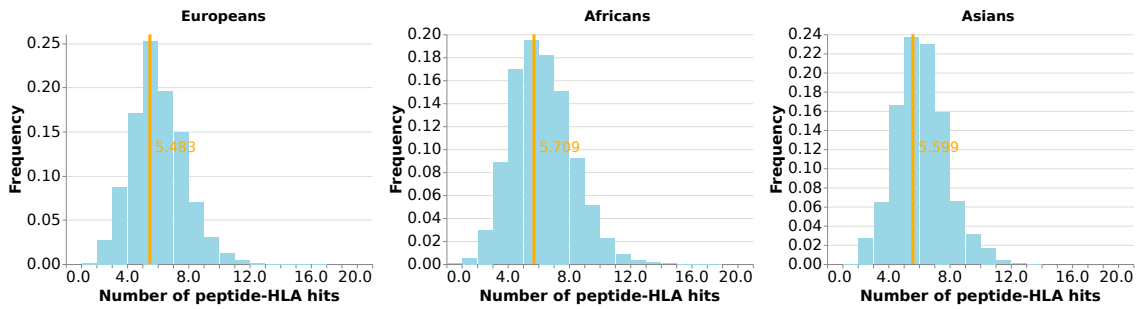


Figure 43: HOGVAX MHC class I genotype vaccine candidate peptide-HLA hit distribution and expected number of hits.

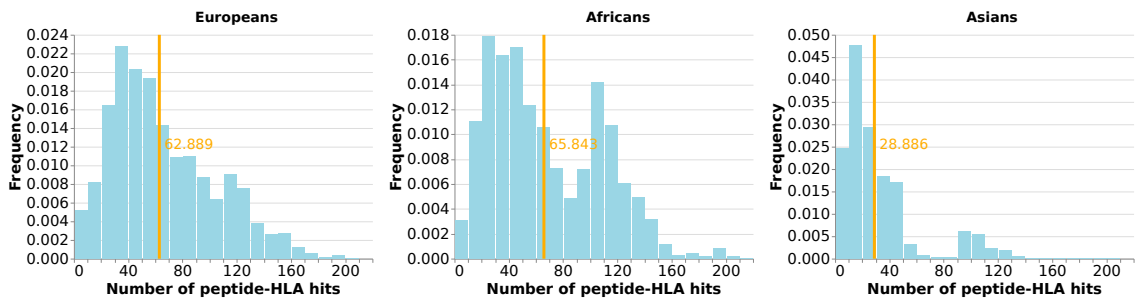


Figure 44: HOGVAX MHC class II genotype vaccine candidate peptide-HLA hit distribution and expected number of hits.

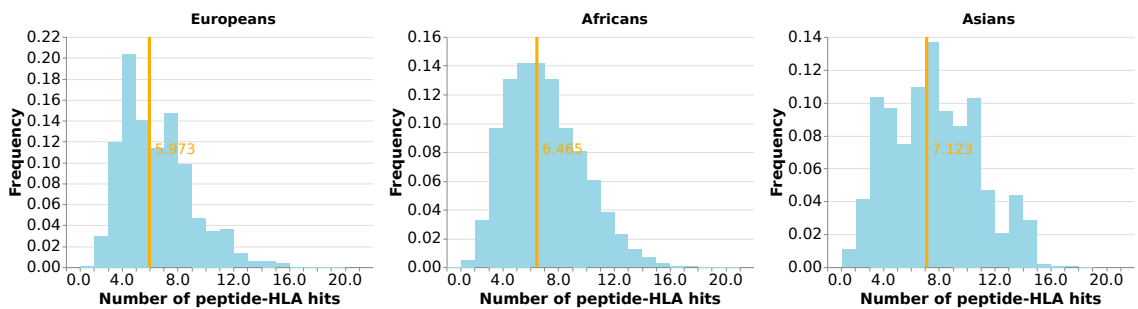


Figure 45: OptiVax-Robust MHC class I genotype vaccine candidate peptide-HLA hit distribution and expected number of hits.

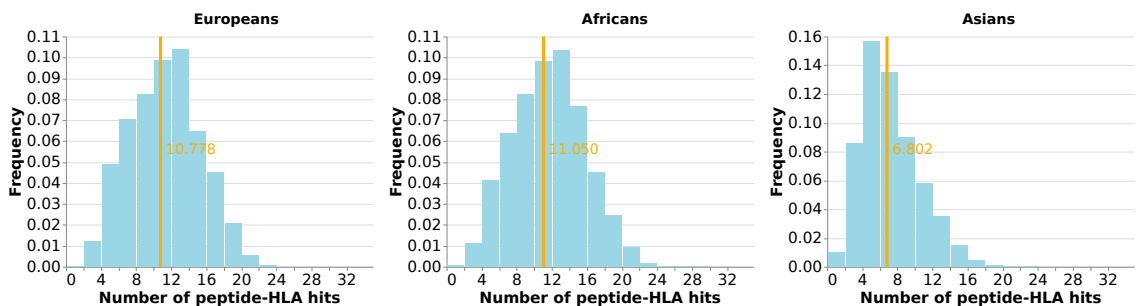


Figure 46: OptiVax-Robust MHC class II genotype vaccine candidate peptide-HLA hit distribution and expected number of hits.

### B.3 Designing Vaccines from the SARS-CoV-2 Spike Protein

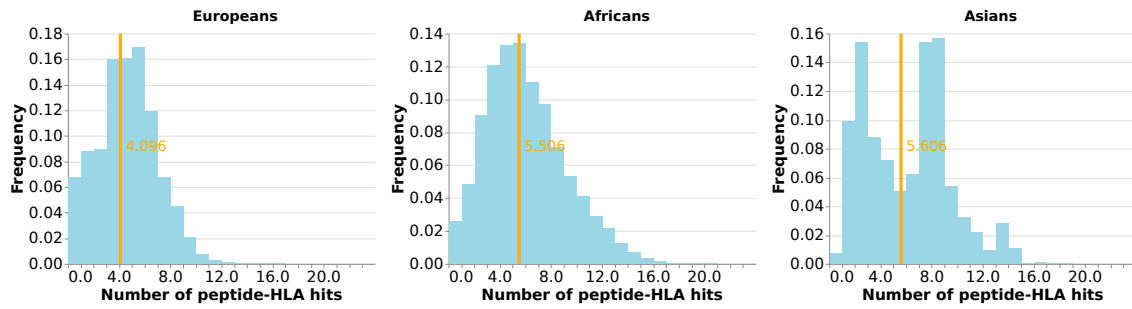


Figure 47: MHC I HOGVAX spike protein vaccine candidate peptide-HLA hit distribution and expected number of hits.

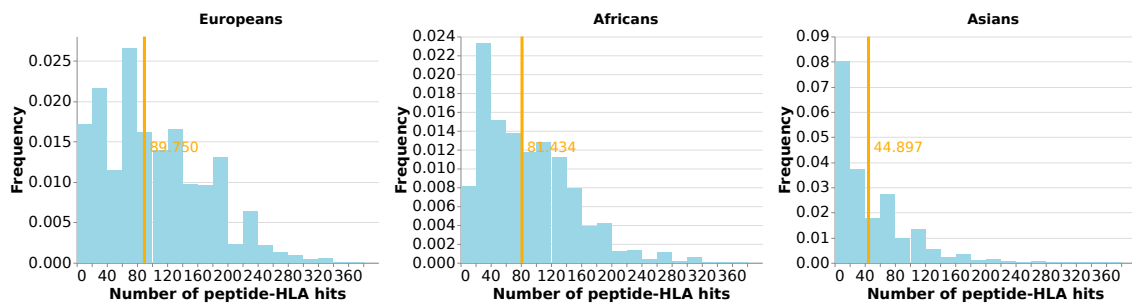


Figure 48: MHC II HOGVAX spike protein vaccine candidate peptide-HLA hit distribution and expected number of hits.

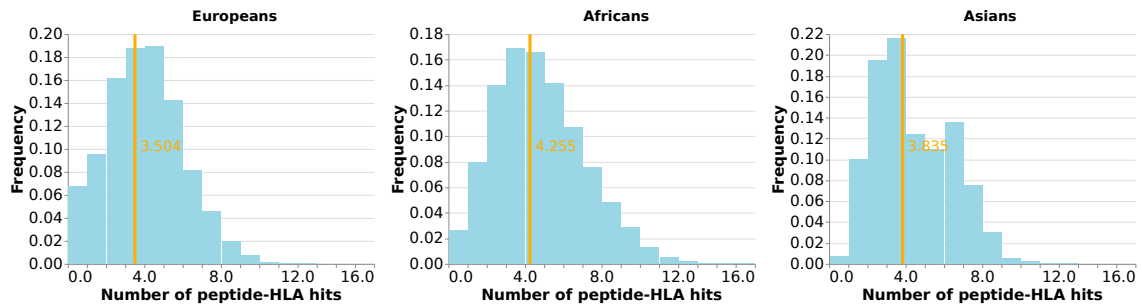


Figure 49: MHC I OptiVax-Unlinked spike protein vaccine candidate peptide-HLA hit distribution and expected number of hits.

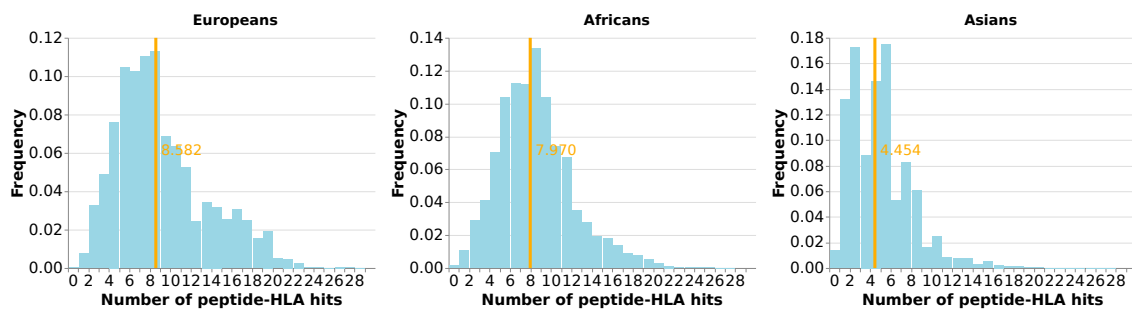


Figure 50: MHC II OptiVax-Unlinked spike protein vaccine candidate peptide-HLA hit distribution and expected number of hits.

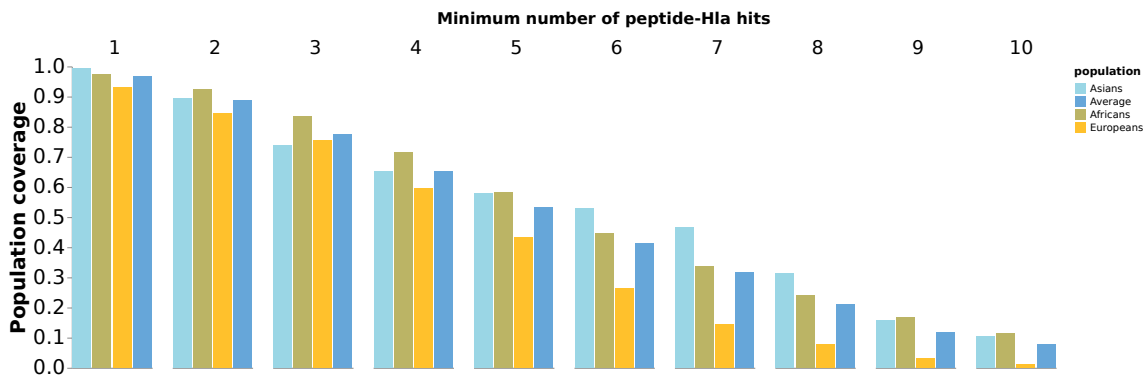


Figure 51: MHC I HOGVAX spike protein vaccine evaluation with EvalVax-Robust

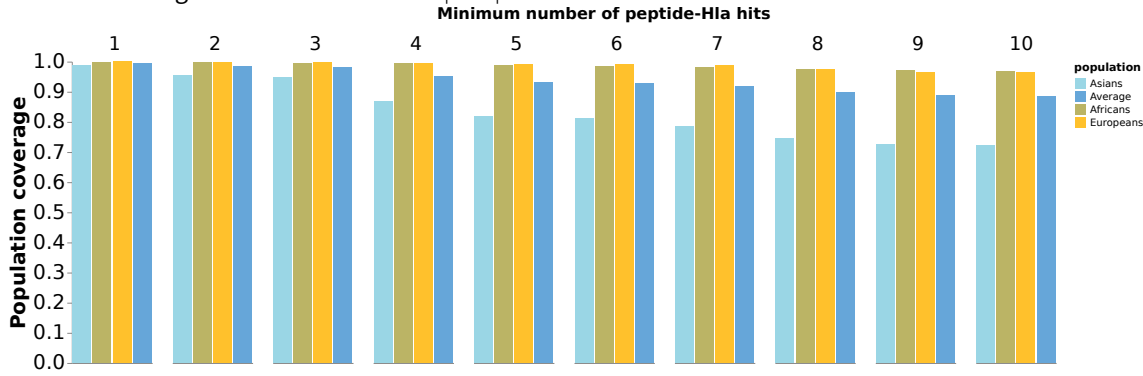


Figure 52: MHC II HOGVAX spike protein vaccine evaluation with EvalVax-Robust

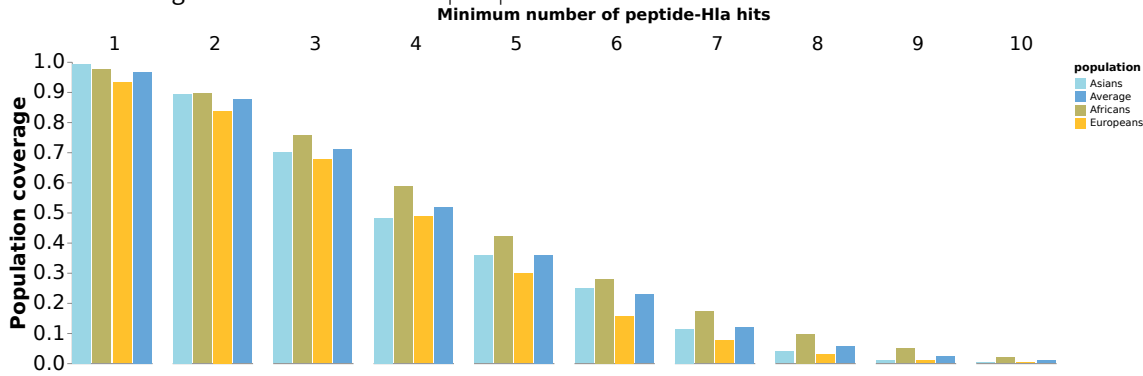


Figure 53: MHC I OptiVax spike protein vaccine evaluation with EvalVax-Robust

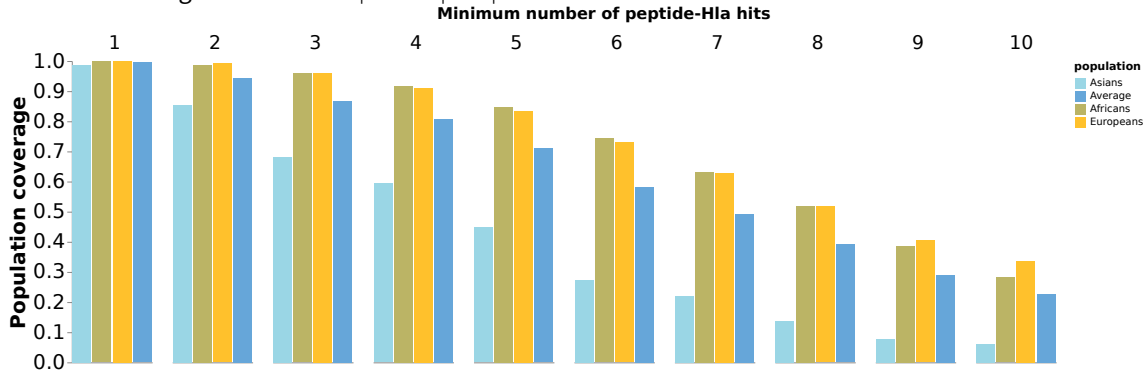


Figure 54: MHC II OptiVax spike protein vaccine evaluation with EvalVax-Robust

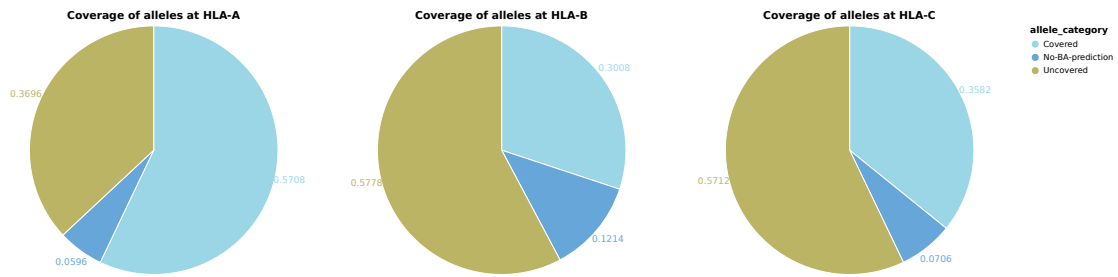


Figure 55: MHC I HOGVAX spike protein vaccine evaluation using HOGVAX metric.

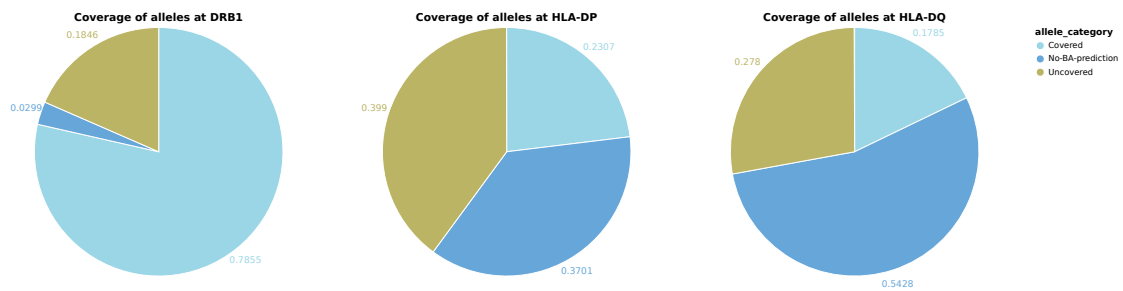


Figure 56: MHC II HOGVAX spike protein vaccine evaluation using HOGVAX metric.

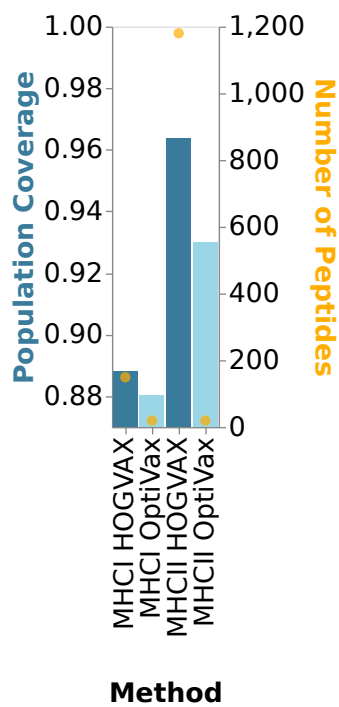


Figure 57: EvalVax-Unlinked evaluation of spike protein candidates computed by OptiVax and HOGVAX.

## B.4 Combined Vaccine for MHC Class I and II

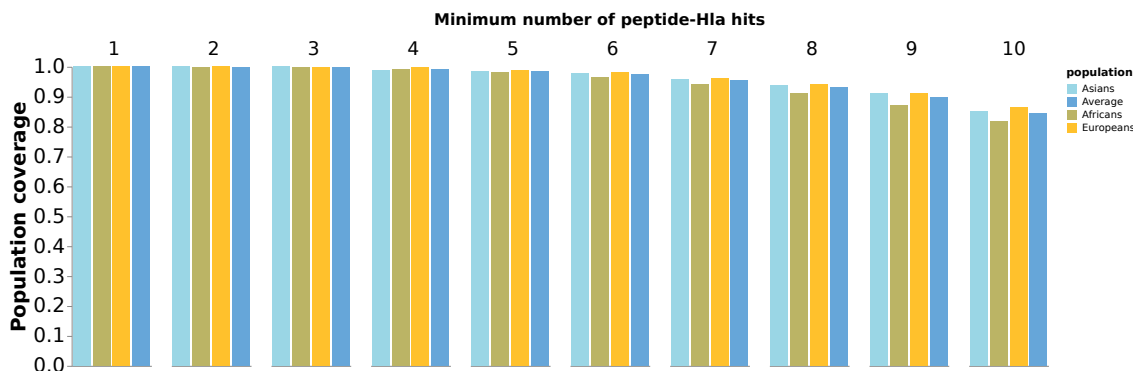


Figure 58: Evaluation of combined MHC class I and II HOGVAX vaccine with EvalVax-Robust on MHC class I data

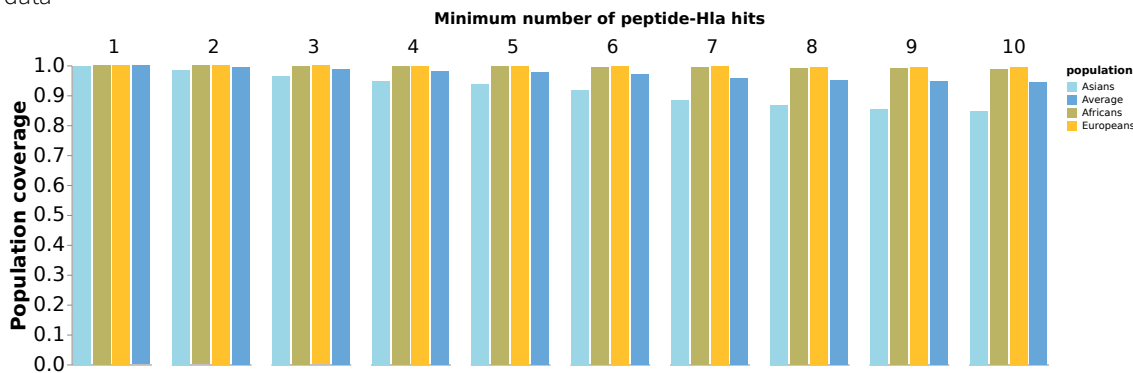


Figure 59: Evaluation of combined MHC class I and II HOGVAX vaccine with EvalVax-Robust on MHC class II data

## B.5 Vaccine Efficacy of Concatenation Approach compared to HOGVAX

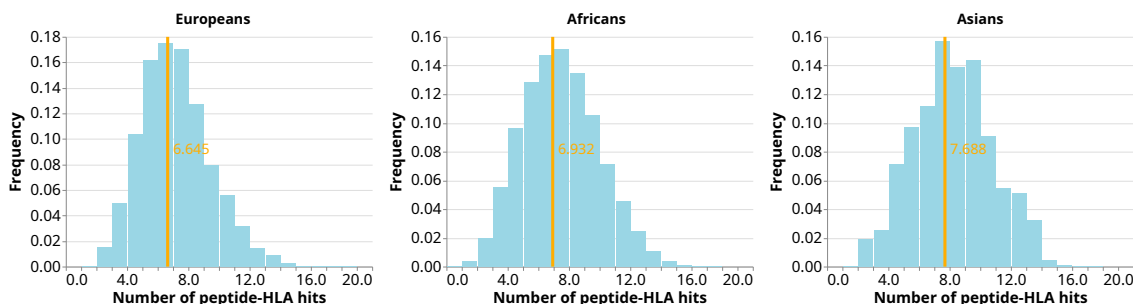


Figure 60: Concatenation MHC class I vaccine candidate peptide-HLA hit distribution and expected number of hits.

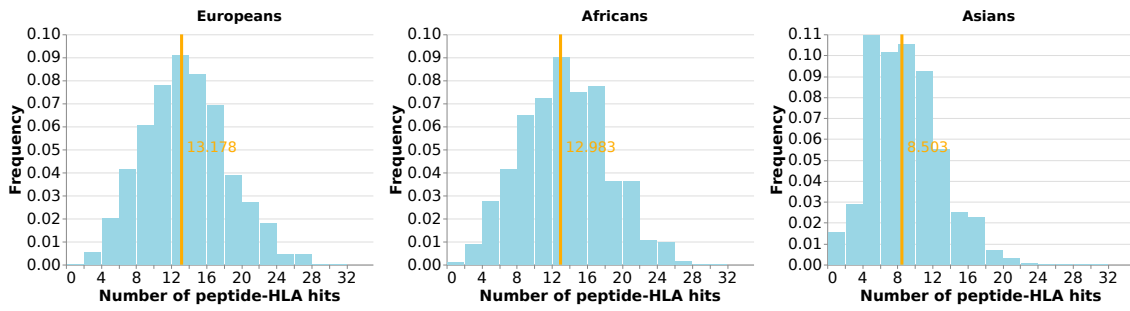


Figure 61: Concatenation MHC class II vaccine candidate peptide-HLA hit distribution and expected number of hits.

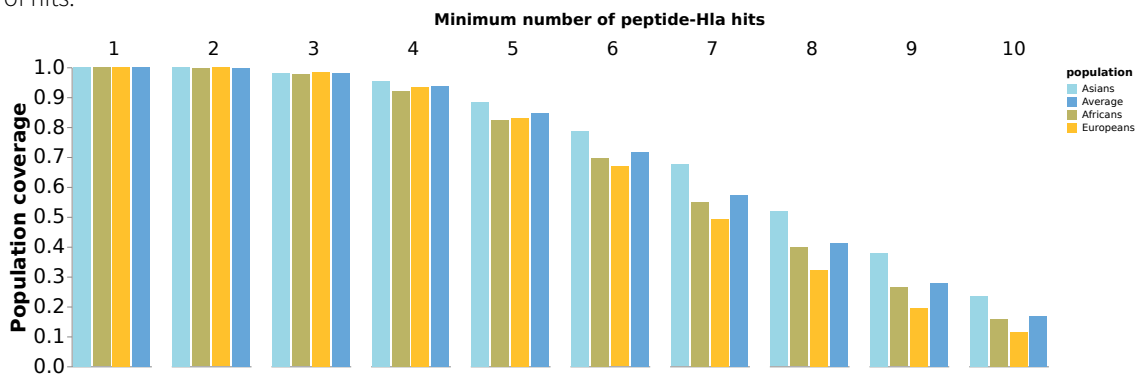


Figure 62: Concatenation MHC class I evaluation using EvalVax-Robust

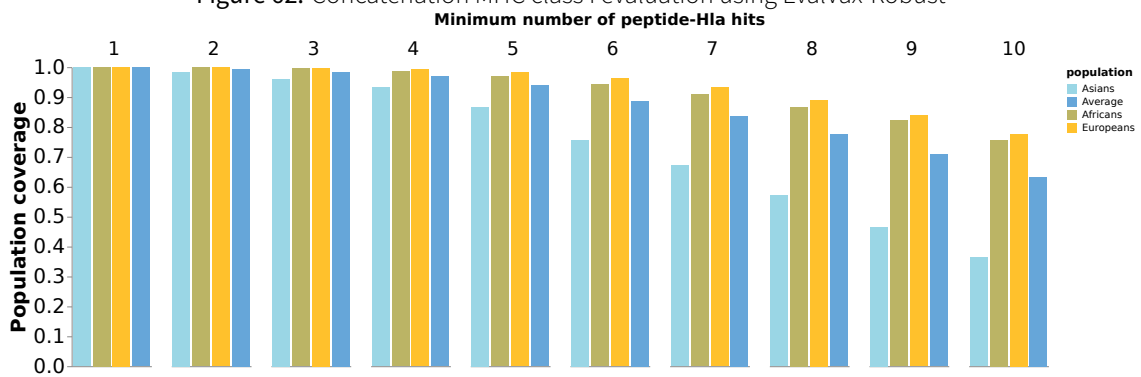


Figure 63: Concatenation MHC class II evaluation using EvalVax-Robust