A combinatorial vaccine design method using lambda-superstrings

Luis Martínez
Universidad del País Vasco-Euskal Herriko Unibertsitatea-University of the Basque Country
First workshop on interactions between mathematics and social sciences
One of the main goals of a vaccine is to recognize certain sequences of aminoacids, which are called epitopes.
One of the main goals of a vaccine is to recognize certain sequences of aminoacids, which are called epitopes. The problem is that many times the target is continually moving due to mutations that cause the change of those epitopes.
One of the main goals of a vaccine is to recognize certain sequences of aminoacids, which are called epitopes. The problem is that many times the target is continually moving due to mutations that cause the change of those epitopes.
In order to face this problem, we are interested in that the vaccine recognizes (cover) the most frequent epitopes (the better conserved ones).
In order to face this problem, we are interested in that the vaccine recognizes (cover) the most frequent epitopes (the better conserved ones).
In order to face this problem, we are interested in that the vaccine recognizes (cover) the most frequent epitopes (the better conserved ones).
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance
1. The way of choosing the adjuvant
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance:

1. The way of choosing the adjuvant
2. The realization of in-vitro tests
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance

1. The way of choosing the adjuvant
2. The realization of in-vitro tests
3. The realization of animal tests
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance

1. The way of choosing the adjuvant
2. The realization of in-vitro tests
3. The realization of animal tests
4. The realization of clinical tests
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance

1. The way of choosing the adjuvant
2. The realization of in-vitro tests
3. The realization of animal tests
4. The realization of clinical tests
5. The distribution of the vaccine
Of course, this mathematical modelization omits a lot of fundamental stages and elements such as for instance:

1. The way of choosing the adjuvant
2. The realization of in-vitro tests
3. The realization of animal tests
4. The realization of clinical tests
5. The distribution of the vaccine
6. And many, many more
Definition

Given $k$ base strings $S_1, \ldots, S_k$ and a string $t$, we define the frequency $f(t)$ of $t$ in $\{S_1, \ldots, S_k\}$ as the number of strings in $S_1, \ldots, S_k$ containing $t$.  

Example

Let us consider the strings

$S_1 = 0110101111$
$S_2 = 0010111100$
$S_3 = 1001001000$
$S_4 = 1101000000$
$S_5 = 1000011011$

The frequency of $111$ is 2
The frequency of $000$ is 3
The frequency of $010$ is 4
The frequency of $01$ is 5
Definition

Given $k$ base strings $S_1, \ldots, S_k$ and a string $t$, we define the frequency $f(t)$ of $t$ in $\{S_1, \ldots, S_k\}$ as the number of strings in $S_1, \ldots, S_k$ containing $t$.

Example

Let us consider the strings

$S_1 = 0110101111$, $S_2 = 0010111100$, $S_3 = 1001001000$, $S_4 = 1101000000$, $S_5 = 1000011011$. 
Definition

Given $k$ base strings $S_1, \ldots, S_k$ and a string $t$, we define the frequency $f(t)$ of $t$ in $\{S_1, \ldots, S_k\}$ as the number of strings in $S_1, \ldots, S_k$ containing $t$.

Example

Let us consider the strings

$S_1 = 0110101111$, $S_2 = 0010111100$, $S_3 = 1001001000$,
$S_4 = 1101000000$, $S_5 = 1000011011$.

The frequency of 111 is 2
Definition

Given \( k \) base strings \( S_1, \ldots, S_k \) and a string \( t \), we define the frequency \( f(t) \) of \( t \) in \( \{S_1, \ldots, S_k\} \) as the number of strings in \( S_1, \ldots, S_k \) containing \( t \).

Example

Let us consider the strings
\[
S_1 = 0110101111, \quad S_2 = 0010111100, \quad S_3 = 1001001000, \\
S_4 = 1101000000, \quad S_5 = 1000011011.
\]

The frequency of 111 is 2
The frequency of 000 is 3
Definition

Given $k$ base strings $S_1, \ldots, S_k$ and a string $t$, we define the frequency $f(t)$ of $t$ in $\{S_1, \ldots, S_k\}$ as the number of strings in $S_1, \ldots, S_k$ containing $t$.

Example

Let us consider the strings

$S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000,$

$S_4 = 1101000000, S_5 = 1000011011.$

The frequency of 111 is 2
The frequency of 000 is 3
The frequency of 010 is 4
**Definition**

Given $k$ base strings $S_1, \ldots, S_k$ and a string $t$, we define the frequency $f(t)$ of $t$ in $\{S_1, \ldots, S_k\}$ as the number of strings in $S_1, \ldots, S_k$ containing $t$.

**Example**

Let us consider the strings

$S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000,$
$S_4 = 1101000000, S_5 = 1000011011.$

The frequency of 111 is 2
The frequency of 000 is 3
The frequency of 010 is 4
The frequency of 01 is 5
Usually we are interested in having a set \( \mathcal{T} \) of target strings contained in some of the base strings \( S_1, \ldots, S_k \) and a string \( v \) containing the chains of \( \mathcal{T} \) which are more frequent in \( S_1, \ldots, S_k \).
Usually we are interested in having a set $\mathcal{T}$ of target strings contained in some of the base strings $S_1, \ldots, S_k$ and a string $v$ containing the chains of $\mathcal{T}$ which are more frequent in $S_1, \ldots, S_k$.

**Definition**

Given $S_1, \ldots, S_k, \mathcal{T}$ and $v$, we define the **coverage level of** $v$ to be

$$c(v) = \frac{\sum_{t \in \mathcal{T} : t \text{ substring of } v} f(t)}{\sum_{t \in \mathcal{T} : t \text{ substring of some } S_i} f(t)}.$$
Example

Let us consider again the strings

\( S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, \)
\( S_4 = 1101000000, S_5 = 1000011011 \)

with \( T = \{000, 001, 010, 011, 100, 101, 110, 111\} \) and the string

\( v = 01001 \)
Example

Let us consider again the strings

\[ S_1 = 0110101111, \; S_2 = 0010111100, \; S_3 = 1001001000, \]
\[ S_4 = 1101000000, \; S_5 = 1000011011 \]

with \( T = \{000, 001, 010, 011, 100, 101, 110, 111\} \) and the string
\( v = 01001 \)

The respective frequencies of the target strings are 3, 3, 4, 3, 4, 4, 2, and therefore

\[ \sum_{t \in T : t \text{ substring of some } S_i} f(t) = 27. \]
Example

*Let us consider again the strings*

\[ S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, \]
\[ S_4 = 1101000000, S_5 = 1000011011 \]
*with \( T = \{000, 001, 010, 011, 100, 101, 110, 111\} \) and the string \( v = 01001 \)*

*The respective frequencies of the target strings are 3, 3, 4, 3, 4, 4, 2, and therefore\[
\sum_{t \in T : t \text{ substring of some } S_i} f(t) = 27.
\]*

*On the other hand, the target strings contained in \( v \) are 010, 100 and 001, and hence \( \sum_{t \in \mathcal{T} : t \text{ substring of } v} f(t) = 11 \),*
Example

Let us consider again the strings

\[ S_1 = 0110101111, \quad S_2 = 0010111100, \quad S_3 = 1001001000, \]
\[ S_4 = 1101000000, \quad S_5 = 1000011011 \]

with \( T = \{000, 001, 010, 011, 100, 101, 110, 111\} \) and the string \( v = 01001 \)

The respective frequencies of the target strings are 3, 3, 4, 3, 4, 4, 2, and therefore

\[ \sum_{t \in T : t \text{ substring of some } S_i} f(t) = 27. \]

On the other hand, the target strings contained in \( v \) are 010, 100 and 001, and hence

\[ \sum_{t \in T : t \text{ substring of } v} f(t) = 11, \]

Thus, \( c(v) = 11/27 \approx 0.407 \)
But, although it is necessary that a candidate vaccine covers the most frequent epitopes, it is not sufficient.
But, although it is necessary that a candidate vaccine covers the most frequent epitopes, it is not sufficient. We are interested also in that it covers them in a balanced way.
But, although it is necessary that a candidate vaccine covers the **most frequent** epitopes, it is not sufficient. We are interested also in that it covers them in a **balanced** way.
And this leads us to the definition of $\lambda$-superstring.
And this leads us to the definition of $\lambda$-superstring.

**Definition**

*Given k base strings $S_1, \ldots, S_k$, a set $T$ of target string and a natural number $\lambda$, we will call $\lambda$-superstring for $(S_1, \ldots, S_k, T)$ to a string $v$ satisfying that, for each $i \in \{1, \ldots, k\}$, at least $\lambda$ target strings are substrings of both $S_i$ and $v$.***
Let us return again to the example with $S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, S_4 = 1101000000, S_5 = 1000011011$, where $T = \{000, 001, 010, 011, 100, 101, 110, 111\}$, for which we had the string $v = 01001$, whose coverage level was $11/27$. 
Let us return again to the example with 
$S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000,$
$S_4 = 1101000000, S_5 = 1000011011.$
where $T = \{000, 001, 010, 011, 100, 101, 110, 111\}$, for which we had the string $v = 01001$, whose coverage level was $11/27$. The distribution of the number of target strings covered by $v$ in the $S_i$ is $(1, 3, 3, 2, 2).$
Let us return again to the example with
\[ S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, \]
\[ S_4 = 1101000000, S_5 = 1000011011. \]
where \( \mathcal{T} = \{000, 001, 010, 011, 100, 101, 110, 111\} \), for which we had the string \( v = 01001 \), whose coverage level was \( 11/27 \). The distribution of the number of target strings covered by \( v \) in the \( S_i \) is \((1, 3, 3, 2, 2)\).

If we consider instead the string \( v' = 00101 \), which is of the same length that \( v \),
Let us return again to the example with

\[ S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, \]
\[ S_4 = 1101000000, S_5 = 1000011011. \]

where \( \mathcal{T} = \{000, 001, 010, 011, 100, 101, 110, 111\} \), for which we had the string \( \mathbf{v} = 01001 \), whose coverage level was \( 11/27 \). The distribution of the number of target strings covered by \( \mathbf{v} \) in the \( S_i \) is \( (1, 3, 3, 2, 2) \).

If we consider instead the string \( \mathbf{v'} = 00101 \), which is of the same length that \( \mathbf{v} \), its coverage level is again \( 11/27 \),
Let us return again to the example with
\( S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, \)
\( S_4 = 1101000000, S_5 = 1000011011. \)
where \( T = \{000, 001, 010, 011, 100, 101, 110, 111\} \), for which we
had the string \( v = 01001 \), whose coverage level was 11/27. The
distribution of the number of target strings covered by \( v \) in the \( S_i \)
is \((1, 3, 3, 2, 2)\).
If we consider instead the string \( v' = 00101 \), which is of the same
length that \( v \), its coverage level is again 11/27, and the
distribution of the number of target strings covered is \((2, 3, 2, 2, 2)\),
Let us return again to the example with 
\( S_1 = 0110101111, S_2 = 0010111100, S_3 = 1001001000, 
S_4 = 1101000000, S_5 = 1000011011 \).

where \( \mathcal{T} = \{000, 001, 010, 011, 100, 101, 110, 111\} \), for which we
had the string \( \nu = 01001 \), whose coverage level was 11/27. The
distribution of the number of target strings covered by \( \nu \) in the \( S_i \)
is \((1, 3, 3, 2, 2)\).

If we consider instead the string \( \nu' = 00101 \), which is of the same
length that \( \nu \), its coverage level is again 11/27, and the
distribution of the number of target strings covered is \((2, 3, 2, 2, 2)\),
and the distribution is more balanced.
When the target strings are $A^\ell$, where $A$ is the considered alphabet, and the base strings $S_1, \ldots, S_k$ are of the same length $m$, we can get good mathematical properties for the coverage level of the $\lambda$-superstrings, so that when $\lambda$ goes to $m - \ell + 1$, the coverage level $c(v)$ goes to 1.
When the target strings are $A^\ell$, where $A$ is the considered alphabet, and the base strings $S_1, \ldots, S_k$ are of the same length $m$, we can get good mathematical properties for the coverage level of the $\lambda$-superstrings, so that when $\lambda$ goes to $m - \ell + 1$, the coverage level $c(v)$ goes to 1.

**Teorema**

If $T = A^\ell$ and $S_1, \ldots, S_k \in A^m$ for some positive integer $\ell$, then the coverage level of any $\lambda$-superstring $v$ satisfies the inequality $c(v) \geq \frac{\lambda}{m - \ell + 1}$. 
We are interested, from the point of view of the applications to the computational design of vaccines, in getting $\lambda$-superstrings of minimum length,
We are interested, from the point of view of the applications to the computational design of vaccines, in getting $\lambda$-superstrings of minimum length, and this lead us to the following combinatorial optimization problem:
We are interested, from the point of view of the applications to the computational design of vaccines, in getting \( \lambda \)-superstrings of minimum length, and this lead us to the following combinatorial optimization problem:

\[
\text{To find, given } S_1, \ldots, S_k \text{ y } T, \text{ a shortest } \lambda \text{-superstring.}
\]
It is natural to ask about the computational complexity of the problem of combinatorial optimization just introduced.
It is natural to ask about the computational complexity of the problem of combinatorial optimization just introduced.
Complexity of the solution

Let us first introduce some classes of problems that are studied in complexity theory:

**Definition**
A problem is in class $P$ if it can be solved in at most $f(n)$ steps, where $f$ is a polynomial and $n$ is the size (in bits) of the input.

**Example**
The problem of solving a system of linear equations is in $P$. 

A combinatorial vaccine design method using lambda-superstrings
Let us first introduce some classes of problems that are studied in complexity theory:
Let us first introduce some classes of problems that are studied in complexity theory:

**Definition**

*Problem is in class P if it can be solved in at most \( f(n) \) steps, where \( f \) is a polynomial and \( n \) is the size (in bits) of the input.*
Let us first introduce some classes of problems that are studied in complexity theory:

**Definition**

A problem is in class $P$ if it can be solved in at most $f(n)$ steps, where $f$ is a polynomial and $n$ is the size (in bits) of the input.

**Example**

The problem of solving a system of linear equations is in $P$. 
Complexity of the solution

**Definition**

*Informally speaking, a problem is in NP if a solution can be tested in polynomial time.*
Complexity of the solution

**Definition**

*Informally speaking, a problem is in NP if a solution can be tested in polynomial time.*

**Example**

*The problem of determining a proper coloring in a graph using a given number of colors is in NP.*
Another complexity class is the one of \( NP \)-hard problems.
Another complexity class is the one of $NP$-hard problems.

**Definition**

A problem $H$ is $NP$-hard if every problem in $NP$ can be polynomially transformed in $H$. 
Another complexity class is the one of $NP$-hard problems.

**Definition**

A problem $H$ is $NP$-hard if every problem in $NP$ can be polinomially transformed in $H$.

The problem of finding a shortest $\lambda$-superstring is $NP$-hard.
Another complexity class is the one of $NP$-hard problems

**Definition**

A problem $H$ is $NP$-hard if every problem in $NP$ can be polynomially transformed in $H$.

The problem of finding a shortest $\lambda$-superstring is $NP$-hard. Still worse, the problem of finding a $\lambda$-superstring of length close to the minimum one is $NP$-hard.
Hill-climbing algorithm

The previous result makes necessary the use of heuristic methods of combinatorial optimization.
The previous result makes necessary the use of heuristic methods of combinatorial optimization. In particular, we have used a hill-climbing algorithm.
Hill-climbing algorithm

The previous result makes necessary the use of heuristic methods of combinatorial optimization. In particular, we have used a hill-climbing algorithm.

(i) First, we ensembl, for a given $\lambda$, sequences of $\lambda$ consecutive target strings corresponding to the base strings.
The previous result makes necessary the use of heuristic methods of combinatorial optimization. In particular, we have used a hill-climbing algorithm.

(i) First, we ensemble, for a given \( \lambda \), sequences of \( \lambda \) consecutive target strings corresponding to the base strings.

(ii) Then, we apply consecutively two kinds of transformations, while we have a \( \lambda \)-superstring, to the obtained string.

(iii) In the transformations of the first kind, we eliminate one of the target strings.

(iv) In the transformations of the second kind, we change one of the target string for a different one.
The previous result makes necessary the use of heuristic methods of combinatorial optimization. In particular, we have used a hill-climbing algorithm.

(i) First, we ensemble, for a given $\lambda$, sequences of $\lambda$ consecutive target strings corresponding to the base strings.

(ii) Then, we apply consecutively two kinds of transformations, while we have a $\lambda$-superstring, to the obtained string.

(iii) In the transformations of the first kind, we eliminate one of the target strings.
The previous result makes necessary the use of heuristic methods of combinatorial optimization. In particular, we have used a hill-climbing algorithm.

(i) First, we ensemble, for a given $\lambda$, sequences of $\lambda$ consecutive target strings corresponding to the base strings.

(ii) Then, we apply consecutively two kinds of transformations, while we have a $\lambda$-superstring, to the obtained string.

(iii) In the transformations of the first kind, we eliminate one of the target strings.

(iv) In the transformations of the second kind, we change one of the target string for a different one.
(v) When none of the transformations of the first kind or of the second kind gives a $\lambda$-superstring, we record the obtained string.
(v) When none of the transformations of the first kind or of the second kind gives a $\lambda$-superstring, we record the obtained string.

(vi) The previous process is repeated a given number of times, and we keep the shortest $\lambda$-superstring of the obtained ones.
We have combined the Hill-climbing algorithm described below for \( \lambda = 45 \) with an ulterior adition of the most frequent target strings of length 9 to get a string from a set of 169 strings of the Nef protein of HIV.
<table>
<thead>
<tr>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
<th>Accession</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB012824</td>
<td>AF120887</td>
<td>AF129375</td>
<td>AF203172</td>
<td>AF238268</td>
<td>AY121441</td>
<td>AY835772</td>
<td>L15515</td>
</tr>
<tr>
<td>AB034257</td>
<td>AF120898</td>
<td>AF129376</td>
<td>AF203180</td>
<td>AF252897</td>
<td>AY173951</td>
<td>AY835776</td>
<td>L15518</td>
</tr>
<tr>
<td>AB034272</td>
<td>AF120909</td>
<td>AF129377</td>
<td>AF203188</td>
<td>AF252910</td>
<td>AY308762</td>
<td>AY835799</td>
<td>M17451</td>
</tr>
<tr>
<td>AB078005</td>
<td>AF129334</td>
<td>AF129378</td>
<td>AF203192</td>
<td>AF462708</td>
<td>AY314063</td>
<td>AY835780</td>
<td>M21098</td>
</tr>
<tr>
<td>AB221005</td>
<td>AF129335</td>
<td>AF129379</td>
<td>AF203194</td>
<td>AF462753</td>
<td>AY331285</td>
<td>AY857022</td>
<td>M26727</td>
</tr>
<tr>
<td>AF004394</td>
<td>AF129342</td>
<td>AF129382</td>
<td>AF203198</td>
<td>AF538302</td>
<td>AY331290</td>
<td>AY857144</td>
<td>M58173</td>
</tr>
<tr>
<td>AF011471</td>
<td>AF129343</td>
<td>AF129388</td>
<td>AF219672</td>
<td>AF538304</td>
<td>AY331293</td>
<td>AY899356</td>
<td>M93259</td>
</tr>
<tr>
<td>AF011474</td>
<td>AF129346</td>
<td>AF129389</td>
<td>AF219685</td>
<td>AF538305</td>
<td>AY352275</td>
<td>AY899382</td>
<td>U03295</td>
</tr>
<tr>
<td>AF011481</td>
<td>AF129347</td>
<td>AF129390</td>
<td>AF219691</td>
<td>AF538306</td>
<td>AY444311</td>
<td>DQ007902</td>
<td>U03338</td>
</tr>
<tr>
<td>AF011487</td>
<td>AF129350</td>
<td>AF129392</td>
<td>AF219729</td>
<td>AJ271445</td>
<td>AY713408</td>
<td>DQ085869</td>
<td>U03343</td>
</tr>
<tr>
<td>AF011493</td>
<td>AF129351</td>
<td>AF129394</td>
<td>AF219755</td>
<td>AJ430664</td>
<td>AY739040</td>
<td>DQ121815</td>
<td>U12055</td>
</tr>
<tr>
<td>AF042101</td>
<td>AF129352</td>
<td>AF203108</td>
<td>AF219760</td>
<td>AY037269</td>
<td>AY779550</td>
<td>DQ121883</td>
<td>U16863</td>
</tr>
<tr>
<td>AF047082</td>
<td>AF129354</td>
<td>AF203111</td>
<td>AF219765</td>
<td>AY037282</td>
<td>AY786630</td>
<td>DQ127537</td>
<td>U16875</td>
</tr>
<tr>
<td>AF063926</td>
<td>AF129355</td>
<td>AF203116</td>
<td>AF219771</td>
<td>AY116676</td>
<td>AY786750</td>
<td>DQ127548</td>
<td>U16934</td>
</tr>
<tr>
<td>AF069139</td>
<td>AF129362</td>
<td>AF203126</td>
<td>AF219782</td>
<td>AY116713</td>
<td>AY835748</td>
<td>DQ487191</td>
<td>U23487</td>
</tr>
<tr>
<td>AF120745</td>
<td>AF129364</td>
<td>AF203137</td>
<td>AF219792</td>
<td>AY116714</td>
<td>AY835751</td>
<td>DQ659737</td>
<td>U24455</td>
</tr>
<tr>
<td>AF120772</td>
<td>AF129369</td>
<td>AF203141</td>
<td>AF219800</td>
<td>AY116727</td>
<td>AY835753</td>
<td>L07422</td>
<td>U26087</td>
</tr>
<tr>
<td>AF120840</td>
<td>AF129370</td>
<td>AF203153</td>
<td>AF219812</td>
<td>AY116781</td>
<td>AY835762</td>
<td>L15482</td>
<td>U26110</td>
</tr>
<tr>
<td>AF120851</td>
<td>AF129372</td>
<td>AF203161</td>
<td>AF219819</td>
<td>AY116805</td>
<td>AY835765</td>
<td>L15489</td>
<td>U26119</td>
</tr>
<tr>
<td>AF120867</td>
<td>AF129373</td>
<td>AF203165</td>
<td>AF219845</td>
<td>AY116830</td>
<td>AY835770</td>
<td>L15500</td>
<td>U26138</td>
</tr>
</tbody>
</table>
The resulting \( \lambda \)-superstring was

\[
YTPGPGRFPLTFGWCFKLVVPDPEEVGFVYQVPLRPMTYKAAVDSLHFLQNYTPGPGRYPYPLTFGWCFKLVVP
QNYTPPGPGVRYPLTGWPTVRERMRRAEPAAEVGAVERDLEHGAITSSNTAATNADCAWLPERMTYKAALDLHFLR
EGGLEDLHSQKRQDILDLWYHTQGYFPAADGVGAASRDLDEKHMDFDPEREYLRFSRLAFFHHVARELHEYYKD
CFKLVPVEPEKIEANEGENNNLHIPLMSLHGMEDPEKEVLWVFDSRLVFPEVEKIEANEGENNNQLHELLMSQHMGGKW
SKRSVEKANEGENNAACAWLEAQQDEEEVGFVPRQVPLRPMTYKGALDLHFLKEAREKHPEYYYKQEIQLDLWVYHTQG
YFPDWMGGKWSKSSITSSNTAANNADCAWLEAQEEEEVGFVPRQVPLRPMTYKGAHLSHFLKEKKGEGHIYQSQRQDILD
LVYHNSLLHMSQHGMDFDEPEKELWVFDSRLAFFHMRHELHPEYKNCHELHMSLHGMDFDEPKGGSGLYQKRQDILD
DLWVYNTQGYFDPWQNYTPPGIRYPLTFGWPAVRERRMAEPAADGVGAVERDLEHGAITSSNTAT
\]
Hill-climbing algorithm

The resulting $\lambda$-superstring was

YTPGPGRTRFPLTFGCFLVPVDPEEVGFPPVKPQVPLRPMTYKAAVDLSHFLQNYTPGPGRTRYPLTFGCFLVPPVEPD
QNYTPGPGRVRYPLTFGWPTVRERMRRAEPAAGVGAVSRDLERHGAITSSNNTAATNADCAWLERPMTYKAAALDSHFLR
EKGGLEGLIHSLQKRQDILDLDLYIHTQGYFPAGDVHAASRDLEGKHMDDPEREVLEWRFDSSLAFHHVARELHPESYKDC
CFKLVPVEPEKIEANEAGENNSLLHPSLHGMEDPEKEVLMKWKFDSSLVPVEPEKVEEANEAGENNNCLLHPMSQHMGGKW
SKRSVEKANENGQNAACAWLEAEEVGFPVRPQVPRLRPMTYKGAELSHFLEAIREKHEPYKYRQEILDLWLRYHTQGYFMDW
MGKWSKSSITSSNTAANNADCAWLEAEEVGFPVRPMTYKGAELSHFLEKGGLEGLVYSSQQRDILDLDLW
VYHNSSLHPMSQHGMDDPEKEVLMKWKFDSSLAFHHMARELHPESYKNCNLLHPSLHGMDDPEKGGLEGLIYSQKRQDILD
DLWVYNTQGNYFPDWQNYTPGPGRTRYPLTFGCFLVPVARERMRAEPAADGVGASRDLEGKHAITSSNTAT

That string contained all the well-conserved syrings (at 90 %)
The resulting $\lambda$-superstring was

\begin{verbatim}
YTPGPGTRFPLTFGWCFLVPDVPEEVGFVVKPQVPLRPMTYKAADVLSHFLQNYTPGPGTRYPLTFGWCFLVPEPD
QNYTPGPGVRYPLTFGWPTVRERMRRAEPAAEAGVGAISRDPRLRERHGAISSNTAATNADCAWLERPMTYKAALDSHFLR
EKGGLEGLIHSQKRQDILDCLWYHTQGYFPAADGVAASRDLEKHGMDDPEREVLWRFDSRLAFHHVARELHPEYYKD
CFKLVPVEPEKIEEANEGENNSLLHPMSLHGMDPEKEVLMWKFDSRLVPVEPEKEVIEEANEGENNCLLHPMSQHMGGKW
SKRSEEKANEGENNAACAWLEAQDEEVEGFPVRPQVPLRPMTYKGDLSHFLKEAREKHPEYYKQREIDLDLWYHTQG
YFPDWMGGKWSKSSITSSNTAANNADCAWLEAEHIEEVEGFPVRPMTYKGAVDLHFLKEKGLEGLVYSQRRQDILDLD
VYHNSSLHPMSQHGMDDPEKEVLMWKFDSRLAFHHMARELHPEYYKNCLLHPMSLHGMDDPEKEGGLEGLYSQKRQDIL
DLWYNTQGYFPDWWQNYTPGPGIRYPLTFGWPAVRERMRAEPADGVAISRDELKHGAITSSNTAT
\end{verbatim}

That string contained all the well-conserved syrings (at 90 %) and we reached the same coverage level (62 %) that the one obtained by Nickle et al. in “Coping with viral diversity in HIV vaccine design”.
The resulting $\lambda$-superstring was

YTPGPGRFPLTFGWCFLVPDPEEVGFPVKPQVPLRPMTYKAADVLDHFLQNYTPGPGRYPLTFGWCFLVPLVDPE
QNYTPGPGRVRYPLTFGWPTVRERMRRAEAEGVGAISRDLERHGTAITSSNTAATNADCWLERPMTTYKAALDSHFLR
EKGGLEGLIHSQKQRQDILDILWYHTQGYPFAADGVGAAARSDLEKHGMDDPEREVLWRFDSRLAFHPELHPEYYKD
CFKLVPVEPEKIEEANEGENNSSLHPMSLHGMDEPEKEVLWKFDSRLVPPEPEKVEEANEGENNCLLHPMSQHMGGKW
SKRSVEKANEGENNAACAWLEAQEEEEVGFPRQVPLRPMTYKGALDSHFLKEAREKHPEYYKQRSEILDLWYHTQG
YFPDWMGGKWSSITSSNTAANNADCAWLEAQEEEEEVGFPRMPTYKGAVDLSHFLKEKGLEGLVYSQRQDILDLY
VYHSNLHPMSQHGMDDPEKEVLMWKFDSRLAFHHMARELHPEYYKNCLLHPMSLHGMDDPEKGGLEGLYSQKRQDILD
DLWVYNTQGYFPDWQNYTPGPGRYPLTFGWPADAVDMRAAADGVGAISRDLKEHGAITSNTAT

That string contained all the well-conserved syrings (at 90 %)
And we reached the same coverage level (62 %) that the one
obtained by Nickle et al. in “Coping with viral diversity in HIV
vaccine design”.
We did a similar analysis for the Gag protein, and we got also the
same coverage level (82 %) that the one obtained by Nickle et al.
When the number of target strings and the value of $\lambda$ is relatively small, it is possible to solve the problem of finding a shortest $\lambda$-superstring by using integer programming.
Integer programming

\[
\begin{align*}
\text{mín} & \quad \sum_{i,j} w_{ij} x_{ij} \\
\text{s.t.} & \quad y_{s^*} = 1 \\
& \quad \sum_{i \in V : i \neq j} x_{ij} = y_j \quad \forall j \in V \\
& \quad \sum_{j \in V : j \neq i} x_{ij} = y_i \quad \forall i \in V \\
& \quad \sum_{i \in X_j} y_i \geq \lambda \quad \forall j \in \{1, \ldots, n\} \\
& \quad 0 \leq x_{ij} \leq 1, \quad x_{ij} \text{ integer} \\
& \quad 0 \leq y_i \leq 1, \quad y_i \text{ integer}
\end{align*}
\]
We implemented the algorithm by using the CPLEX software.
We implemented the algorithm by using the CPLEX software and we got, for the same 166 strings of the Nef protein used in the Hill-climbing algorithm and the epitopes of Nef appearing in the HIV Molecular Immunology Database, the following 20-superstring:
We implemented the algorithm by using the CPLEX software and we got, for the same 166 strings of the Nef protein used in the Hill-climbing algorithm and the epitopes of Nef appearing in the HIV Molecular Immunology Database, the following 20-superstring:

```
FLKEKGGLDGLWLEAQEEEEEVGFVPVRPVQVPLRPMTYKAAVDLHSFLKEKGGLEGLIYSQKRQDILDLDWVYHTQGYFDP
WQNYTPGPGRYTPGVRYPCTLFWCKLVHPVKFWDSRLAFHHVARELHPEY
```
We implemented the algorithm by using the CPLEX software and we got, for the same 166 strings of the Nef protein used in the Hill-climbing algorithm and the epitopes of Nef appearing in the HIV Molecular Immunology Database, the following 20-superstring:

```
FLKEKGGGLDGLWLEAQEEEEVGFVPVRPQVPLRPMTYKAAVDLHFLKEKGGLEGLIYSQKRQDILDLDWVYHTQGYFPD
WQNYTPGPIRYTPPGVRYPFLWGFWKVFDSRLAFHHVARELHPEY
```

which has much shorter length (131).
Thank you very much for your attention!