Reichman University Efi Arazi School of Computer Science

Reichman University

Efi Arazi School of Computer Science
M.Sc. program

Developing a flexible and comprehensive software app for design of synthetic DNA sequences without unwanted patterns

by

Hadar Pur

Final Project proposal
M.Sc. degree, School of Computer Science Reichman University (The Interdisciplinary Center, Herzliya)

June 2023

Table of Content

Developing a flexible and comprehensive software app for design of synthetic DNA sequences without unwanted patterns

Pur Hadar

June 2023

1 Introduction and Background

Artificial DNA sequencing has become a central tool in molecular biology, offering insights into the function and behavior of living organisms. This process involves determining the precise order of nucleotides (A, C, T, and G) in a DNA molecule, which has been made faster, cheaper, and more accurate by advancements in sequencing technologies.

Designing artificial DNA sequences involves considering several constraints, including the desired function or purpose of the sequence, its stability and efficiency, and the potential for unintended effects. Unwanted binding interactions between the DNA sequence and other molecules, like proteins, can lead to unintended consequences such as the activation of oncogenes or suppression of tumor suppressor genes. Thus, designing artificial DNA sequences requires careful consideration of these constraints to ensure that the synthesized sequence functions as intended and has minimal unwanted interactions with other molecules.

On the positive side, the design involves specifying an optimal target sequence and determining how local changes to this sequence may impact its functionality. On the negative side, it involves avoiding any unwanted binding of proteins to the DNA sequence by eliminating certain short sequence patterns (motifs) that bind proteins with high likelihood. While several design tools exist to address this issue, there is no formal solution to the problem yet.

In a recent study [1] [2], Zehavit Leibovich and Ilan Gronau examined the problem of eliminating sequence patterns that can lead to unwanted binding interactions. The thesis suggests several algorithms for eliminating these unwanted patterns with minimal interference to the desired function of the synthesized DNA, but all of them except the latest one, includes limitations that don't fit to our goal. In Zehavit's article, she proposes an efficient theoretical algorithm that eliminates the unwanted sequence pattern using a scoring scheme for each codon(example below).

Zehavit's thesis describes a simple example to motivate the main algorithm. In Zehavit's example, $S = TACACA$, where the first triplet (TAC) is a codon for Tyrosine, and the second triplet (ACA) is a codon for Threonine.

The scoring scheme will looks as follow:

	Tyrosine (Tyr)			Threonine (Thr)		
S = . . .	T	A	C	A	C	A
cost(I,A) = . . .	w	0	∞	0	w	0
cost(I,T) = . . .	0	w	x	w	w	x1
cost(I,C) = . . .	w	w	0	w	0	x2
cost(I,G) = . . .	w	w	∞	w	w	x3

Table 1

Sequence S contains an unwanted pattern TACA, which is associated with the binding of BlaI repressor. The cost table, cost(i,b), is defined for every position i=1..6 and every base b in {A...G}. The value of cost(i,b) is determined based on a few basic guidelines:

- If $b = S_i$ (the i'th base in the target sequence S), then $cost(b, i) = 0$
- If substituting S_i with b results in another codon of the same amino acid, then $cost(b, i) = x \geq 0$.
- If substituting S_i with b results in codon of different amino acid codon, then $cost(b, i) = w \gg x$.
- If substituting S_i with b results in codon of a stop codon, then $cost(b, i) \to \infty$.

The genetic code (Figure 1) is used to define a biologically-motivated cost table for each codon and our main goal is to minimize the cost of unwanted pattern TACA elimination using this approach.

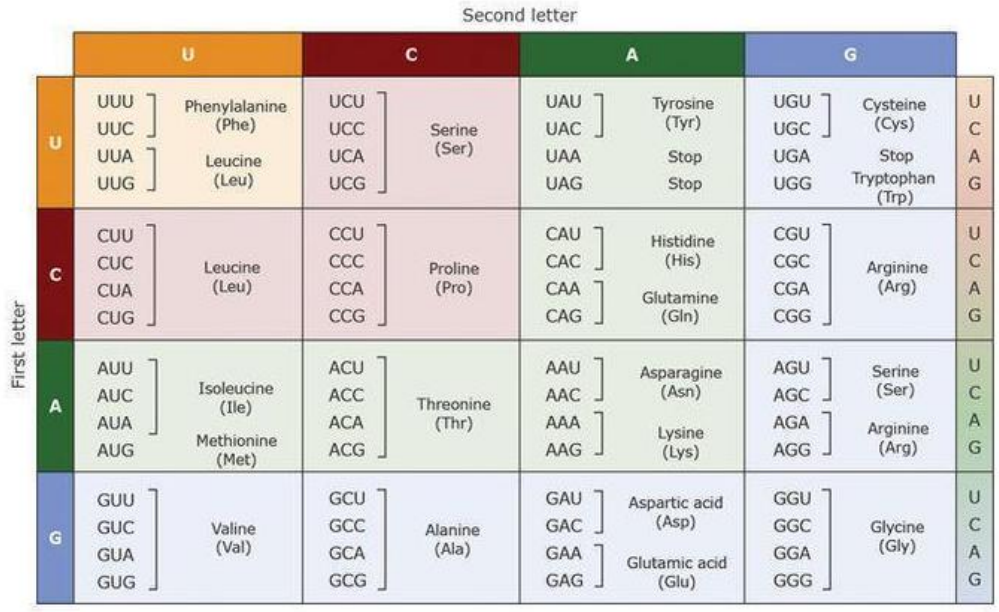

Figure 1: The Genetic code table.

The genetic code is a set of rules used by all cells to know how to make a specific protein. Each gene's code contains 4 bases of DNA: Adenine (A), Cytosine (C), Guanine (G) and Thymine (T). These bases create codons of three letters that specify all amino acids.Figure source: https://www.sciencelearn.org.nz/images/2333-the-genetic-code

The elimination approach that suggested in the article is true if we want to change only one base for each codon and for such case there is a theoretical algorithm(Algorithm 1) that based on computing an FSM, $KMP_{\wp}$, which generates all and only sequences that do not contain unwanted binding sites, and then using a dynamic programming algorithm to compute a min-cost path in this FSM.

Algorithm 1 Computing a min-cost $\mathcal{P}$-clean sequence of length n

Compute an FSM(V, f) over alphabet Σ that generates all and only $\mathcal{P}$-clean sequences.
Initialize: $A[0, v] = \begin{cases} 0 & \text{if } v = v_{init} \\ \infty & \text{otherwise} \end{cases}$
for $i = 1...n$ and $v \in V$ **do**:
    $A^*[i, v] = (u^*, \sigma^*) = \underset{u,\sigma: f(u,\sigma)=v}{\text{argmin}} \{A[i-1, u] + cost(i, \sigma)\}$
    $A[i, v] = A[i-1, u^*] + cost(i, \sigma^*)$
end for
$v_n = \underset{v \in V}{\text{argmin}} A[n, v]$
for $i = 1...n$ **do**:
    $(v_{i-1}, S_i) = A^*[i, v_i]$
end for

2 Project objectives

Designing a synthetic DNA molecule involves meeting various objectives, some of which may be in conflict with each other. One essential objective is to eliminate undesirable protein binding sites that can impede the molecule's intended function. While most design tools can handle this requirement, they lack a systematic approach that ensures the removal of all unwanted sites when a feasible solution is available. Additionally, the algorithms used by these tools (when published) are frequently unsophisticated and inefficient.

Our main objective is to develop a synthetic DNA tool that is specifically designed for biological users, utilizing Zehavit's algorithm (as shown in Algorithm 1) with minor modifications. This tool will feature a user-friendly interface that allows biologists to easily communicate their biological preferences. Currently, to use Zehavit's algorithm, researchers are required to provide the DNA sequence along with all the necessary cost tables and unwanted patterns, which can be a cumbersome and time-consuming process.

The primary goal of this tool is to provide an efficient and effective method for designing artificial DNA sequences that minimize unwanted binding interactions between the DNA and other molecules, such as proteins. Zehavit's algorithm employs a scoring scheme to identify and eliminate unwanted sequence patterns that may lead to unwanted binding interactions. By incorporating this algorithm into a user-friendly interface, our tool will provide researchers and scientists with a reliable and easy-to-use platform for designing optimized synthetic DNA sequences for their specific needs.

Looking ahead, there are several challenges that we must address. One such challenge is determining what is considered "easy" or "difficult" for biological users, and identifying their specific needs. This is crucial to ensure that our synthetic DNA tool is user-friendly and meets the needs of the scientific community. Another challenge is accurately defining the cost function for coding regions. The cost function is a mathematical function that assigns a numerical value to a DNA sequence based on specific criteria. To ensure efficient translation of synthetic DNA sequences into functional proteins, the cost function for coding regions should consider factors such as codon usage bias, codon optimality, and secondary structure. Therefore, accurately defining the cost function is crucial to optimizing synthetic DNA sequences for efficient protein expression.

By addressing these challenges, we can ensure that our synthetic DNA tool is effective, user-friendly, and meets the specific needs of the scientific community.

3 Project outline

This tool will be implemented in Python for the algorithms side using packages such as:

- pyJASPAR for Motif usage
- Biopython for biological computation
- Tkinter/PyGUI/PyQt5 for the Gui usage

The main idea is to create a capability to use this tool by executing it from the command line and from the GUI.

The flow defined below:

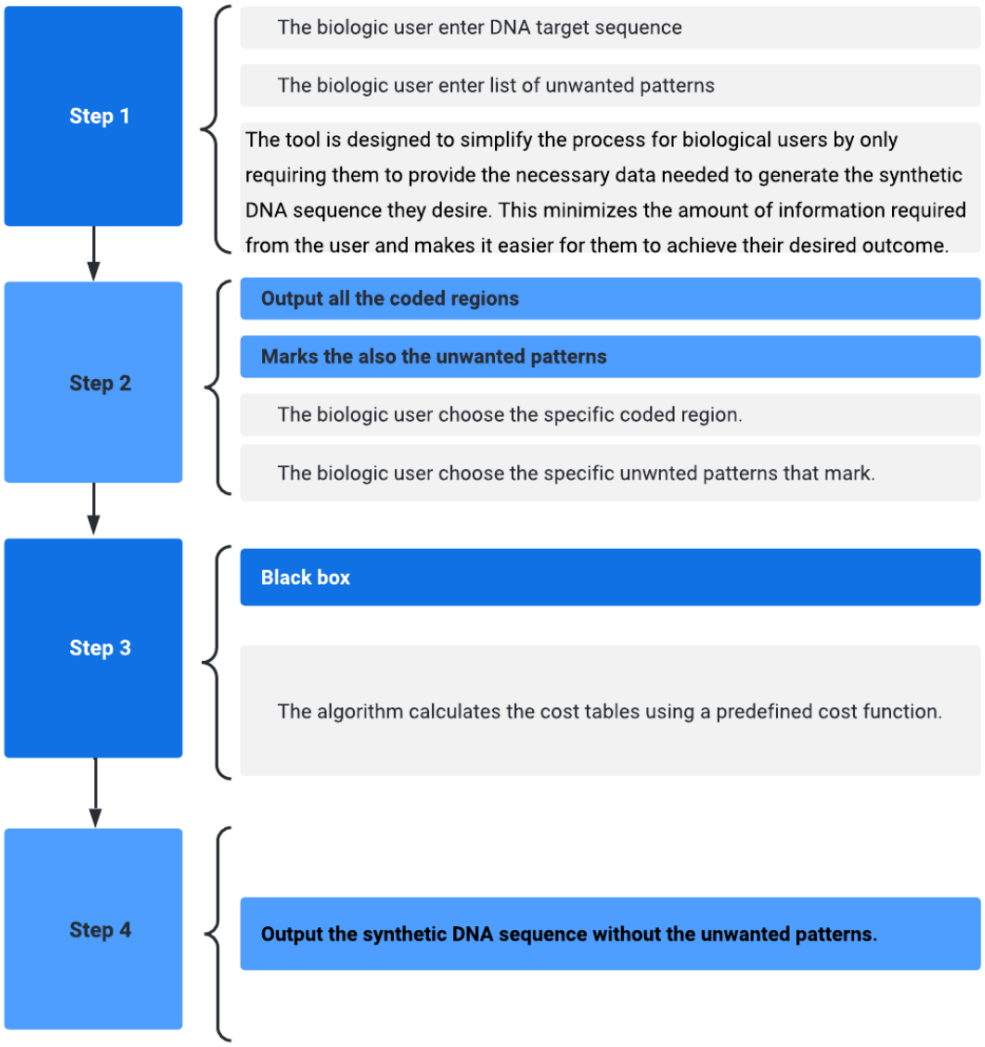

Figure 2

4 Preliminary results

4.1 Cost function for coding regions:

As an initial step of the project, we took the example above and used the basic ideas behind the cost table mentioned (as listed above) to define a cost table for each of the 64 possible codons. A detailed summary for all 64 codons is available in the Appendices.

There are 2 Amino acids that are coded by only one codon. For example, Tryptophan:

	Tryptophan (Trp)		
S =	T	G	G
cost(I,A) =	w	∞	∞
cost(I,T) =	0	w	w
cost(I,C) =	w	w	w
cost(I,G) =	w	0	0

Table 2: Cost table for Tryptophan

There are 9 Amino acids that are coded by 2 codons. For example, Phenylalanine:

	Phenylalanine (Phe)		
S =	T	T	T
cost(I,A) =	w	w	w
cost(I,T) =	0	0	0
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

	Phenylalanine (Phe)		
S =	T	T	C
cost(I,A) =	w	w	w
cost(I,T) =	0	0	x1
cost(I,C) =	w	w	0
cost(I,G) =	w	w	w

Table 3: Cost tables for Phenylalanine

Phenylalanine can be summarized using a single scoring table. By observing the original tables above:

- The 2 tables above are identical in the first two positions.
- The 3rd position is slightly different, they contain 0 for the "original bases".

For the 3rd base, there are 2 choices for that specific amino acid. To be able group the tables above, there is a need to define a static table known ahead which contains the x's cost, without considering the original base. Therefore, the cost is $x_i,\ 1 \leq i \leq 2$.

The grouping table will be:

	Phenylalanine (Phe)		
S =	T	T	T/C
cost(I,A) =	w	w	w
cost(I,T) =	0	0	x1
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

Table 4: Grouped cost table for Phenylalanine

There are 13 Amino acids that are coded by more than 2 codons. For example, Alanine:

	Alanine (Ala)		
S =	G	C	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	0
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	A
cost(I,A) =	w	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	0

Table 5: Cost tables for Alanine

The grouping table will be:

	Phenylalanine (Phe)		
S =	G	C	*
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

Table 6: Grouped cost table for Alanine

Note that for some AAs, some of the substitutions result in a stop codon. For example, examine the two codons for Glutamic acid:

	Glutamic acid (Glu)		
S =	G	A	A
cost(I,A) =	w	0	0
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	x2

	Glutamic acid (Glu)		
S =	G	A	G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	0

Table 7: Cost tables for Glutamic acid

There are some codons (start codon and three stop codons) that are responsible for the translation into protein sequence. If some base changing will reach stop codons it can affect DNA translation, while start codon might do nothing in the middle of translation. Thus, we will define that if some base reach to start codon ATG will costs w, and to stop codons TAA, TAG *and* TGA will cost ∞.

The grouping table will looks like:

	Glutamic acid (Glu)		
S =	G	A	A/G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	x2

Table 8: Grouped cost table for Glutamic acid

Another limitation we found with this basic approach to cost tables has to do with amino acids that are not symmetric and cannot be grouped in one table. The non-symmetric issue will be expressed with different tables who can reach a stop codon by base changer (for at most 4 codons per amino acid) or amino acids that contain more than 4 codons, which cannot be grouped to one table. Those amino acids are Glycine, Arginine, Serine and Leucine. If we will observe Arginine for example:

	Arginine (Arg)		
S =	C	G	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	0	w	x3
cost(I,G) =	w	0	x4

	Arginine (Arg)		
S =	C	G	A
cost(I,A) =	w	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	0	w	x3
cost(I,G) =	w	0	x4

	Arginine (Arg)		
S =	C	G	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	0	w	0
cost(I,G) =	w	0	x4

	Arginine (Arg)		
S =	C	G	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	0	w	x3
cost(I,G) =	w	0	0

	Arginine (Arg)		
S =	A	G	A
cost(I,A) =	0	w	0
cost(I,T) =	w	w	w
cost(I,C) =	x	w	w
cost(I,G) =	w	0	x6

	Arginine (Arg)		
S =	A	G	G
cost(I,A) =	0	w	x5
cost(I,T) =	w	w	w
cost(I,C) =	x	w	w
cost(I,G) =	w	0	0

Table 9: Cost tables for Arginine

In the Appendices section I provide a complete list of cost tables for all 64 codons, including symmetry grouping when possible. After writing up these tables, we noticed that these cost tables assume that we can achieve an optimal elimination set if we are allowed to substitute only one base per codon. Let's consider a few examples that demonstrate the potential limitations of this assumption.

Let's consider a case where codon GAA for Glutamic acid is substituted to codon TTA for Leucine. This substitution involves two base substitutions (in positions 1, 2). According to the cost table for Glutamic acid (see below), this costs ∞. However, the true biological cost of this substitution should be w because we are changing the amino acid.

	Glutamic acid (Glu)		
S =	G	A	A/G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	x2

Table 10: Grouped cost table for Glutamic acid

Also, let's consider a case where codon GCA for Alanine is substituted to codon TAA for Stop codon. This substitution involves two base substitutions (in positions 1, 2). According to the cost table for Glutamic acid (see below), this costs $2w$. However, the true biological cost of this substitution should be ∞ because we are reaching a stop codon which will stop the DNA translation.

	Phenylalanine (Phe)		
S =	G	C	*
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

Table 11: Grouped cost table for Alanine

All the issues above lead us to understand that what Zehavit's thesis offers is much more complicated as it seems at the beginning. For cases that we described above, we will need to think of a conditional table solution which obtains our goals.

5 Work plan

Start date	End date	Task	Task mode
10/12/2021	05/06/2023	The project proposal encompasses thorough research of the thesis that forms the foundation of the project. It involves delving deep into the realm of synthetic DNA, meticulously examining the algorithm upon which our project will rely, and presenting an initial design outlining the functionality of the tool.	Project proposal
14/05/2023	30/07/2023	The prototype will provide biologists with a comprehensive range of features, enabling them to conveniently synthesize DNA sequences directly from the terminal. It will empower biologists by facilitating the identification and highlighting of coding regions within a given sequence, giving them the freedom to selectively exclude specific regions as needed. By utilizing the selected coding regions, we will synthesize the sequence using predefined cost tables, along with the optimal elimination of unwanted binding sites algorithm proposed in Zehavit's thesis (Algorithm 1).	Prototype
1/08/2023	30/09/2023	The shell execution will encompass a comprehensive implementation of the algorithm, utilizing the essential data provided by the biologist. Its purpose will be to generate the desired DNA sequence, free from any undesired binding sites.	Shell execution
30/09/2023	30/11/2023	The GUI execution will seamlessly integrate all the functionalities of the shell execution within the graphical user interface (GUI). This ensures a smooth and user-friendly experience, allowing biologists to access and utilize the complete set of features provided by the shell execution directly through the GUI interface.	GUI execution
30/11/2023	30/01/2024	The report encompasses a comprehensive overview of the project, incorporating its objectives, goals, design aspects, results, and tool designs. It also addresses the challenges encountered throughout the project, both in terms of setbacks and achievements. Additionally, the report will showcase the final product, presenting it as the culmination of the project's efforts.	Project final report submission

References

[1] Zehavit Leibovich. "Eliminating unwanted patterns with minimal interferences". In: (June 2021). URL: https://www.runi.ac.il/media/iawjvg5e/zehavit-thesis-final.pdf.

[2] Zehavit Leibovich and Ilan Gronau. "Optimal Design of Synthetic DNA Sequences Without Unwanted Binding Sites". In: *Journal of Computational Biology* 29.9 (Sept. 2022), pp. 663–670. URL: https://www.liebertpub.com/doi/10.1089/cmb.2021.0417.

A Appendices

A.1 Symmetric amino acids scoring scheme

There are 16 amino acids that their cost tables are symmetric and can be summarized to a single table without any condition.

2 of the them coded by only 1 codon, those are Tryptophan and Methionine

A.1.1 Tryptophan (Trp)

	Tryptophan (Trp)		
S =	T	G	G
cost(I,A) =	w	∞	∞
cost(I,T) =	0	w	w
cost(I,C) =	w	w	w
cost(I,G) =	w	0	0

A.1.2 Methionine (Met)

	Methionine (Met)		
S =	A	T	G
cost(I,A) =	0	∞	∞
cost(I,T) =	∞	0	∞
cost(I,C) =	∞	∞	∞
cost(I,G) =	∞	∞	0

The other 14 amino acids coded by multiple codons that differ in their 3rd base.

A.1.3 Phenylalanine (Phe)

	Phenylalanine (Phe)		
S =	T	T	T
cost(I,A) =	w	w	w
cost(I,T) =	0	0	0
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

	Phenylalanine (Phe)		
S =	T	T	C
cost(I,A) =	w	w	w
cost(I,T) =	0	0	x1
cost(I,C) =	w	w	0
cost(I,G) =	w	w	w

It can be grouping to one table:

	Phenylalanine (Phe)		
S =	T	T	T/C
cost(I,A) =	w	w	w
cost(I,T) =	0	0	x1
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

A.1.4 Histidine (His)

	Histidine (His)		
S =	C	A	T
cost(I,A) =	w	0	w
cost(I,T) =	w	w	0
cost(I,C) =	0	w	x2
cost(I,G) =	w	w	w

	Histidine (His)		
S =	C	A	C
cost(I,A) =	w	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	0	w	0
cost(I,G) =	w	w	w

It can be grouping to one table:

	Histidine (His)		
S =	C	A	T/C
cost(I,A) =	w	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	0	w	x2
cost(I,G) =	w	w	w

A.1.5 Asparagine (Asn)

	Asparagine (Asn)		
S =	A	A	T
cost(I,A) =	0	0	w
cost(I,T) =	w	w	0
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

	Asparagine (Asn)		
S =	A	A	C
cost(I,A) =	0	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	w	w	0
cost(I,G) =	w	w	w

It can be grouping to one table:

	Asparagine (Asn)		
S =	A	A	T/C
cost(I,A) =	0	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	w

A.1.6 Aspartic acid (Asp)

	Aspartic acid (Asp)		
S =	G	A	T
cost(I,A) =	w	0	w
cost(I,T) =	w	w	0
cost(I,C) =	w	w	x2
cost(I,G) =	0	w	w

	Aspartic acid (Asp)		
S =	G	A	C
cost(I,A) =	w	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	w	w	0
cost(I,G) =	0	w	w

It can be grouping to one table:

	Aspartic acid (Asp)		
S =	G	A	T/C
cost(I,A) =	w	0	w
cost(I,T) =	w	w	x1
cost(I,C) =	w	w	x2
cost(I,G) =	0	w	w

A.1.7 Tyrosine (Tyr)

	Tyrosine (Tyr)		
S =	T	A	T
cost(I,A) =	w	0	∞
cost(I,T) =	0	w	0
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	∞

	Tyrosine (Tyr)		
S =	T	A	C
cost(I,A) =	w	0	∞
cost(I,T) =	0	w	x1
cost(I,C) =	w	w	0
cost(I,G) =	w	w	∞

In this case, if we replace the 3rd base with C or G it will produce a stop codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Tyrosine (Tyr)		
S =	T	A	T/C
cost(I,A) =	w	0	∞
cost(I,T) =	0	w	x1
cost(I,C) =	w	w	x2
cost(I,G) =	w	w	∞

A.1.8 Glutamine (Gln)

	Glutamine (Gln)		
S =	C	A	A
cost(I,A) =	w	0	0
cost(I,T) =	∞	w	w
cost(I,C) =	0	w	w
cost(I,G) =	w	w	x2

	Glutamine (Gln)		
S =	C	A	G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	0	w	w
cost(I,G) =	w	w	0

In this case, if we replace 1st base C with T it will produce a stop codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Glutamine (Gln)		
S =	C	A	A/G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	0	w	w
cost(I,G) =	w	w	x2

A.1.9 Lysine (Lys)

	Lysine (Lys)		
S =	A	A	A
cost(I,A) =	0	0	0
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	w	w	x2

	Lysine (Lys)		
S =	A	A	G
cost(I,A) =	0	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	w	w	0

In this case, if we replace the 1st base A with T it will produce a stop codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Lysine (Lys)		
S =	A	A	A/G
cost(I,A) =	0	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	w	w	x2

A.1.10 Glutamic acid (Glu)

	Glutamic acid (Glu)		
S =	G	A	A
cost(I,A) =	w	0	0
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	x2

	Glutamic acid (Glu)		
S =	G	A	G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	0

In this case, if we replace the 1st base G with T it will produce a stop codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Glutamic acid (Glu)		
S =	G	A	A/G
cost(I,A) =	w	0	x1
cost(I,T) =	∞	w	w
cost(I,C) =	w	w	w
cost(I,G) =	0	w	x2

A.1.11 Cysteine (Cys)

	Cysteine (Cys)		
S =	T	G	T
cost(I,A) =	w	w	∞
cost(I,T) =	0	w	0
cost(I,C) =	w	w	x2
cost(I,G) =	w	0	w

	Cysteine (Cys)		
S =	T	G	C
cost(I,A) =	w	w	∞
cost(I,T) =	0	w	x1
cost(I,C) =	w	w	0
cost(I,G) =	w	0	w

In this case, if we replace the 3rd base with A it will produce a start codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Cysteine (Cys)		
S =	T	G	T/C
cost(I,A) =	w	w	∞
cost(I,T) =	0	w	x1
cost(I,C) =	w	w	x2
cost(I,G) =	w	0	w

A.1.12 Isoleucine (Ile)

	Isoleucine (Ile)		
S =	A	T	T
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	w	w	x3
cost(I,G) =	w	0	∞

	Isoleucine (Ile)		
S =	A	T	C
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	w	0
cost(I,G) =	w	0	∞

	Isoleucine (Ile)		
S =	A	T	A
cost(I,A) =	0	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	w	w	x3
cost(I,G) =	w	0	∞

In this case, if we replace the 3rd base with G it will produce a start codon, so we must take it under consideration. Therefore, it can be grouping to one table:

	Isoleucine (Ile)		
S =	A	T	T/C/A
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	w	x3
cost(I,G) =	w	0	∞

A.1.13 Alanine (Ala)

	Alanine (Ala)		
S =	G	C	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	0
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	A
cost(I,A) =	w	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

	Alanine (Ala)		
S =	G	C	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	0

It can be grouping to one table:

	Alanine (Ala)		
S =	G	C	*
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	0	w	x4

A.1.14 Proline (Pro)

	Proline (Pro)		
S =	C	C	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	0	0	x3
cost(I,G) =	w	w	x4

	Proline (Pro)		
S =	C	C	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	0	0	0
cost(I,G) =	w	w	x4

	Proline (Pro)		
S =	C	C	A
cost(I,A) =	w	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	0	0	x3
cost(I,G) =	w	w	x4

	Proline (Pro)		
S =	C	C	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	0	0	x3
cost(I,G) =	w	w	0

It can be grouping to one table:

	Proline (Pro)		
S =	C	C	*
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	0	0	x3
cost(I,G) =	w	w	x4

A.1.15 Threonine (Thr)

	Threonine (Thr)		
S =	A	C	T
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	w	0	x3
cost(I,G) =	w	w	x4

	Threonine (Thr)		
S =	A	C	C
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	0
cost(I,G) =	w	w	x4

	Threonine (Thr)		
S =	A	C	A
cost(I,A) =	0	w	0
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	w	w	x4

	Threonine (Thr)		
S =	A	C	G
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	w	w	0

It can be grouping to one table:

	Threonine (Thr)		
S =	A	C	*
cost(I,A) =	0	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	0	x3
cost(I,G) =	w	w	x4

A.1.16 Valine (val)

	Valine (val)		
S =	G	T	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	0	0
cost(I,C) =	w	w	x3
cost(I,G) =	0	w	x4

	Valine (val)		
S =	G	T	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	0	x2
cost(I,C) =	w	w	0
cost(I,G) =	0	w	x4

	Valine (val)		
S =	G	T	A
cost(I,A) =	w	w	0
cost(I,T) =	w	0	x2
cost(I,C) =	w	w	x3
cost(I,G) =	0	w	x4

	Valine (val)		
S =	G	T	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	0	x2
cost(I,C) =	w	w	x3
cost(I,G) =	0	w	0

It can be grouping to one table:

	Valine (val)		
S =	G	T	*
cost(I,A) =	w	w	x1
cost(I,T) =	w	0	x2
cost(I,C) =	w	w	x3
cost(I,G) =	0	w	x4

A.2 Non-Symmetric amino acids scoring scheme with 4 codons

There is one amino acid that is coded by 4 codons, but the symmetry is violated by 1st or the 2nd base with a case that produces a stop codon.

A.2.1 Glycine (Gly)

	Glycine (Gly)		
S =	G	G	T
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	0
cost(I,C) =	w	w	x3
cost(I,G) =	0	0	x4

	Glycine (Gly)		
S =	G	G	C
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	w	0
cost(I,G) =	0	0	x4

	Glycine (Gly)		
S =	G	G	G
cost(I,A) =	w	w	x1
cost(I,T) =	w	w	x2
cost(I,C) =	w	w	x3
cost(I,G) =	0	0	0

	Glycine (Gly)		
S =	G	G	A
cost(I,A) =	w	w	0
cost(I,T) =	∞	w	x2
cost(I,C) =	w	w	x3
cost(I,G) =	0	0	x4

In this case, if we replace the 1st base G with T it will produce a stop codon if the 3rd base is A, so we must take it under consideration.

A.3 Non-Symmetric amino acids scoring scheme with 6 codons

There are 3 amino acids coded by 6 codons, which cannot be summarized to a single table.

A.3.1 Arginine (Arg)

	Arginine (Arg)		
S =	C	G	T
cost(I,A) =	w	w	x
cost(I,T) =	w	w	0
cost(I,C) =	0	w	x
cost(I,G) =	w	0	x

	Arginine (Arg)		
S =	C	G	A
cost(I,A) =	x	w	0
cost(I,T) =	∞	w	x
cost(I,C) =	0	w	x
cost(I,G) =	w	0	x

	Arginine (Arg)		
S =	C	G	C
cost(I,A) =	w	w	x
cost(I,T) =	w	w	x
cost(I,C) =	0	w	0
cost(I,G) =	w	0	x

	Arginine (Arg)		
S =	C	G	G
cost(I,A) =	x	w	x
cost(I,T) =	w	w	x
cost(I,C) =	0	w	x
cost(I,G) =	w	0	0

	Arginine (Arg)		
S =	A	G	A
cost(I,A) =	0	w	0
cost(I,T) =	∞	w	w
cost(I,C) =	x	w	w
cost(I,G) =	w	0	x

	Arginine (Arg)		
S =	A	G	G
cost(I,A) =	0	w	x
cost(I,T) =	w	w	w
cost(I,C) =	x	w	w
cost(I,G) =	w	0	0

A.3.2 Serine (Ser)

	Serine (Ser)		
S =	T	C	T
cost(I,A) =	w	w	x
cost(I,T) =	0	w	0
cost(I,C) =	w	0	x
cost(I,G) =	w	w	x

	Serine (Ser)		
S =	T	C	C
cost(I,A) =	w	w	x
cost(I,T) =	0	w	x
cost(I,C) =	w	0	0
cost(I,G) =	w	w	x

	Serine (Ser)		
S =	T	C	A
cost(I,A) =	w	w	0
cost(I,T) =	0	w	x
cost(I,C) =	w	0	x
cost(I,G) =	w	∞	x

	Serine (Ser)		
S =	T	C	G
cost(I,A) =	w	w	x
cost(I,T) =	0	w	x
cost(I,C) =	w	0	x
cost(I,G) =	w	w	0

	Serine (Ser)		
S =	A	G	T
cost(I,A) =	0	w	w
cost(I,T) =	w	w	0
cost(I,C) =	w	w	x
cost(I,G) =	w	0	w

	Serine (Ser)		
S =	A	G	C
cost(I,A) =	0	w	w
cost(I,T) =	w	w	x
cost(I,C) =	w	w	0
cost(I,G) =	w	0	w

A.3.3 Leucine (Leu)

	Leucine (Leu)		
S =	C	T	T
cost(I,A) =	w	w	x
cost(I,T) =	w	0	0
cost(I,C) =	0	w	x
cost(I,G) =	w	w	x

	Leucine (Leu)		
S =	C	T	C
cost(I,A) =	w	w	x
cost(I,T) =	w	0	x
cost(I,C) =	0	w	0
cost(I,G) =	w	w	x

	Leucine (Leu)		
S =	C	T	A
cost(I,A) =	w	w	0
cost(I,T) =	x	0	x
cost(I,C) =	0	w	x
cost(I,G) =	w	w	x

	Leucine (Leu)		
S =	C	T	G
cost(I,A) =	w	w	x
cost(I,T) =	x	0	x
cost(I,C) =	0	w	x
cost(I,G) =	w	w	0

	Leucine (Leu)		
S =	T	T	A
cost(I,A) =	w	∞	w
cost(I,T) =	0	0	0
cost(I,C) =	x	w	x
cost(I,G) =	w	∞	w

	Leucine (Leu)		
S =	T	T	G
cost(I,A) =	w	∞	w
cost(I,T) =	0	0	x
cost(I,C) =	x	w	0
cost(I,G) =	w	w	w

A.4 Three stop codons

There are 3 stop codons which are not symmetric and cannot be summarized to any common table.

A.4.1 Stop

	Stop		
S =	T	A	A
cost(I,A) =	∞	0	0
cost(I,T) =	0	∞	∞
cost(I,C) =	∞	∞	∞
cost(I,G) =	∞	x	x

	Stop		
S =	T	A	G
cost(I,A) =	∞	0	x
cost(I,T) =	0	∞	∞
cost(I,C) =	∞	∞	∞
cost(I,G) =	∞	∞	0

	Stop		
S =	T	G	A
cost(I,A) =	∞	x	0
cost(I,T) =	0	∞	∞
cost(I,C) =	∞	∞	∞
cost(I,G) =	∞	0	∞

That issue, cannot be grouped to any table for example, if we take a look at TAG, there is only option to replace the third base, same for TGA with the second base. However, for TAA, there are options for the second and the third bases, because it combines the 2 codons above.