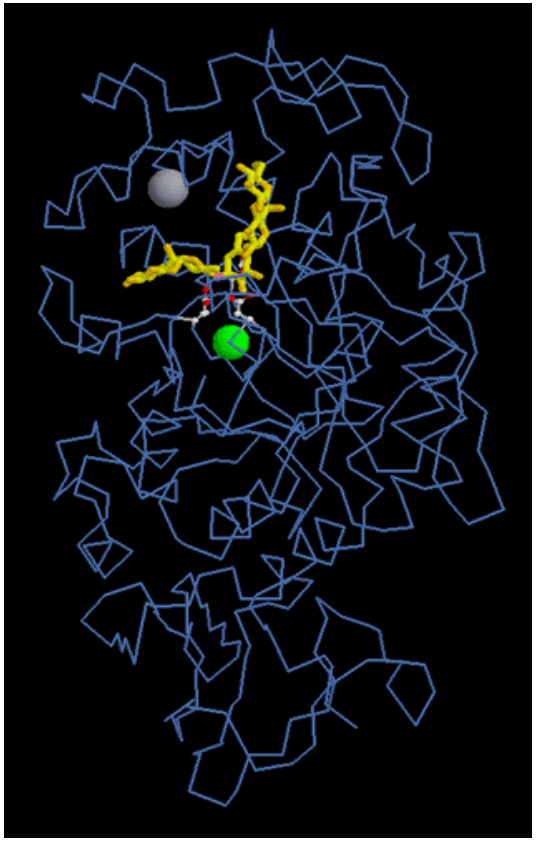
Efficient codon optimization with motif engineering

Anne Condon and Chris Thachuk
University of British Columbia

IWOCA 2011
June 20, 2011

Why synthesize proteins?



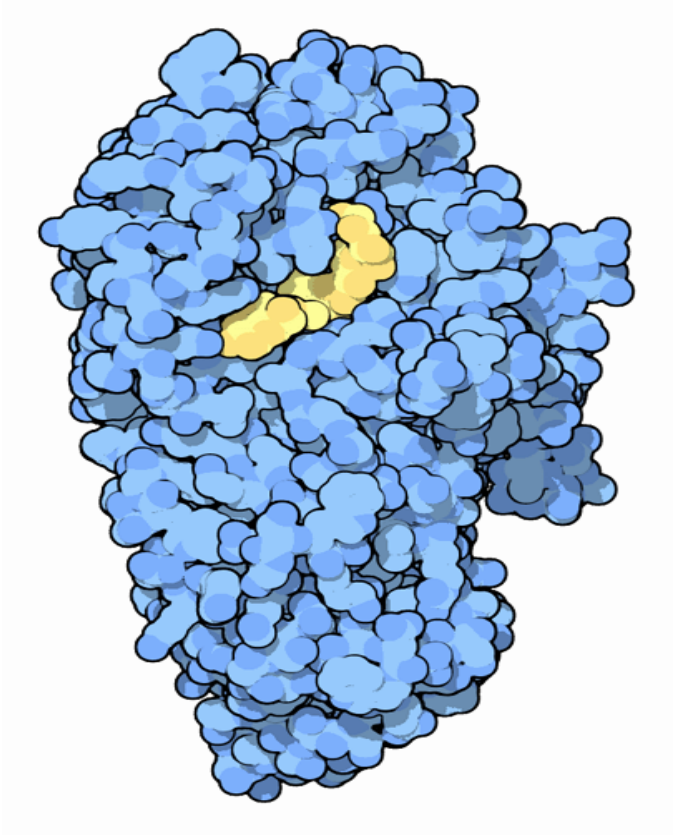


Image credit: Protein Data Bank

Central dogma

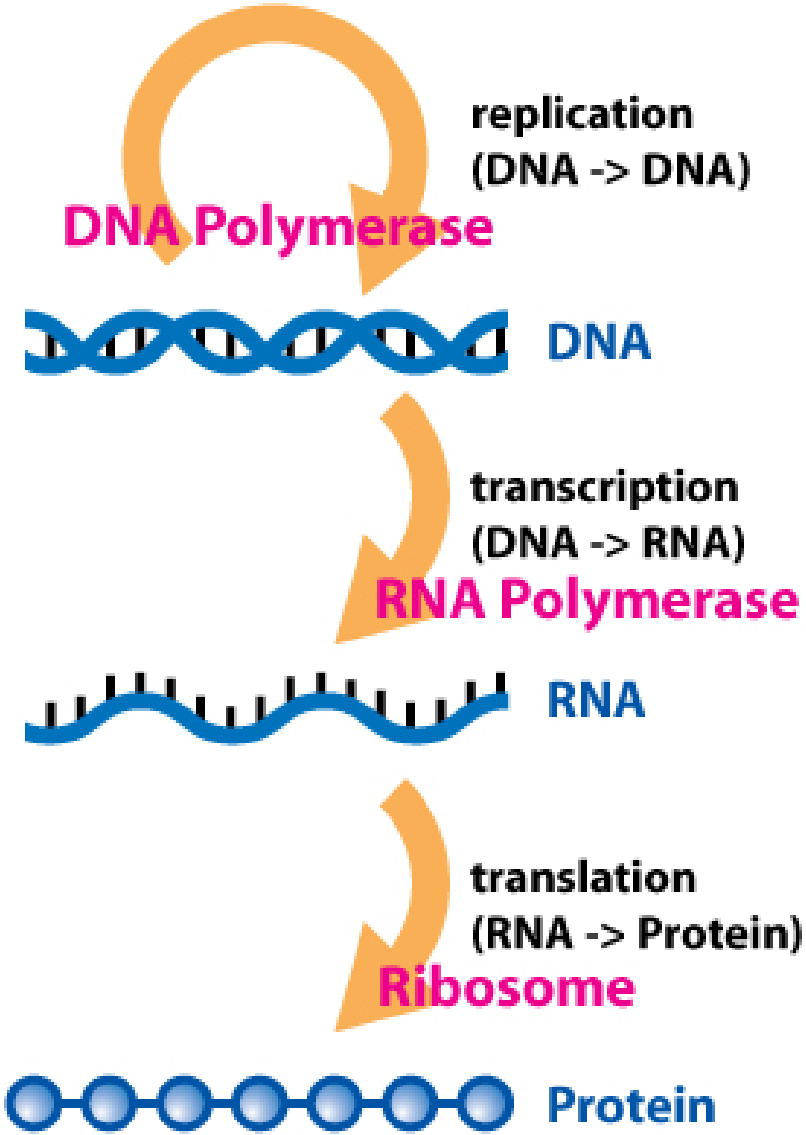
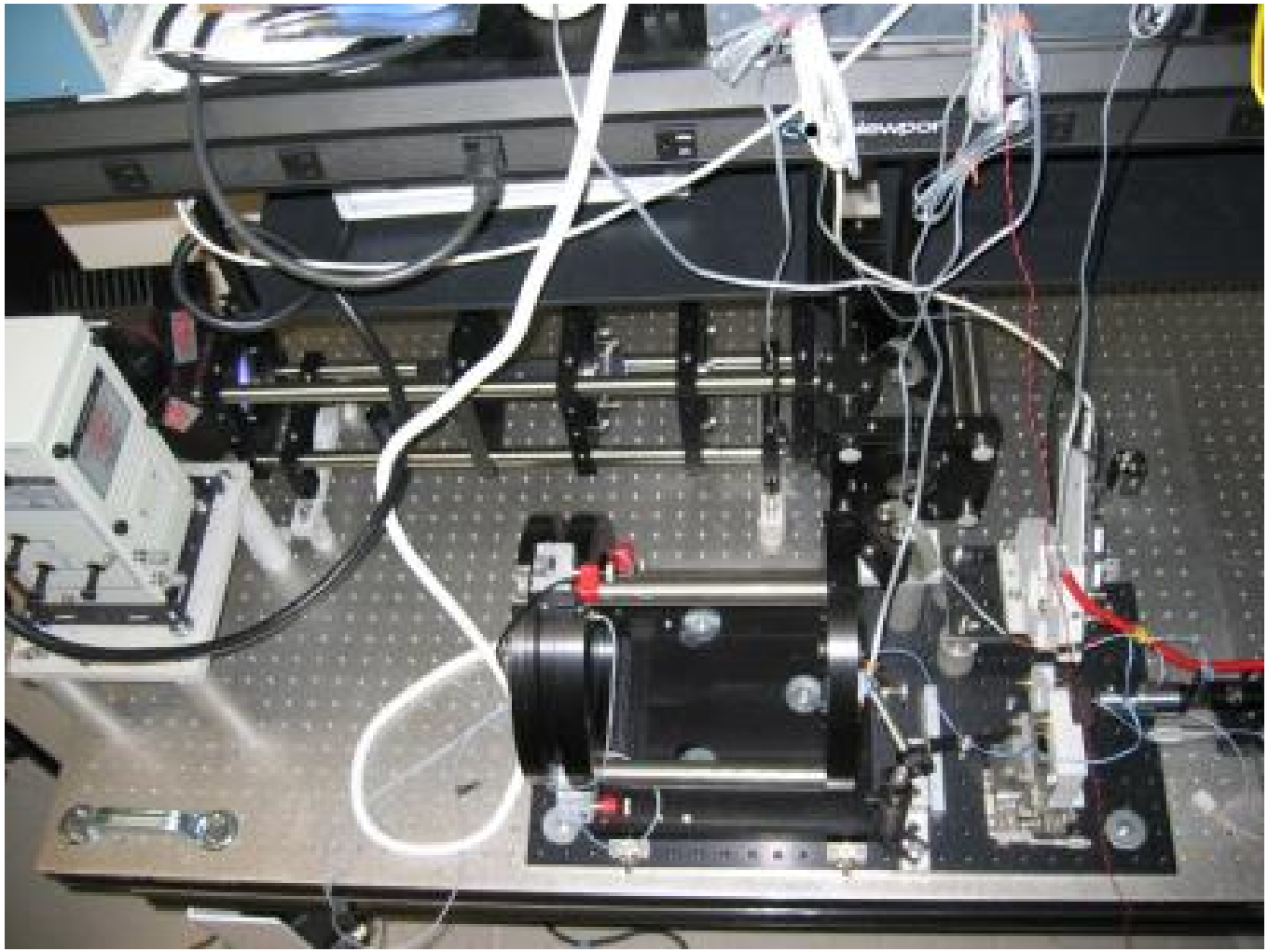
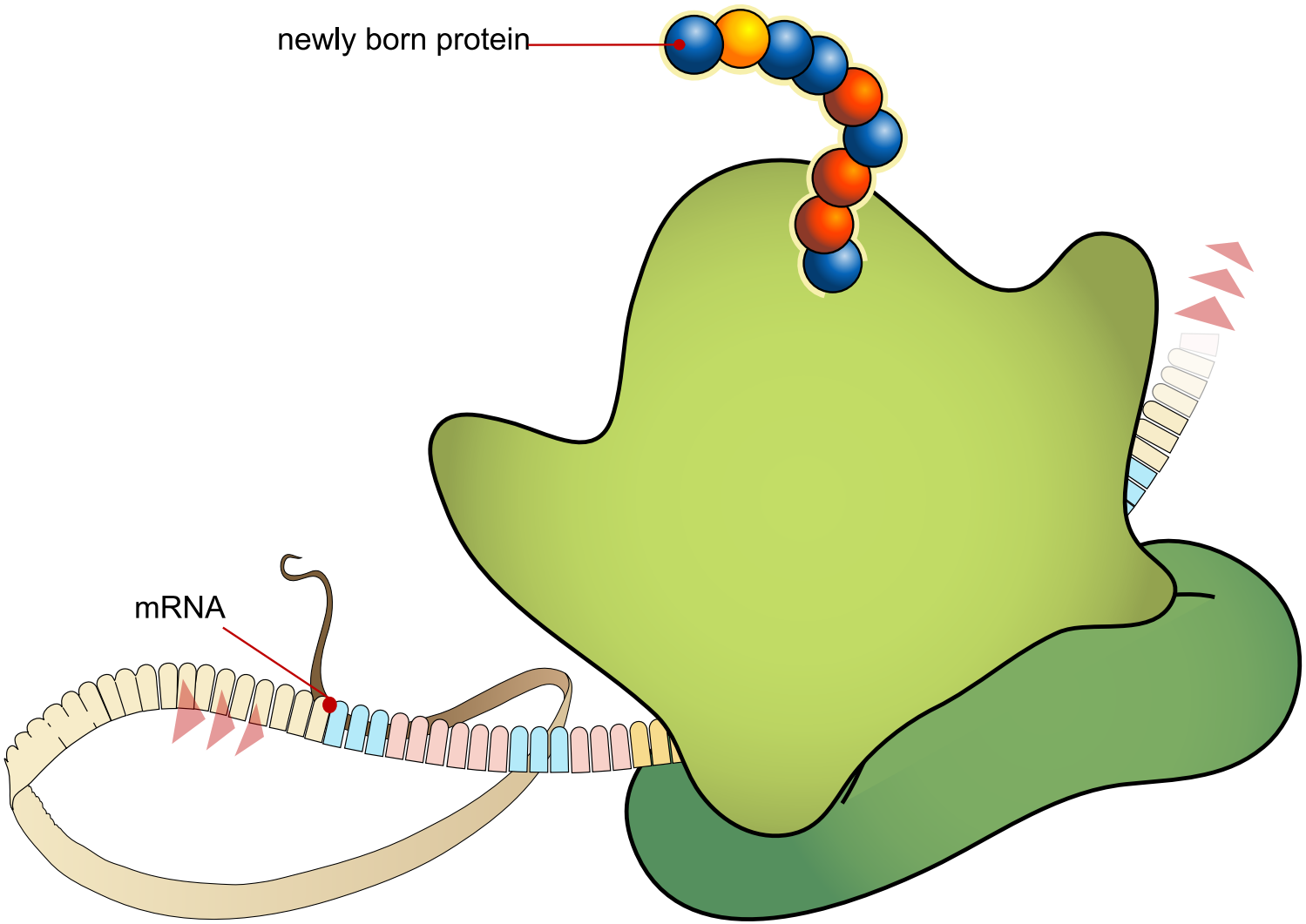
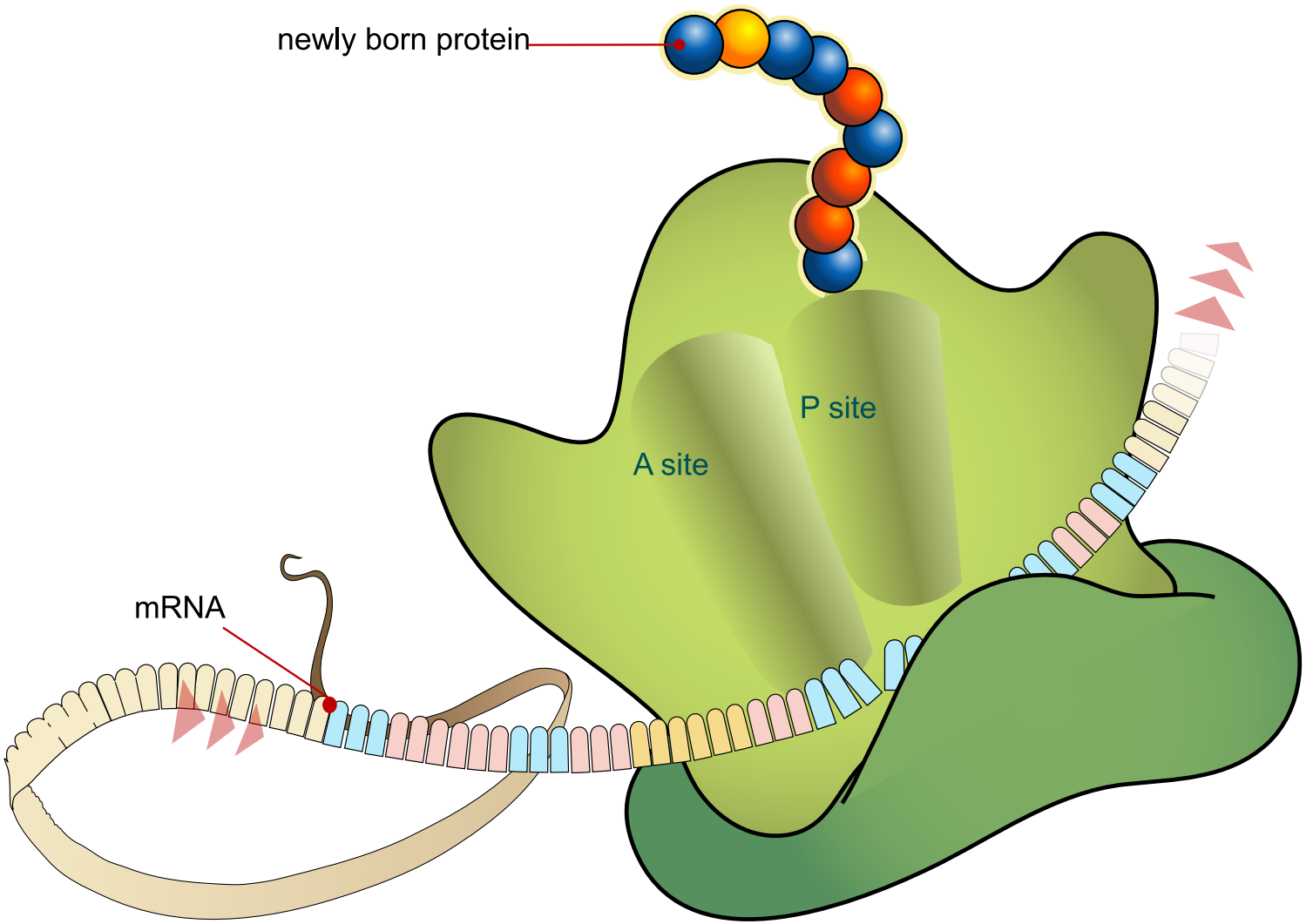


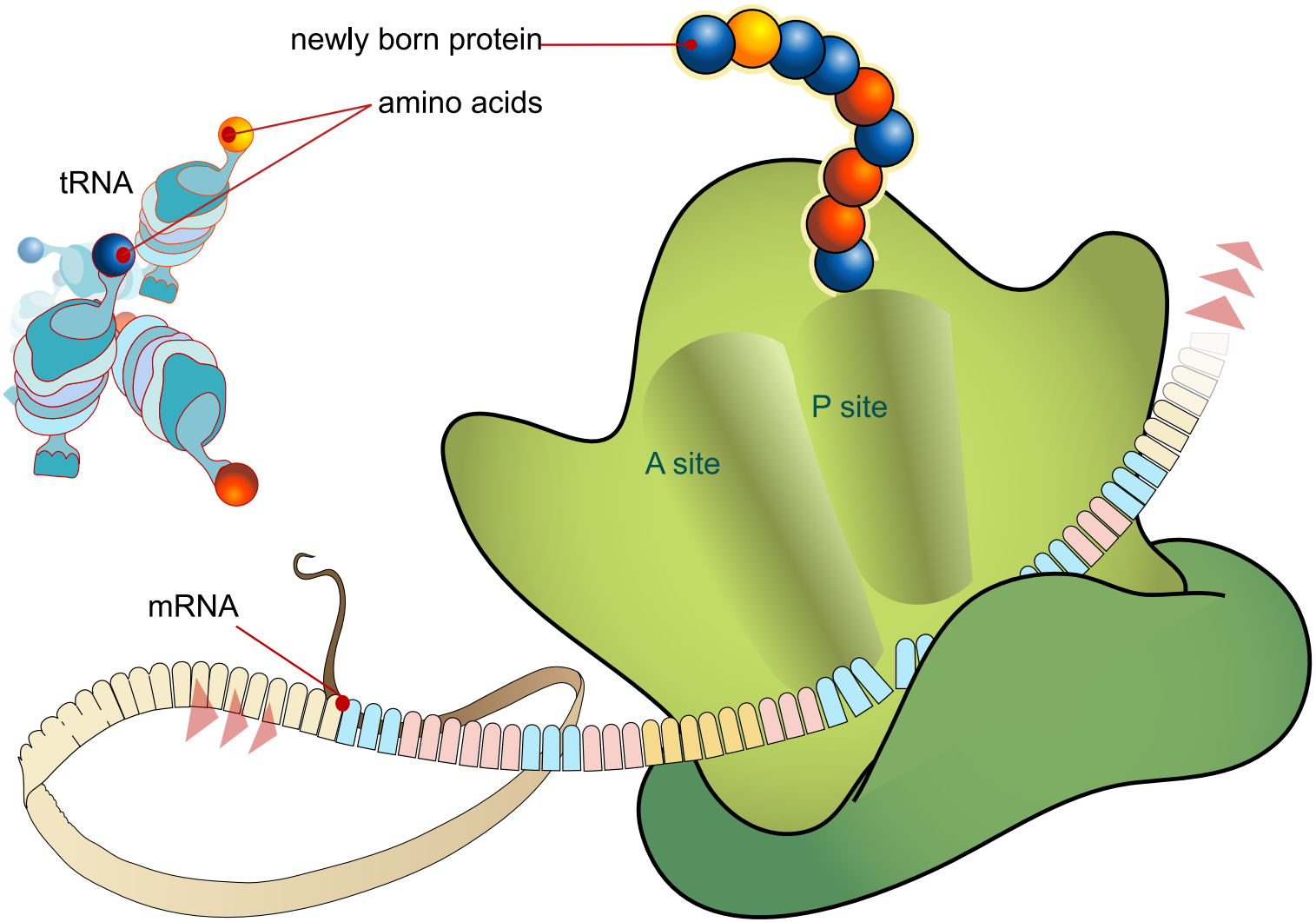
Image credit: Daniel Horspool

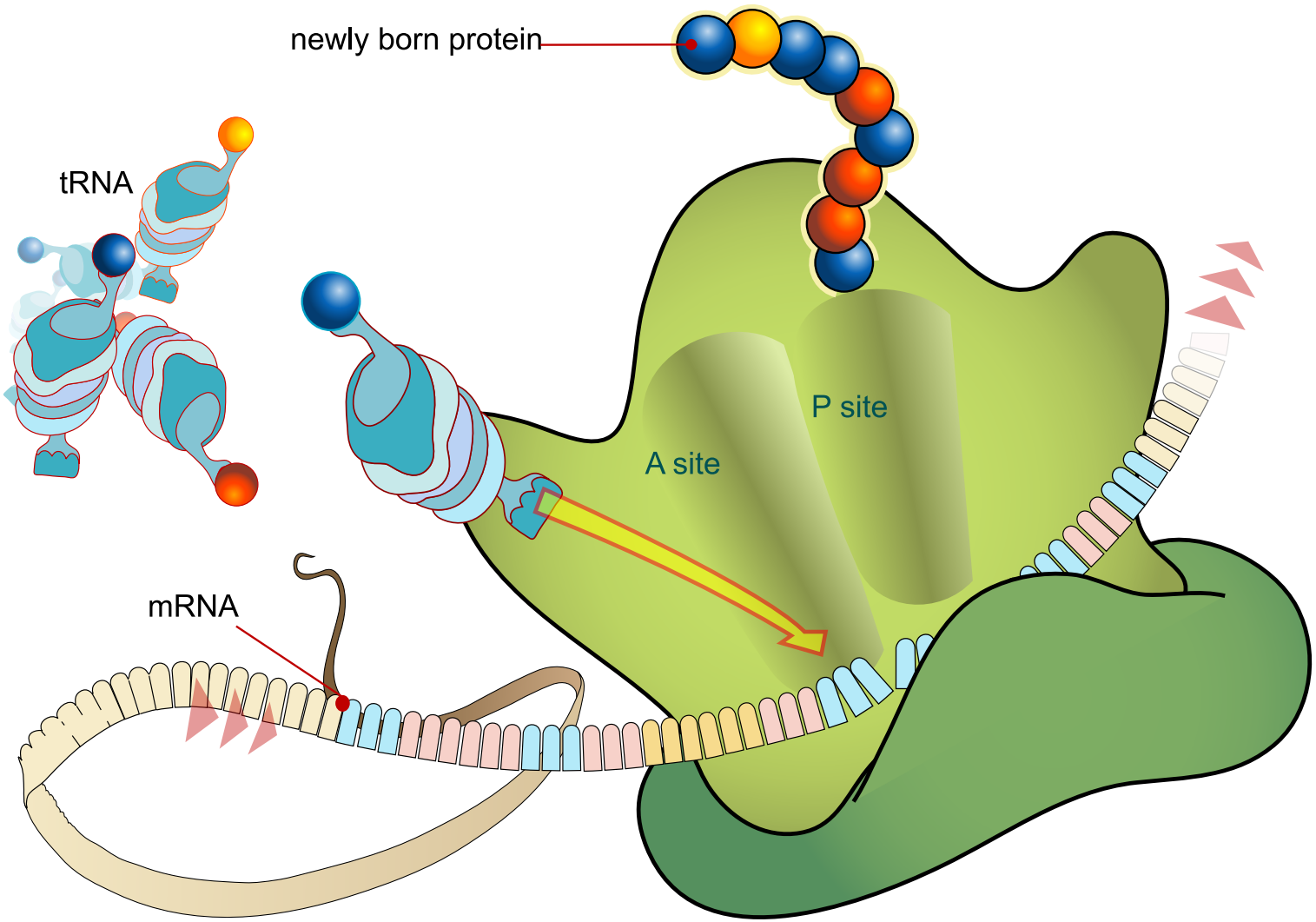
How to synthesize DNA/RNA

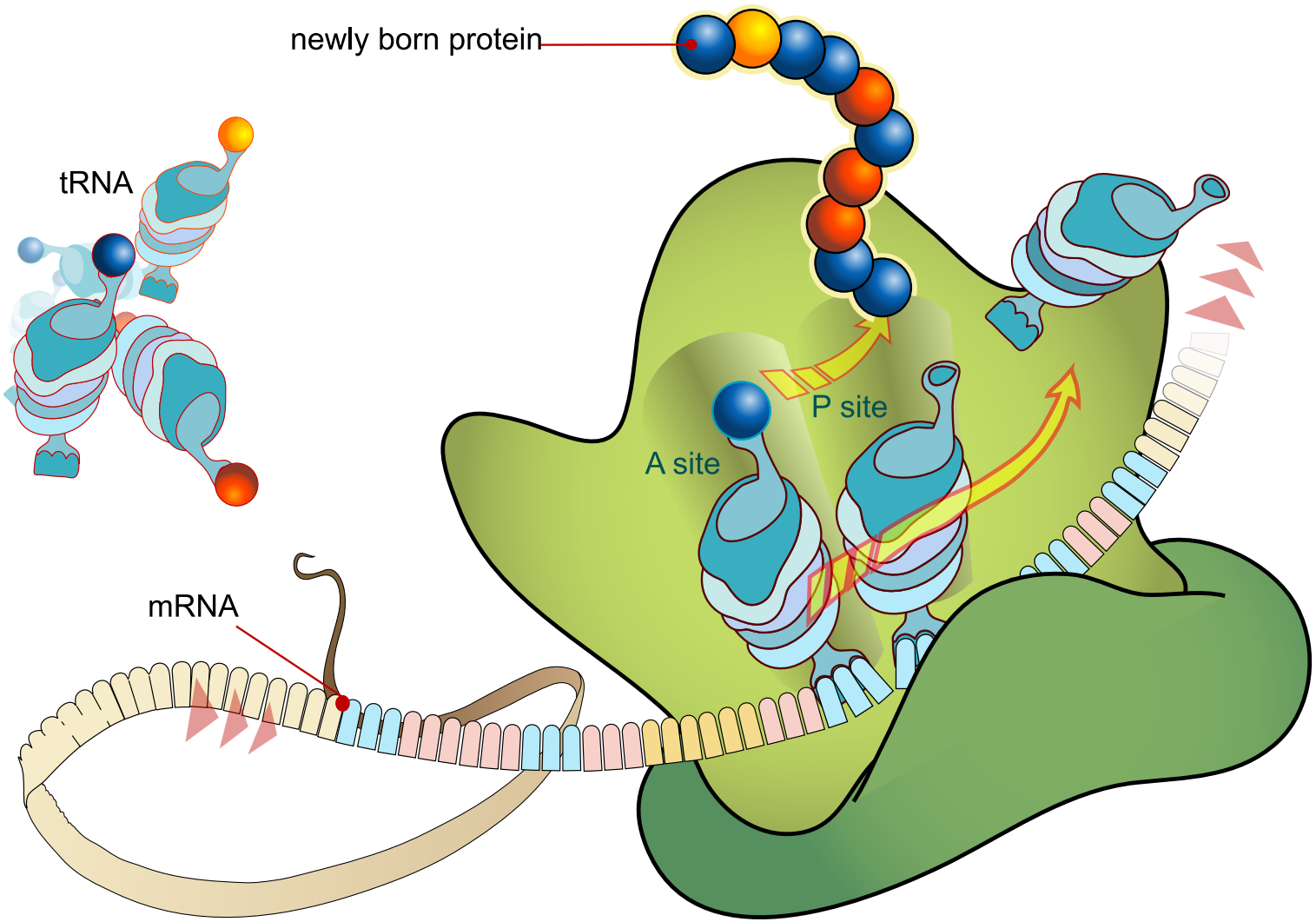


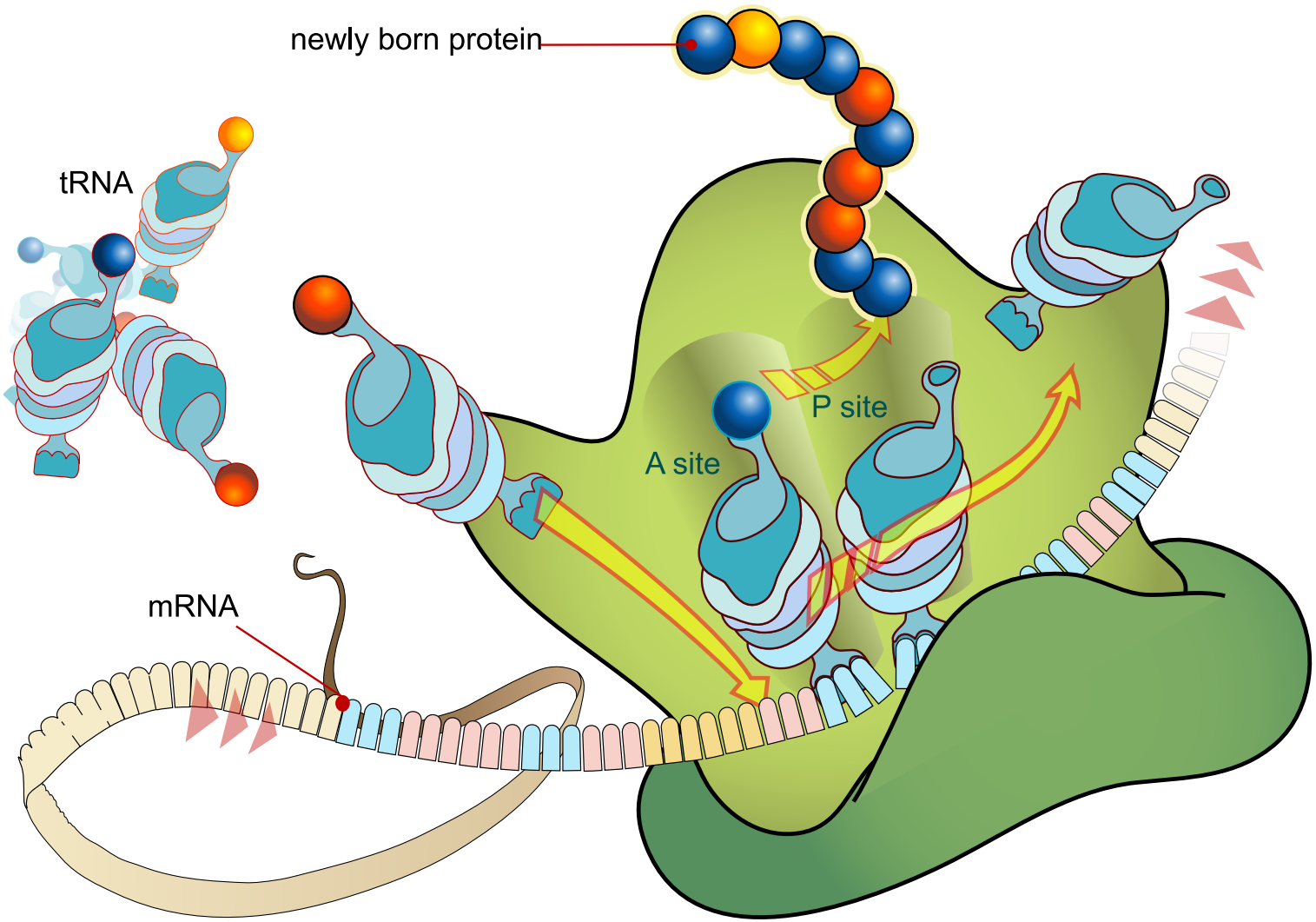

newly born protein
mRNA


newly born protein
P site
A site
mRNA


newly born protein
amino acids
tRNA
P site
A site
mRNA

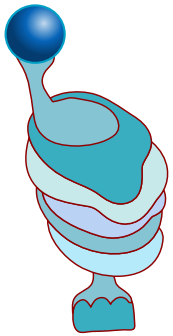

newly born protein
tRNA
P site
A site
mRNA


newly born protein
tRNA
P site
A site
mRNA


newly born protein
tRNA
P site
A site
mRNA

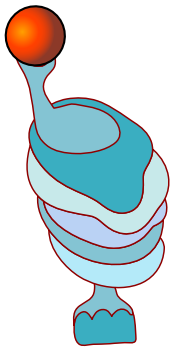
Genetic code is redundant

Tyrosine



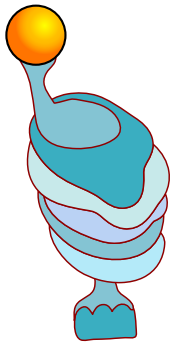
UAU

Alanine



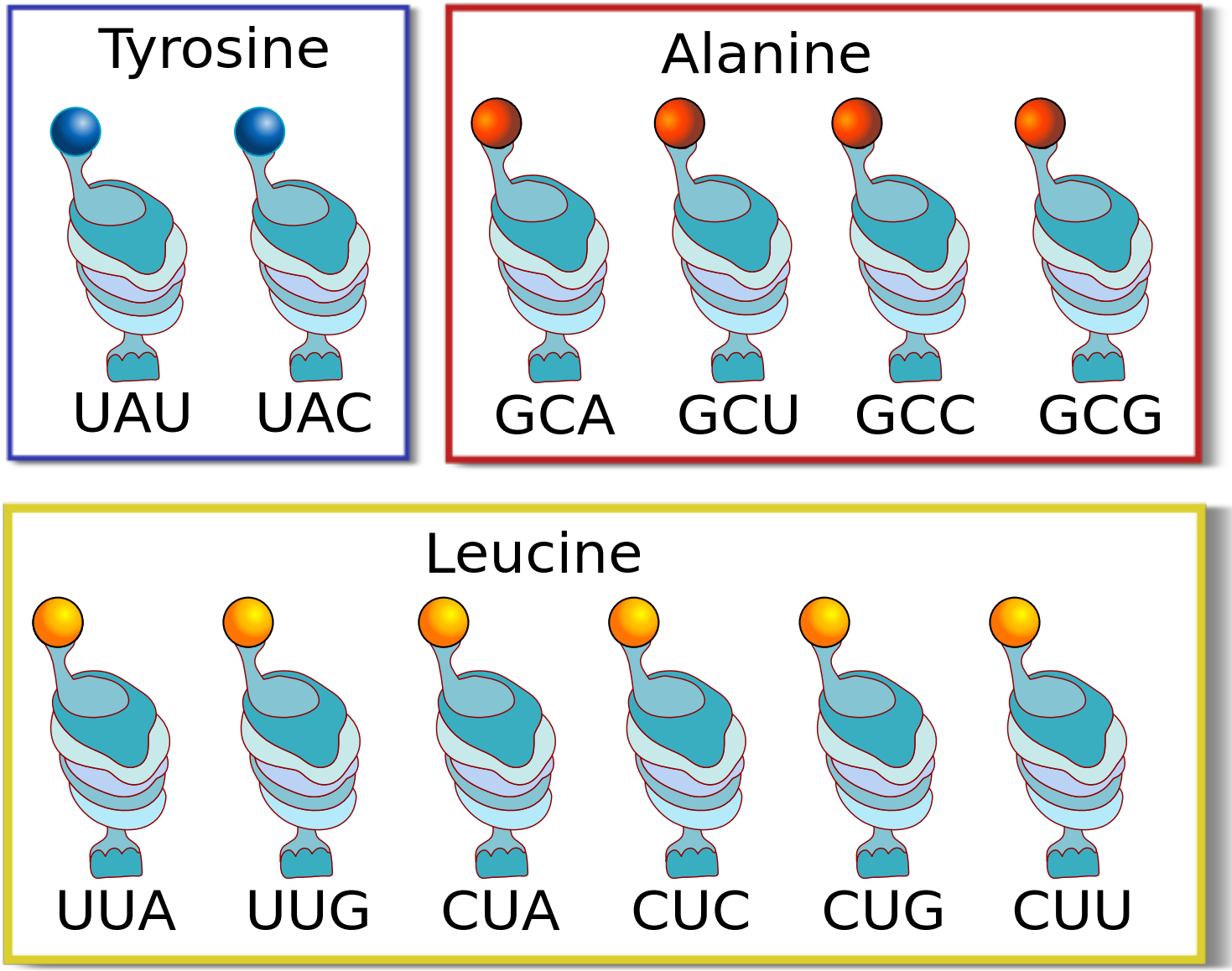
GCU

Leucine

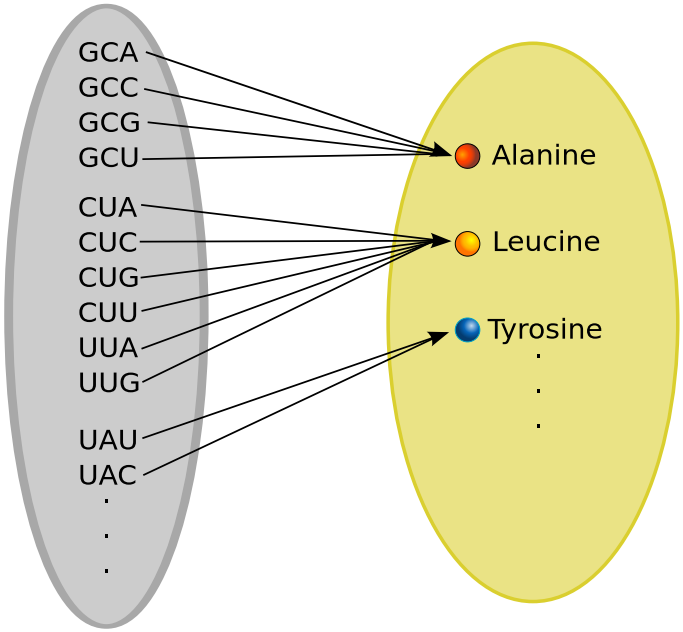


CUA

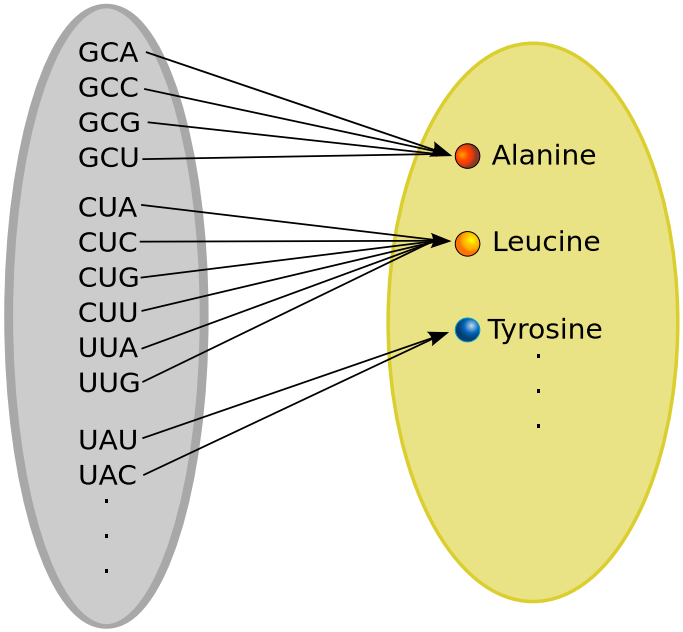
Genetic code is redundant



Genetic code is a many-to-one mapping



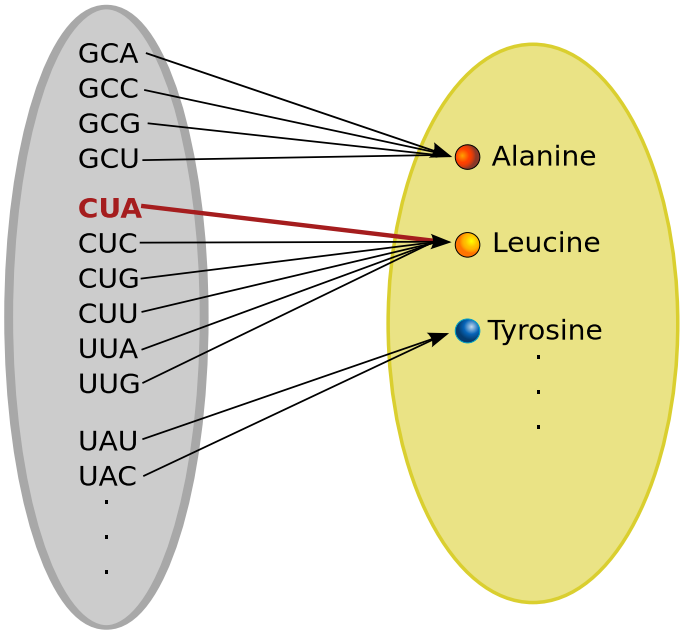
Genetic code is a many-to-one mapping



Definition

- $\lambda(i)$ denotes the i^{th} amino acid
- $\lambda_j(i)$ denotes its j^{th} codon
- $|\lambda(i)|$ denotes its codon count

Genetic code is a many-to-one mapping



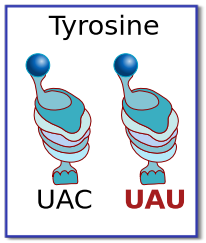
Definition

- $\lambda(i)$ denotes the i^{th} amino acid
- $\lambda_j(i)$ denotes its j^{th} codon
- $|\lambda(i)|$ denotes its codon count

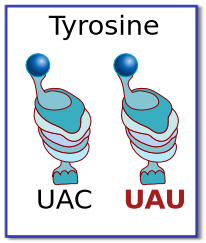
Example

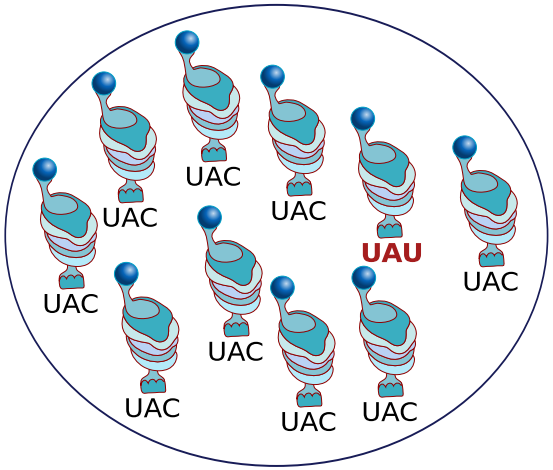
$$\lambda(2) = \texttt{Leucine}$$
$$|\lambda(2)| = 6$$
$$\lambda_1(2) = \texttt{CUA}$$

Codon frequency



Codon frequency

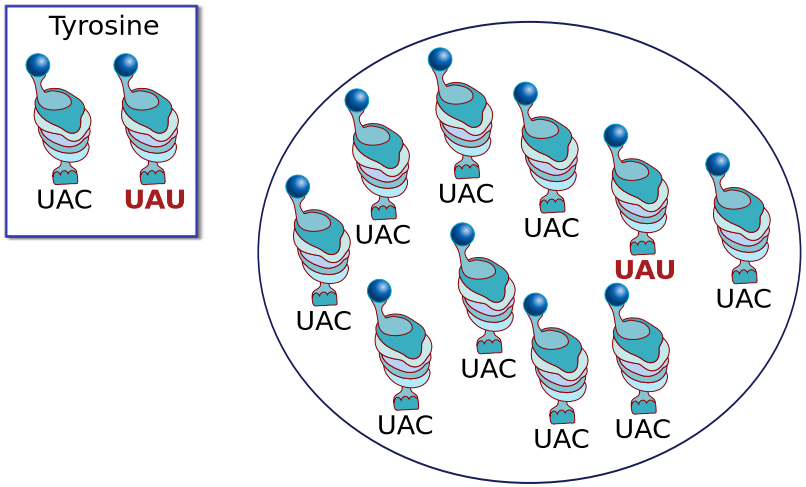




Definition

Let $\rho_j(i)$ denotes the *relative frequency* of j^{th} codon of i^{th} amino acid.

Codon frequency



Definition

Let $\rho_j(i)$ denotes the *relative frequency* of j^{th} codon of i^{th} amino acid.

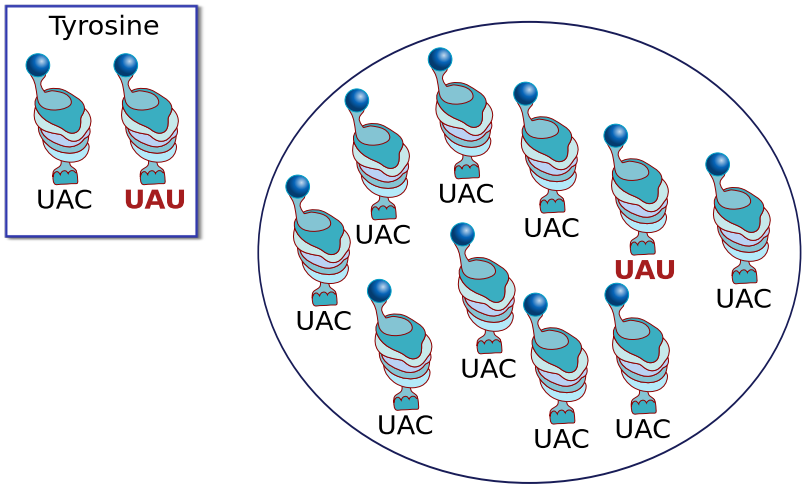
Example

$$\lambda(3) = \texttt{Tyrosine}$$

$$\rho_1(3) = \frac{9}{9+1} = 0.9$$

$$\rho_2(3) = \frac{1}{9+1} = 0.1$$

Codon frequency



Definition

Let $\rho_j(i)$ denotes the *relative frequency* of j^{th} codon of i^{th} amino acid.

Example

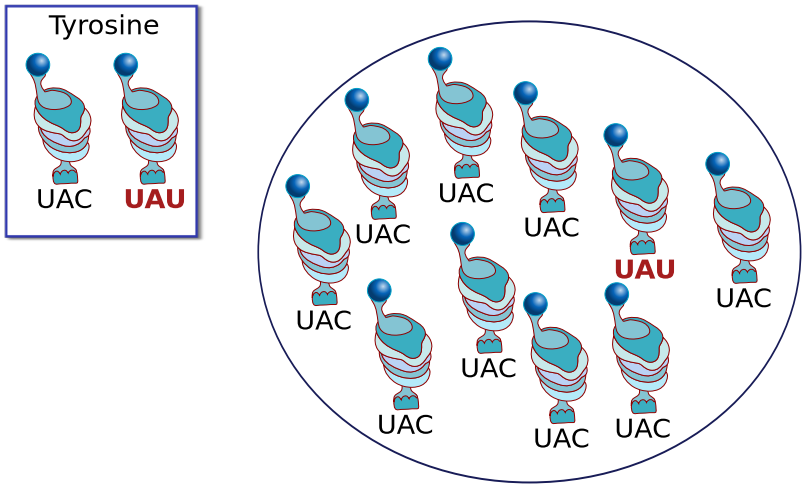
$$\lambda(3) = \texttt{Tyrosine}$$

$$\rho_1(3) = \frac{9}{9+1} = 0.9$$

$$\rho_2(3) = \frac{1}{9+1} = 0.1$$

$\lambda_1(3)$ is a *most frequent codon*

Codon fitness

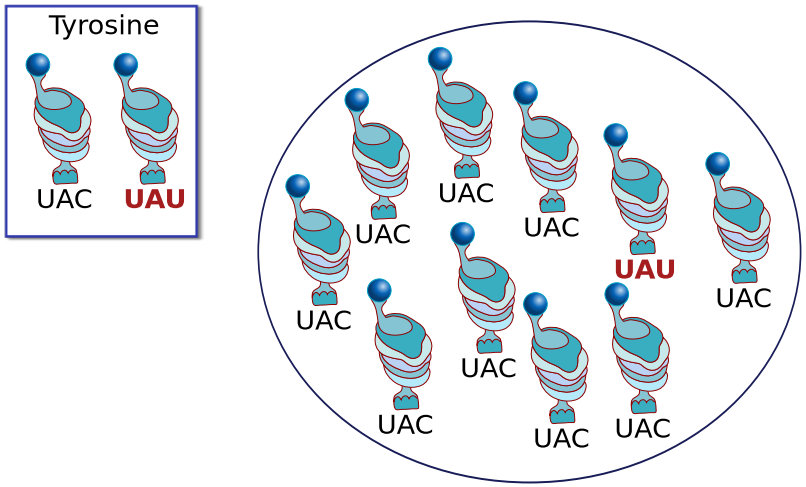


Definition

Let $\tau_j(i)$ denote the *codon fitness* of j^{th} codon of i^{th} amino acid.

A codon's fitness is its frequency relative to the most frequent codon of the same amino acid.

Codon fitness



Definition

Let $\tau_j(i)$ denote the *codon fitness* of j^{th} codon of i^{th} amino acid.

A codon's fitness is its frequency relative to the most frequent codon of the same amino acid.

Example

$$\lambda(3) = \texttt{Tyrosine}$$

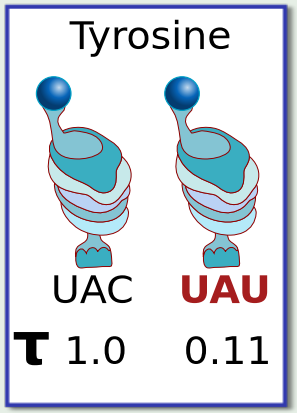
$$\tau_2(3) = \frac{\rho_2(3)}{\rho_1(3)} = \frac{0.1}{0.9} \approx 0.11$$

The codon adaption index

Given an amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, and a corresponding codon design $S = c_1, c_2, \ldots, c_{|A|}$:

$$\mathrm{CAI}(S, A) = \left(\prod_{i=1}^{|A|} \tau_{c_i}(\alpha_i) \right)^{\frac{1}{|A|}}$$

Example



A	Tyrosine	Tyrosine	Tyrosine	Tyrosine
S	UAC	UAU	UAU	UAC
	1.0	0.11	0.11	1.0

$$CAI(S, A) = (1.0 \times 0.11 \times 0.11 \times 1.0)^{\frac{1}{4}} \approx 0.33$$

Optimizing CAI

The CAI codon optimization problem

Instance: Amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$.

Problem: Find a codon design S^* corresponding to A such that:

- $CAI(S^*, A) = \max\{CAI(S, A) | S \in \mathbf{S}(A)\}$

where $\mathbf{S}(A)$ is the set of all codon designs for A.

Optimizing CAI

The CAI codon optimization problem

Instance: Amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$.

Problem: Find a codon design S^* corresponding to A such that:

- $CAI(S^*, A) = \max\{CAI(S, A) | S \in \mathbf{S}(A)\}$

where $\mathbf{S}(A)$ is the set of all codon designs for A.

Note: trivially solvable in linear time

Forbidden motifs and desirable motifs

The catch

Certain motifs (substrings) should **not appear** in the codon design.

Forbidden motifs and desirable motifs

The catch

Certain motifs (substrings) should **not appear** in the codon design.

- restriction enzyme motifs
  (*e.g.*, MlyI restriction enzyme motif: `GAGTC`)

Forbidden motifs and desirable motifs

The catch

Certain motifs (substrings) should **not appear** in the codon design.

- restriction enzyme motifs
  (*e.g.*, MlyI restriction enzyme motif: `GAGTC`)
- polyhomomeric regions
  (*e.g.*, `CCCCCC`)

Forbidden motifs and desirable motifs

The catch

Certain motifs (substrings) should **not appear** in the codon design.

- restriction enzyme motifs
  (*e.g.*, MlyI restriction enzyme motif: `GAGTC`)
- polyhomomeric regions
  (*e.g.*, `CCCCCC`)

Certain motifs (substrings) are desirable in the codon design.

- Immuno stimulatory motifs

Forbidden motifs and desirable motifs

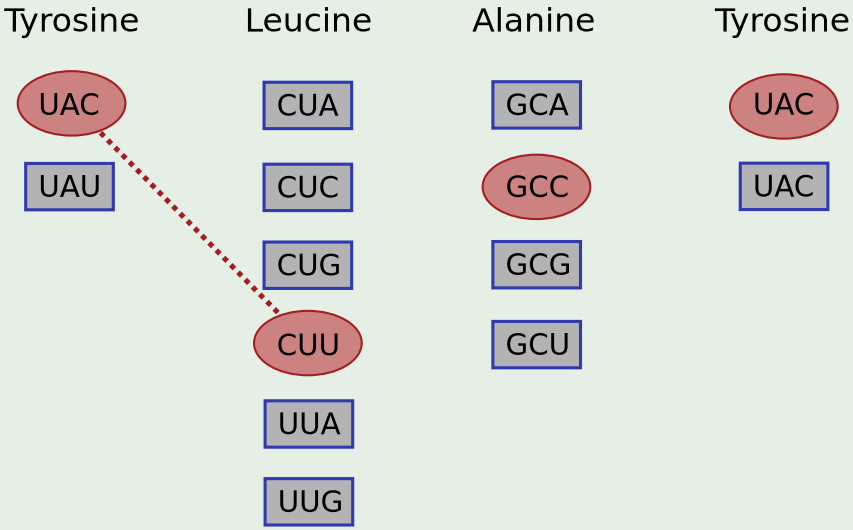
Example

Tyrosine	Leucine	Alanine	Tyrosine
UAC	CUA	GCA	UAC
UAU	CUC	GCC	UAC
	CUG	GCG	
	CUU	GCU	
	UUA		
	UUG		

$$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$$

Forbidden motifs and desirable motifs

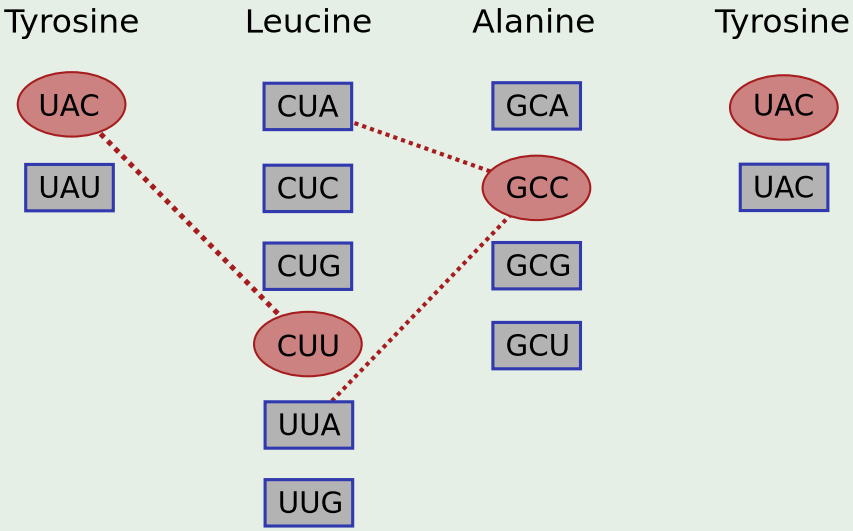
Example



$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$

Forbidden motifs and desirable motifs

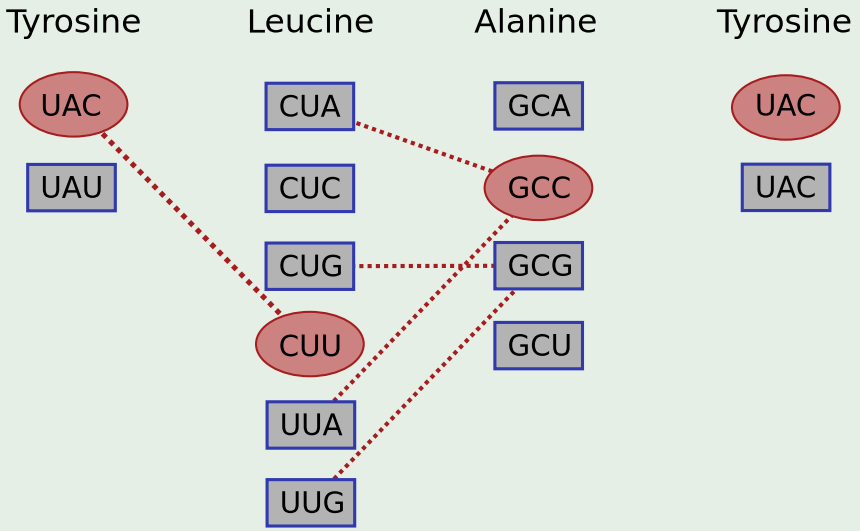
Example



$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$

Forbidden motifs and desirable motifs

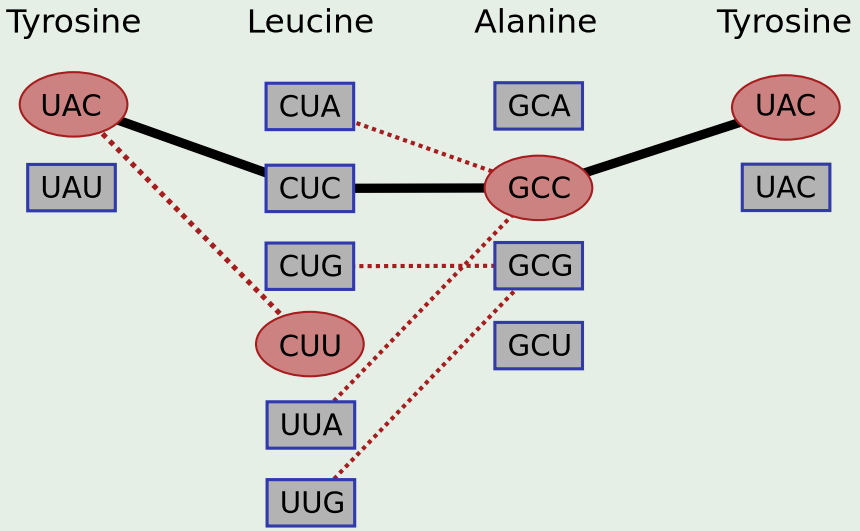
Example



$$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$$

Forbidden motifs and desirable motifs

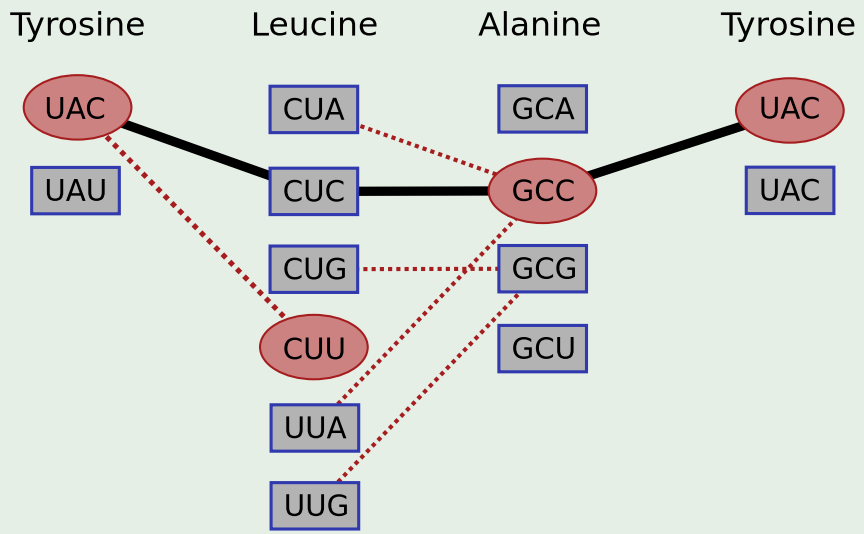
Example



$$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$$

Forbidden motifs and desirable motifs

Example



$\mathcal{F} = \{\mathtt{CCUU}, \mathtt{AGC}, \mathtt{UGGC}\}$

Detecting motifs

Equivalent to the dictionary matching problem.

- Classic dictionary (Aho & Corasick 1975)
- Succinct dictionary (Belazzougui 2010, Thachuk 2011)

Text of length h scanned for patterns in $O(h)$ time.

Motifs are of constant length

Observation

If the largest forbidden or desired motif is of length g, then any forbidden or desired motif can span at most $k+1$ consecutive codons, where $k = \lceil g/3 \rceil$.

Optimizing CAI and considering motifs

The CAI codon optimization problem with motif engineering

Instance: Amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, a set of forbidden motifs $\mathcal{F}$, and a set of desired motifs $\mathcal{D}$.

Optimizing CAI and considering motifs

The CAI codon optimization problem with motif engineering

Instance: Amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, a set of forbidden motifs $\mathcal{F}$, and a set of desired motifs $\mathcal{D}$.

Problem: Find a codon design S^* corresponding to A such that:

- S^* is *valid*, with respect to $\mathcal{F}$ and $\mathcal{D}$,
- $CAI(S^*, A) = \max\{CAI(S, A) | S \in \mathbf{S}(A)\}$

Optimizing CAI and considering motifs

The CAI codon optimization problem with motif engineering

Instance: Amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_{|A|}$, a set of forbidden motifs $\mathcal{F}$, and a set of desired motifs $\mathcal{D}$.

Problem: Find a codon design S^* corresponding to A such that:

- S^* is *valid*, with respect to $\mathcal{F}$ and $\mathcal{D}$,
- $CAI(S^*, A) = \max\{CAI(S, A) | S \in \mathbf{S}(A)\}$

where $\mathbf{S}(A)$ is the set of all valid codon designs for A.

We will develop a dynamic programming algorithm.

(We ignore desirable motifs.)

Notation

- $\lambda_j(i)$
  j^{th} codon of i^{th} amino acid
- $\tau_j(i)$
  j^{th} codon's fitness w.r.t i^{th} amino acid

We will develop a dynamic programming algorithm.

(We ignore desirable motifs.)

Notation

- $\lambda_j(i)$
  j^{th} codon of i^{th} amino acid
- $\tau_j(i)$
  j^{th} codon's fitness w.r.t i^{th} amino acid
- $\mathcal{M}_{\mathcal{F}}(\lambda_{c_i}(\alpha_j)\ldots\lambda_{c_{i+q}}(\alpha_{j+q}))$
  count of forbidden motifs
- $\mathcal{M}'_{\mathcal{F}}(\lambda_{c_i}(\alpha_j)\ldots\lambda_{c_{i+q}}(\alpha_{j+q}))$
  count of forbidden motifs ending in last codon

We make use of two k-dimensional DP matrices.

The Forbidden motif DP matrix

$$\mathbf{F}^{\mathbf{i}}_{\mathbf{c_{i-k+1},\cdots,c_{i-1},c_i}}$$

Denotes the minimum possible number of forbidden motifs in a DNA sequence which codes for an amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_i$, given that the last k codons (of i total codons) have indices denoted as $c_{i-k+1}, \ldots, c_{i-1}, c_i$.

We make use of two k-dimensional DP matrices.

The Forbidden motif DP matrix

$$\mathbf{F}^{\mathbf{i}}_{\mathbf{c_{i-k+1},\cdots,c_{i-1},c_i}}$$

Denotes the minimum possible number of forbidden motifs in a DNA sequence which codes for an amino acid sequence $A = \alpha_1, \alpha_2, \ldots, \alpha_i$, given that the last k codons (of i total codons) have indices denoted as $c_{i-k+1}, \ldots, c_{i-1}, c_i$.

The CAI value DP matrix

$$\mathbf{P}^{\mathbf{i}}_{\mathbf{c_{i-k+1},\cdots,c_{i-1},c_i}}$$

Denotes the maximum possible CAI score among all valid sequences.

Simply evaluate all assignments to first k codons.

$$F^k_{c_1,c_2,\ldots,c_{k-1},c_k} = M_{\mathcal{F}}\left(\lambda_{c_1}(\alpha_1)\lambda_{c_2}(\alpha_2)\ldots\lambda_{c_{k-1}}(\alpha_{k-1})\lambda_{c_k}(\alpha_k)\right)$$

$$P^k_{c_1,c_2,\ldots,c_{k-1},c_k} = \prod_{i=1}^{k}\left(\tau_{c_i}(\alpha_i)\right)$$

Simply evaluate all assignments to first k codons.

$$F^k_{c_1,c_2,\ldots,c_{k-1},c_k} = M_{\mathcal{F}}\left(\lambda_{c_1}(\alpha_1)\lambda_{c_2}(\alpha_2)\ldots\lambda_{c_{k-1}}(\alpha_{k-1})\lambda_{c_k}(\alpha_k)\right)$$

$$P^k_{c_1,c_2,\ldots,c_{k-1},c_k} = \prod_{i=1}^{k}\left(\tau_{c_i}(\alpha_i)\right)$$

Complexity analysis

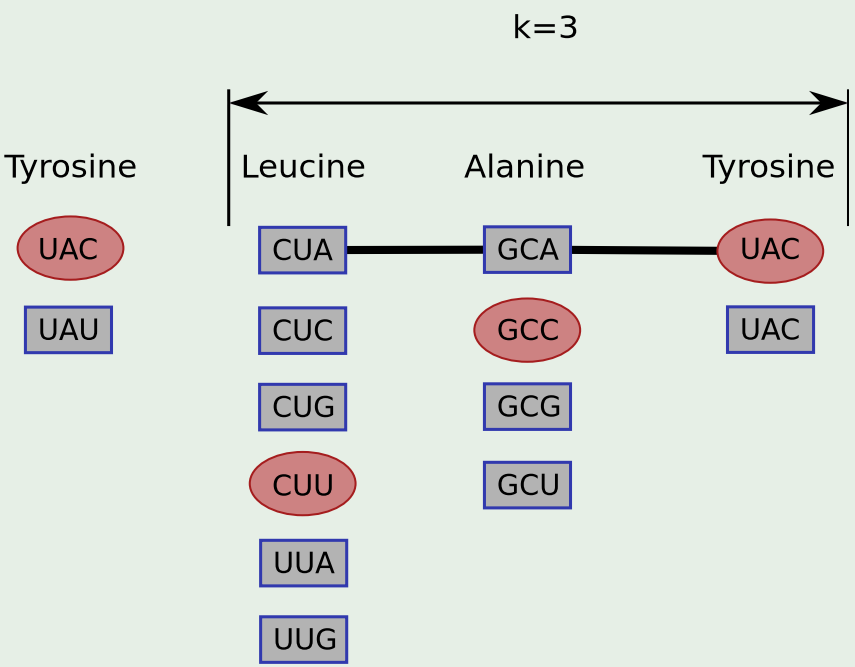
- at most 6 codons for each of the k positions
  $O(6^k)$ assignments
- evaluating one assignment for motifs
  $O(k)$ time
- $O(6^k k)$ time and $O(6^k)$ words of space

The algorithm
The recursive case case ($i > k$)

For a fixed assignment to last k codons:

$$F^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} = \min_{1 \le c_{i-k} \le |\lambda(\alpha_{i-k})|} \left\{ F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k}) \ldots \lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right) \right\}$$

Example

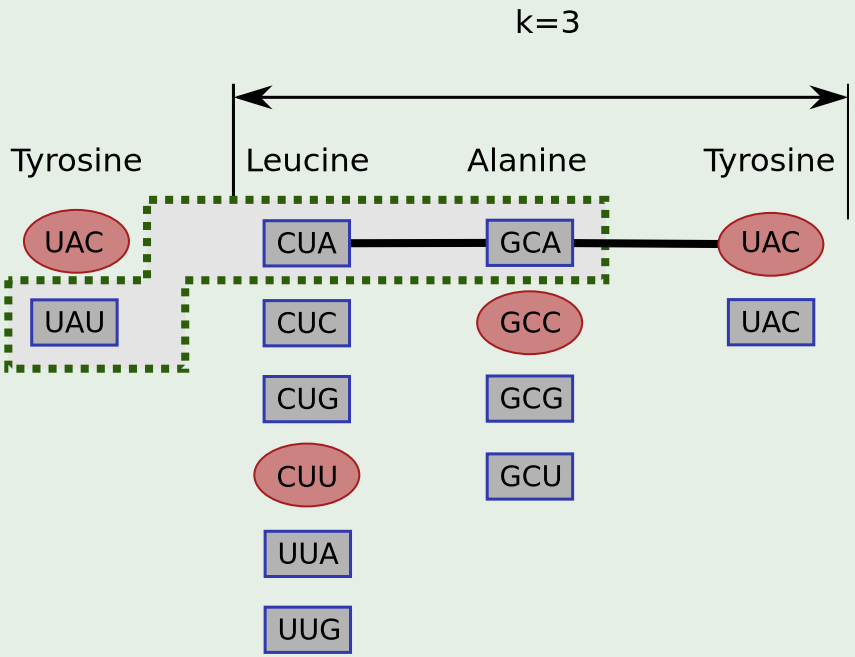


The algorithm
The recursive case case ($i > k$)

For a fixed assignment to last k codons:

$$F^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} = \min_{1 \le c_{i-k} \le |\lambda(\alpha_{i-k})|} \left\{ F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k}) \ldots \lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right) \right\}$$

Example

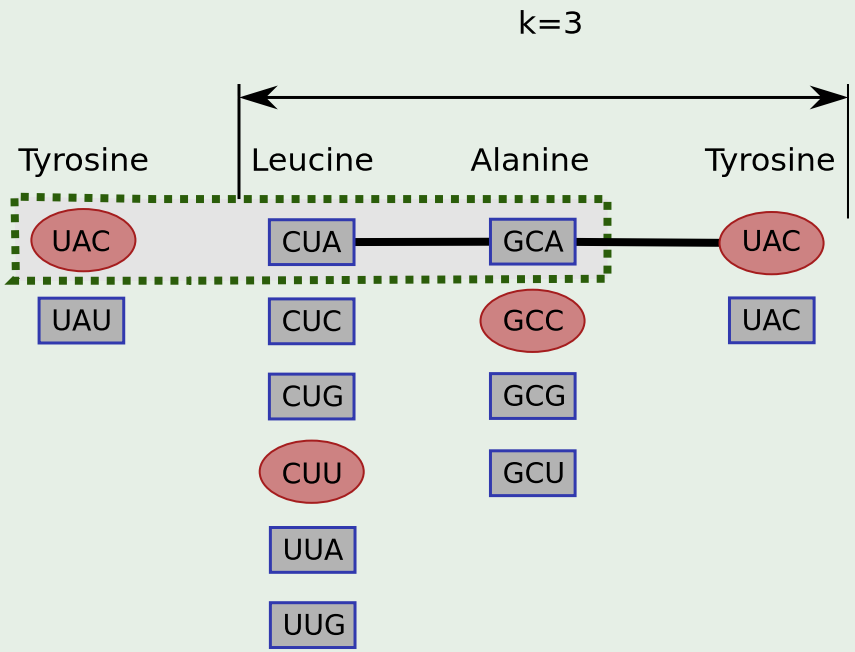


The algorithm
The recursive case case ($i > k$)

For a fixed assignment to last k codons:

$$F^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} = \min_{1 \le c_{i-k} \le |\lambda(\alpha_{i-k})|} \left\{ F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k}) \ldots \lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right) \right\}$$

Example

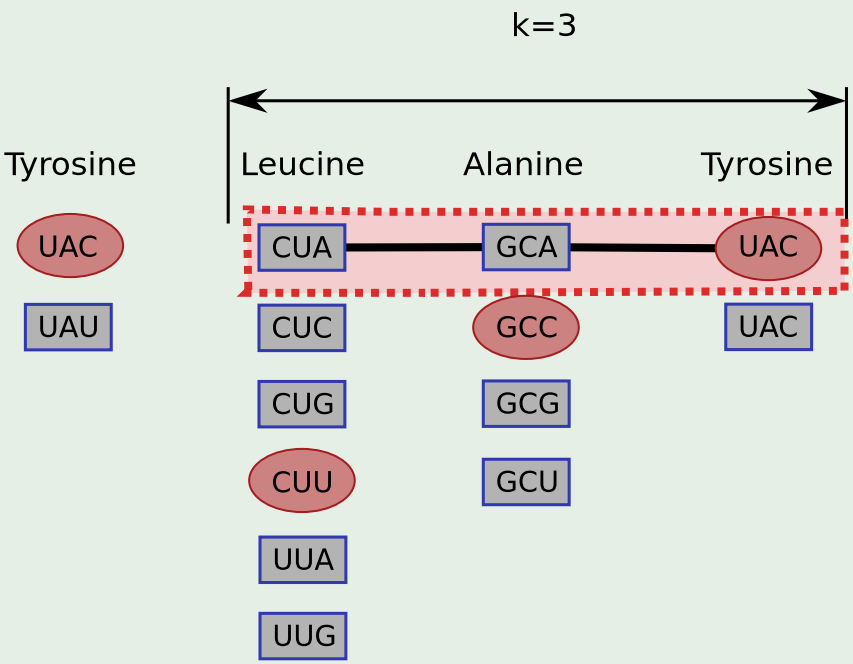


The algorithm
The recursive case case ($i > k$)

For a fixed assignment to last k codons:

$$F^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} = \min_{1 \le c_{i-k} \le |\lambda(\alpha_{i-k})|} \left\{ F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k}) \ldots \lambda_{c_{i-1}}(\alpha_{i-1})\lambda_{c_i}(\alpha_i)\right) \right\}$$

Example

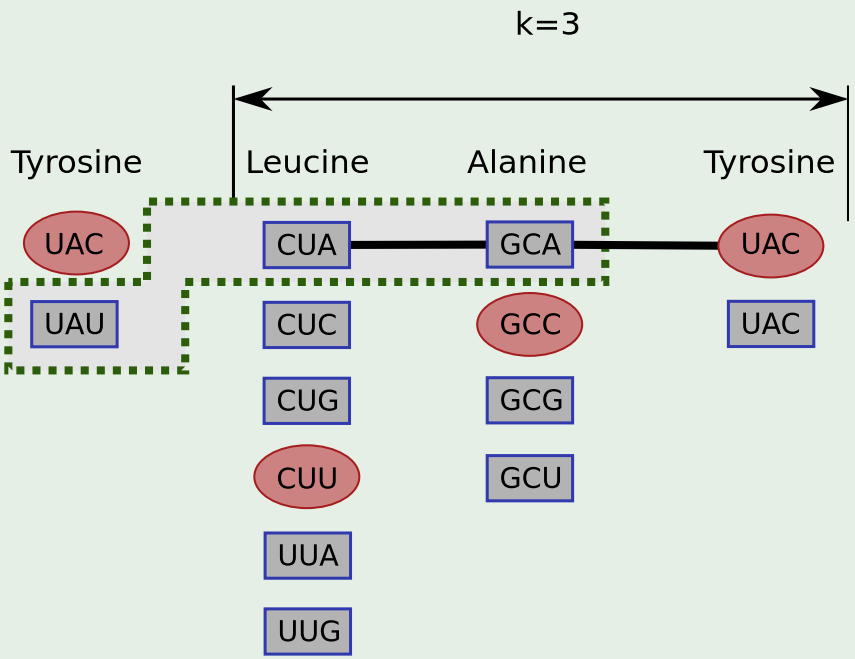


For a fixed assignment to last k codons:

$$P^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} = \max_{1 \le c_{i-k} \le |\lambda(\alpha_{i-k})|}$$

$$\begin{cases} -\infty & \text{, if } F^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} + M'_{\mathcal{F}}\left(\lambda_{c_{i-k}}(\alpha_{i-k}) \ldots \lambda_{c_i}(\alpha_i)\right) \neq F^{i}_{c_{i-k+1},\ldots,c_{i-1},c_i} \\ \tau_{c_i}(\alpha_i) \times P^{i-1}_{c_{i-k},\ldots,c_{i-2},c_{i-1}} & \text{, otherwise} \end{cases}$$

Example



The algorithm

The recursive case case ($i > k$)

$$\widetilde{F_k^i} = \min_{\substack{1 \le c_i \le |\lambda(\alpha_i)| \\ 1 \le c_{i-1} \le |\lambda(\alpha_{i-1})| \\ \vdots \\ 1 \le c_{i-k+1} \le |\lambda(\alpha_{i-k+1})|}} \left\{ F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} \right\}$$

$$\widetilde{P_k^i} = \max_{\substack{1 \le c_i \le |\lambda(\alpha_i)| \\ 1 \le c_{i-1} \le |\lambda(\alpha_{i-1})| \\ \vdots \\ 1 \le c_{i-k+1} \le |\lambda(\alpha_{i-k+1})|}} \left\{ \begin{array}{ll} P^i_{c_{i-k+1},\ldots,c_{i-1},c_i} & \text{, if } F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \widetilde{F_k^i} \\ -\infty & \text{, otherwise} \end{array} \right\}$$

The algorithm

The recursive case case ($i > k$)

$$\widetilde{F}_k^i = \min_{\substack{1 \le c_i \le |\lambda(\alpha_i)| \\ 1 \le c_{i-1} \le |\lambda(\alpha_{i-1})| \\ \vdots \\ 1 \le c_{i-k+1} \le |\lambda(\alpha_{i-k+1})|}} \left\{ F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} \right\}$$

$$\widetilde{P}_k^i = \max_{\substack{1 \le c_i \le |\lambda(\alpha_i)| \\ 1 \le c_{i-1} \le |\lambda(\alpha_{i-1})| \\ \vdots \\ 1 \le c_{i-k+1} \le |\lambda(\alpha_{i-k+1})|}} \left\{ \begin{array}{ll} P^i_{c_{i-k+1},\ldots,c_{i-1},c_i} & \text{, if } F^i_{c_{i-k+1},\ldots,c_{i-1},c_i} = \widetilde{F}_k^i \\ -\infty & \text{, otherwise} \end{array} \right\}$$

Complexity analysis

- $O(6^k)$ codon assignments evaluated at n positions
- $O(k)$ time to evaluate assignment
- $O(6^k kn) = O(n)$ time and space overall

  previous state-of-the-art $\theta(n^2)$ time/space (Satya et al., 2003)

Experimental setup

Data set

- 3,157 sequences from GENCODE subset of ENCODE dataset
  - comprises 1% of human genome
  - representative of genome in various characteristics
  - range in length from 75 to 8186 bases
  - mean length of 173 bases (267 bases standard deviation)

Experimental setup

Data set

- 3,157 sequences from GENCODE subset of ENCODE dataset
  - comprises 1% of human genome
  - representative of genome in various characteristics
  - range in length from 75 to 8186 bases
  - mean length of 173 bases (267 bases standard deviation)
- using codon frequencies of Escherichia coli

Experimental setup

Data set

- 3,157 sequences from GENCODE subset of ENCODE dataset
  - comprises 1% of human genome
  - representative of genome in various characteristics
  - range in length from 75 to 8186 bases
  - mean length of 173 bases (267 bases standard deviation)
- using codon frequencies of Escherichia coli
- $|\mathcal{F}| = 10 \quad |\mathcal{D}| = 33$

  $k = 3$ (motifs of length 9 or less)

Experimental setup

Data set

- 3,157 sequences from GENCODE subset of ENCODE dataset
  - comprises 1% of human genome
  - representative of genome in various characteristics
  - range in length from 75 to 8186 bases
  - mean length of 173 bases (267 bases standard deviation)
- using codon frequencies of Escherichia coli
- $|\mathcal{F}| = 10 \quad |\mathcal{D}| = 33$

  $k = 3$ (motifs of length 9 or less)

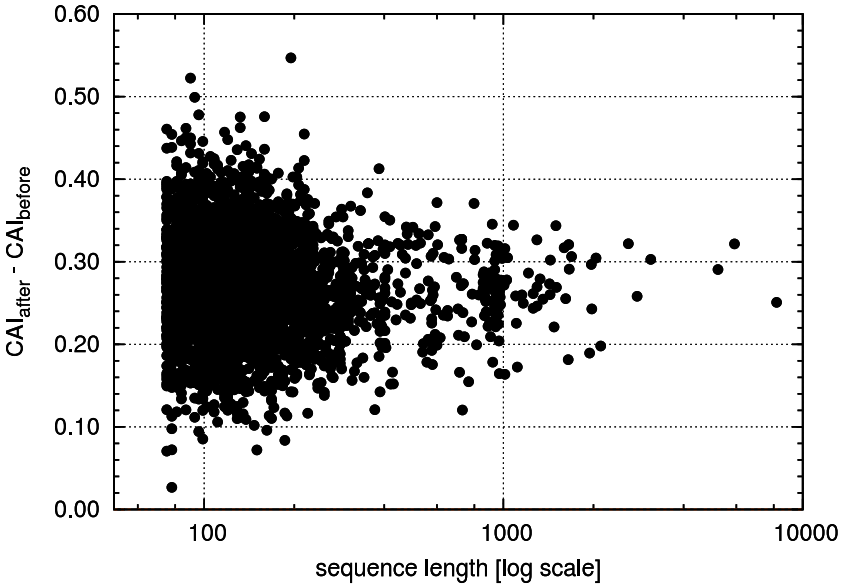
Implementation & Hardware

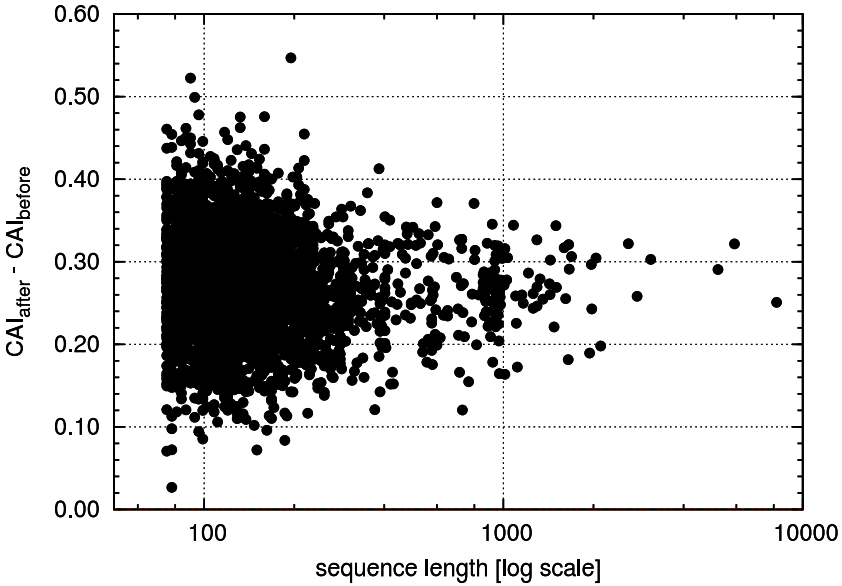
- Implemented in C++
- Pentium IV 2.4 GhZ
- 1 GB RAM

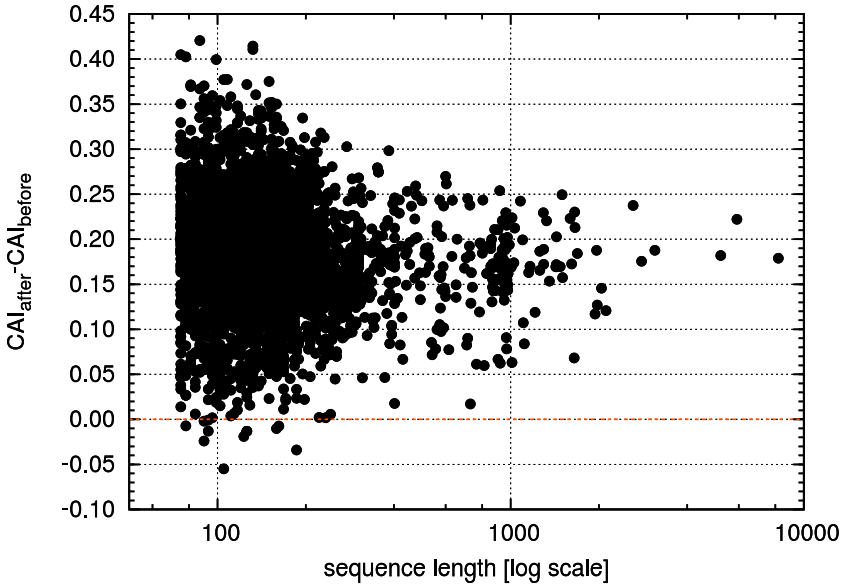
motif sets	CAI value	# forbidden	# desired
none (wild-types)	0.65 (0.06)	9.24 (16.24)	0.49 (1.06)

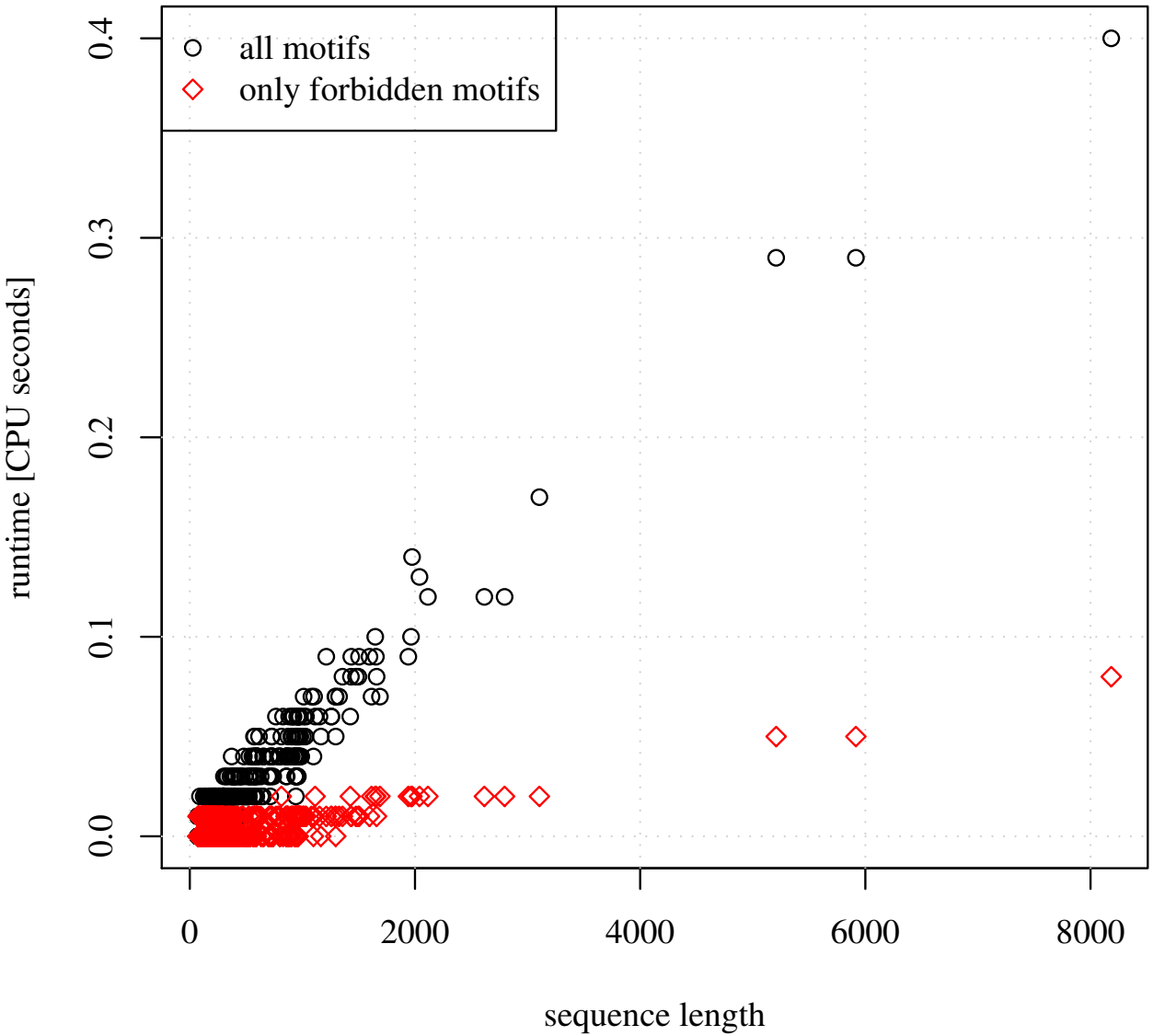
motif sets	CAI value	# forbidden	# desired
none (wild-types)	0.65 (0.06)	9.24 (16.24)	0.49 (1.06)
forbidden	0.92 (0.04)	0.14 (0.45)	0 (0.00)

motif sets	CAI value	# forbidden	# desired
none (wild-types)	0.65 (0.06)	9.24 (16.24)	0.49 (1.06)
forbidden	0.92 (0.04)	0.14 (0.45)	0 (0.00)
forbidden & desired	0.83 (0.05)	0.14 (0.45)	10.13 (14.84)


0.60
0.50
0.40
0.30
0.20
0.10
0.00
CAI_after - CAI_before
100
1000
10000
sequence length [log scale]


0.60
0.50
0.40
0.30
0.20
0.10
0.00
CAI$_{after}$ - CAI$_{before}$
100
1000
10000
sequence length [log scale]


0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
-0.05
-0.10
CAI$_{after}$-CAI$_{before}$
100
1000
10000
sequence length [log scale]


all motifs
only forbidden motifs
runtime [CPU seconds]
0.0
0.1
0.2
0.3
0.4
0
2000
4000
6000
8000
sequence length

Conclusions & Future Work

Conclusions

- CAI of gene can be optimized effectively
- motifs can be removed/added effectively
- $O(n)$ time/space algorithm for constant length motifs
- algorithm is fast in practice

Conclusions & Future Work

Conclusions

- CAI of gene can be optimized effectively
- motifs can be removed/added effectively
- $O(n)$ time/space algorithm for constant length motifs
- algorithm is fast in practice

Open Problem

Design codon sequence free of secondary structure