BioSystems 128 (2015) 9–18

Contents lists available at ScienceDirect

BioSystems

journal homepage: www.elsevier.com/locate/biosystems

On models of the genetic code generated by binary dichotomic algorithms

Markus Gumbel[a,*], Elena Fimmel[b], Alberto Danielli[c], Lutz Strüngmann[b]

[a] Mannheim University of Applied Sciences, Institute for Medical Informatics, Paul-Wittsack-Straße 10, D-68163 Mannheim, Germany
[b] Mannheim University of Applied Sciences, Institute for Applied Mathematics, Paul-Wittsack-Straße 10, D-68163 Mannheim, Germany
[c] University of Bologna, Department of Pharmacy and Biotechnology, Via Irnerio 42, 40126 Bologna, Italy

ARTICLE INFO

Article history:
Received 30 September 2014
Received in revised form
24 November 2014
Accepted 16 December 2014
Available online 18 December 2014

Keywords:
Genetic code
Dichotomic partition
Rumer
Scan algorithm
Ribosome

ABSTRACT

In this paper we introduce the concept of a BDA-generated model of the genetic code which is based on binary dichotomic algorithms (BDAs). A BDA-generated model is based on binary dichotomic algorithms (BDAs). Such a BDA partitions the set of 64 codons into two disjoint classes of size 32 each and provides a generalization of known partitions like the Rumer dichotomy. We investigate what partitions can be generated when a set of different BDAs is applied sequentially to the set of codons. The search revealed that these models are able to generate code tables with very different numbers of classes ranging from 2 to 64. We have analyzed whether there are models that map the codons to their amino acids. A perfect matching is not possible. However, we present models that describe the standard genetic code with only few errors. There are also models that map all 64 codons uniquely to 64 classes showing that BDAs can be used to identify codons precisely. This could serve as a basis for further mathematical analysis using coding theory, for example. The hypothesis that BDAs might reflect a molecular mechanism taking place in the decoding center of the ribosome is discussed. The scan demonstrated that binary dichotomic partitions are able to model different aspects of the genetic code very well.

The search was performed with our tool Beady-A. This software is freely available at http://mi.informatik.hs-mannheim.de/beady-a. It requires a JVM version 6 or higher.

1. Introduction

Almost all organisms use the same mapping of codons to amino acids – the universal genetic code (see Fig. 1(b)), which maps 64 codons to only 20 amino acid species (and 3 stop signals). There are 18 alternative genetic code tables annotated which are slightly different in the assignment of the codons (Osawa et al., 1992; Jukes and Osawa, 1993).[1] Nevertheless, these minor variations largely preserve the redundancy in amino acid assignment. Hence the degeneracy of orthologous genetic codes is highly conserved. It has been shown that there are more than 10^{84} different genetic codes with the same degeneracy and it was demonstrated that the genetic code is far from being random (Koonin and Novozhilov, 2009). Besides several hypotheses that explain why the genetic code could have been developed as it is (i. e. the frozen accident theory developed by Crick (Crick, 1968; Sella and Ardell, 2006), the stereo-chemical-theory (originally proposed by Gamow, 1954), the co-evolution theory (Wong, 1975; Giulio, 2008) or the adaptive theory (Haig and Hurst, 1991; Freeland et al., 2003), ongoing research still addresses the question whether its structure reflects informational properties useful for error detection or frame retrieval in translation (Crick et al., 1957; Arquès and Michel, 1996; Seligmann, 2007; Guilloux and Jestin, 2012). Recent mathematical models addressed the structure of the genetic code using dichotomic partitions or dichotomic classes, classifying the codons into two partitions of equal size. The dichotomic classes and their non-random correlations were proposed to contribute to frame retrieval and error detection properties, underpinning a robustness of the code (Giannerini, 2012).

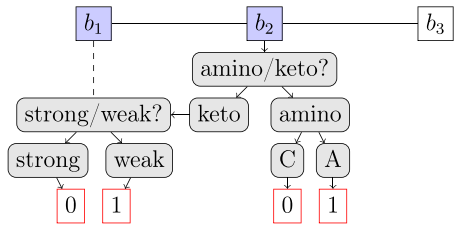
The Russian theoretical physicist Rumer was the first to introduce dichotomic partitions in 1966 (Rumer, 1966). He found that the set of codons can be divided into two subsets of the same cardinality such that (1) for codons belonging to the first subset (represented as digit 0 or class H_0) the first two positions within a codon are sufficient to determine the corresponding amino acid

* Corresponding author. Tel.: +49 621 292 6246.
E-mail addresses: m.gumbel@hs-mannheim.de (M. Gumbel), e.fimmel@hs-mannheim.de (E. Fimmel), alberto.danielli@unibo.it (A. Danielli), l.struengmann@hs-mannheim.de (L. Strüngmann).

[1] See also online at http://www.ncbi.nlm.nih.gov/Taxonomy/Utils/wprintgc.cgi?mode=c.

http://dx.doi.org/10.1016/j.biosystems.2014.12.001
0303-2647/

a)



b)

	U		C		A		G		
U	Phe	**111**	Ser	001	Tyr	101	Cys	**111**	U
U	Phe	**111**	Ser	001	Tyr	101	Cys	**111**	C
U	Leu	**110**	Ser	010	!	110	!	**110**	A
U	Leu	**100**	Ser	000	!	100	Trp	**100**	G
C	Leu	010	Pro	000	His	**100**	Arg	010	U
C	Leu	010	Pro	000	His	**100**	Arg	010	C
C	Leu	010	Pro	010	Gln	**110**	Arg	010	A
C	Leu	000	Pro	000	Gln	**100**	Arg	000	G
A	Ile	**111**	Thr	001	Asn	101	Ser	**111**	U
A	Ile	**111**	Thr	001	Asn	101	Ser	**111**	C
A	Ile	**110**	Thr	010	Lys	**110**	Arg	**110**	A
A	Met	**100**	Thr	000	Lys	**100**	Arg	**100**	G
G	Val	011	Ala	001	Asp	101	Gly	011	U
G	Val	011	Ala	001	Asp	101	Gly	011	C
G	Val	011	Ala	011	Glu	**111**	Gly	011	A
G	Val	001	Ala	001	Glu	**101**	Gly	001	G

c)

BDA	(i_1, i_2)	Q_1	Q_2
Rumer	(2, 1)	(C, A)≙Am	{C, G}≙ S
Parity	(3, 2)	(G, A)≙ R	{C, A}≙ Am
Complementary	(1, 3)	(C, G)≙ S	{G, A}≙ R

Fig. 1. (a) Visual representation of the Rumer-class algorithm (adapted from Fimmel et al., 2013). (b) Universal genetic code and three overlapping dichotomic partitions generating 8 classes (shown in different colors). A cell contains the dichotomic classes (from left to right) Rumer, Parity and Complementary. 0: H_0, 1: H_1. All digits 1 in the Rumer's class are set in bold. (c) Parameters for the three BDAs. Am: Amino, etc. as explained in the text. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

while (2) for codons from the second dichotomic class (digit 1, H_1) the third position is indispensable for it (see Fig. 1(b)). These partitions have also the remarkable feature that a codon of one class can be swapped to the other class and vice versa by a bijective transformation of their bases: $A \leftrightarrow C$ and $U \leftrightarrow G$. Rumer also gave an algorithmic description how the Rumer-class can be determined for a given codon (see Fig. 1(a)).

Gonzalez and co-authors adopted Rumer's algorithm to their model of the genetic code and defined in an analogous way two algorithms for the so called Parity and Hidden classes which matches in a natural way to their model of the genetic code (Gonzalez, 2008). The model of the genetic code developed in Gonzalez (2008) uses a non-power representation of numbers to explain how 64 codons can be mapped to 20 amino acids. Each codon is uniquely represented as a binary string of length 6 (referring to $2^6 = 64$ codons). The Parity class, for instance, is defined as the parity of the associated binary string. It was demonstrated that there are short-range correlations of the Parity and Hidden classes in genes (Giannerini, 2012; Gonzalez et al., 2008).

Interestingly, each of the three algorithms for the calculation of the Rumer-, Parity- and Hidden-class derive their decision from biochemical properties of the bases involved. The algorithms distinguish whether a base is of type purine (denoted as R = {A, G}) or pyrimidine (Y = {C, U}), keto (K = {U, G} or amino (Am = (C, A), or strong (S = {C, G}) or weak (W = {A, U}).

As an example, Fig. 1(a) shows a visual representation of the algorithm producing the Rumer classes. As one can see, the algorithm first determines if the second base of the codon belongs to the amino or keto class and, if not, it considers the first base of the codon and asks whether it is strong or weak.

The algorithmic approach for the definition of dichotomic partitions was recently further investigated and generalized in Fimmel et al. (2013) where the concept of a **Binary Dichotomic Algorithm** (in short: BDA) was introduced and shown to have a tentative biological counterpart in the decoding center of the ribosome. It has been shown that different dichotomic partitions can be linked to the chemical nature of the nucleoside bases which carry the information in the codon (e. g. S/W, Keto/amino; R/Y) (Giannerini (2012)). From an evolutionary point of view this could be of relevance for a decoding process, given that these chemical dichotomies are readily discriminable by biochemical as well as inorganic means, and therefore could provide a simple basis for deciphering the information.

Apparently, partitions and in particular dichotomic partitions like Rumer play an important role in models of the genetic code and it is therefore interesting to study BDAs in general (Hervé Seligmann, 2014; Seligmann, 2014; Demeshkina et al., 2012). The decisions which the algorithms make are binary, i. e. this is a very simple and yet elegant way to make a decision which can be performed on a biological entity like the ribosome or while recognizing a tRNA and its cognate amino acid. This paper investigates the possibility of explaining the structure of the genetic code as an overlapping of dichotomic partitions as generated by a set of BDAs. For instance, Fig. 1(b) shows a code table where the overlapping of the Rumer, Parity and Complementary partition leads to an assignment of the codons into 8 classes represented as a binary string of size 3.

Herein, we propose a new way to create novel models for the genetic code based on overlappings of dichotomic partitions. We will analyze by means of the software Beady-A what kind of code tables of this kind can be generated and whether they are suitable to model aspects of the genetic code. To this aim, different scan algorithms were developed to find solutions suitable to model the genetic code through BDAs.

2. Mathematical background

2.1. Preliminary definitions and notations

In the sequel B= A,C,G,U(T) will denote the set of four nucleotide bases Uracil (Thymine), Cytosine, Adenine, and Guanine, in short $U(T)$, C, A, G. A codon is an element of $\mathcal{B}^3$, e.g. ACU.

In Fimmel et al. (2013) the notion of binary dichotomic algorithms was given taking sequences of nucleotide bases of arbitrary length as input. For our investigations, it suffices to restrict ourselves to codons, i. e. sequences of length three. Let us recall the definition of a BDA in this special case:

Definition 2.1. An ordered pair (H_0, H_1) of subsets $H_0, H_1 \subseteq \mathcal{B}^3$ is called a **dichotomic partition** of $\mathcal{B}^3$ if $H_0 \cap H_1 = \emptyset$, $H_0 \cup H_1 = \mathcal{B}^3$ and $|H_0| = |H_1|$.

In other words, the set of 64 codons is divided into two disjoint subsets of equal size as shown in Fig. 1(b). The next definition shows the algorithmic way to obtain dichotomic partitions as discussed in Fimmel et al. (2013).

Definition 2.2. Let (H_0, H_1) be a dichotomic partition of $\mathcal{B}^3$. We call an algorithm $\mathcal{A}$ a **binary dichotomic algorithm (BDA) with dichotomic partition** (H_0, H_1) if it follows the following scheme:

$\mathcal{A}$ chooses two indices $i_1, i_2 \in \{1, 2, 3\}$ with $i_1 \neq i_2$, an ordered pair of different nucleotide bases $Q_1 = (B_1, B_2)$ and a subset $Q_2 \subset \mathcal{B}$ with $|Q_2| = 2$. Now $\mathcal{A}$ classifies $c = (b_1, b_2, b_3) \in \mathcal{B}^3$ as follows:

(A) if $b_{i_1} \in \{B_1, B_2\}$, then

$$(c \in H_0 \quad \text{if} \quad b_{i_1} = B_1,) \quad \text{and} \quad (c \in H_1 \quad \text{if} \quad b_{i_1} = B_2,)$$

(B) if $b_{i_1} \notin \{B_1, B_2\}$, then

$$(c \in H_0 \quad \text{if} \quad b_{i_2} \in Q_2,) \quad \text{and} \quad (c \in H_1 \quad \text{if} \quad b_{i_2} \notin Q_2 .)$$

We will call Q_1 and Q_2 the **questions** of $\mathcal{A}$, $i_1, i_2 \in \{1, 2, 3\}$ the **indices** of $\mathcal{A}$, and the pair (H_0, H_1) a **dichotomic partition of $\mathcal{B}^3$ generated by the binary dichotomic algorithm $\mathcal{A}$.**

The dichotomic classes Rumer, Parity and Complementary can be obtained by choosing the questions and the positions shown in Fig. 1(c). We will see in the next section that a BDA uniquely defines a dichotomic partition.

2.2. Characteristics of BDA

In this section we are interested in elementary properties of generalized BDAs. We first remark that a given BDA can also be understood as a function $\mathcal{A} : \mathcal{B}^3 \to \{0, 1\}$ just by assigning to a given codon $c \in \mathcal{B}^3$ the number 0 if it is classified for the class H_0 by the BDA and the number 1 otherwise.

$$\mathcal{A}(c) = \begin{cases} 0 , & c \in H_0 \\ 1 , & c \in H_1 \end{cases}$$

Obviously, the definition of a dichotomic partition shows some symmetry since partitions (H_0, H_1) and (H_1, H_0) are formally different but essentially give the same information. We therefore introduce the notion of a complementary BDA.

Definition 2.3. Let $\mathcal{A}$ be a given BDA. We will call a BDA $\bar{\mathcal{A}}$ the *complementary* BDA of $\mathcal{A}$ if the following property holds for all codons c:

$$\bar{A}(c) = 1 - A(c)$$

The next proposition shows how to determine the complementary BDA of a given BDA.

Proposition 1. *Let $\mathcal{A}$ be a BDA with questions $Q_1 = (B_1, B_2)\,(B_1 \neq B_2)$ and $Q_2 = \{B_3, B_4\}$ with $B_3 \neq B_4$. Then $\bar{Q}_1 = (B_2, B_1)$ and $\bar{Q}_2 = \mathcal{B}\backslash Q_2$ create the complementary BDA $\bar{\mathcal{A}}$ of $\mathcal{A}$.*

Proof. The reverse order of $\bar{Q}_1 = (B_2, B_1)$ implies that the partitions are swapped when question 1 is used. If question 2 is used, the complementary set of Q_2 is taken to reverse the partitions according to the definition. □

Clearly, being interested in the number of different sets of dichotomic classes one can omit the complementary BDA once we have the BDA. The following observation will be helpful. Let $\mathcal{A}$ be a BDA with questions $Q_1 = (B_1, B_2)\,(B_1 \neq B_2)$ and $Q_2 = \{B_3, B_4\}$ with $B_3 \neq B_4$, then the class H_0 contains exactly 16 codons with B_1 at position i_1 and exactly 8 codons with B_3 and 8 codons with B_4 at position i_2, respectively, but not B_1 at position i_1. We conclude the following

Proposition 2. *There are 432 different BDAs which generate 432 different dichotomic partitions and 216 different sets of dichotomic classes.*

Proof. Question 1 requires two different nucleotides where the order is important. Hence, there are $4 \cdot 3 = 12$ possibilities. Question 2 requires two different nucleotides where the order is not important as they form a set. There are $(4 \cdot 3)/2 = 6$ possibilities. Q_1 and Q_2 are using two different nucleotide positions i_1 and i_2 ($i_1 \neq i_2$). So there are $3 \cdot 2 = 6$ possibilities. Thus we have $12 \cdot 6 \cdot 6 = 432$ BDAs. However, a BDA and its complementary BDA generate the same classes, so the number of sets of dichotomic classes is only half of the size. In total we have $432/2 = 216$ combinations.

It remains to show that two different BDAs generate two different dichotomic partitions or equivalently, that a dichotomic partition can only be uniquely generated by a BDA. Thus, assume that (H_0, H_1) is a dichotomic partition generated by some BDA $\mathcal{A}$. Then for some position i_1 and some bases B_1, B_2 the set H_0 contains all possible 16 codons with B_1 at position i_1 but no codon with B_2 at position i_1. Clearly, this means that $\mathcal{A}$ has to have (B_1, B_2) as question 1 at position i_1. The remaining 16 codons of H_0 that do not have B_1 at position i_1 must then have one of the remaining bases from $\mathcal{B}\backslash\{B_1, B_2\}$ at position i_1. Now there must be a unique position i_2 and bases B_3, B_4 such that among these 16 codons there are exactly 8 with B_3 at position i_2 and 8 with bases B_4 at position i_2. This implies that $\mathcal{A}$ must have $\{B_3, B_4\}$ as question 2 at position i_2. □

In the sequel we will denote the set of all BDAs by $\mathcal{D}$ and the set of all "different" BDAs, i. e. equivalence classes consisting of two BDAs (which are complementary to each other) generating the same set of dichotomic classes, by $\mathcal{D}^*$.

Remark 3. Obviously, the number of all possibilities to divide the set of codons into ordered pairs of disjoint subsets of the same size (dichotomic classes) is $\binom{64}{32} \approx 5 \cdot 10^{17}$. In particular, the number of dichotomic partitions generated by BDAs (432) is extremely smaller than the number of dichotomic partitions in general.

2.3. BDA-generated models of the genetic code

In this section we will introduce a model of the genetic code that is generated by a sequence of BDAs. Such a model is defined in the following:

Definition 2.4. Let $k \in \mathbb{N}$. We will call a mapping

$$M : \mathcal{B}^3 \to \{0, 1\}^k$$

a **BDA-generated model** of the genetic code of **grade** k if there exist k different BDAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$ such that for all $c \in \mathcal{B}^3$ the following equation holds:

$$M(c) = (\mathcal{A}_1(c), \mathcal{A}_2(c), \ldots, \mathcal{A}_k(c))$$

Each BDA-generated model of grade k assigns to a codon $c \in \mathcal{B}^3$ a binary string of length k such that the j-th coordinate is 1 if c is classified by $\mathcal{A}_j$ for the dichotomic class H_1 and 0 if c is classified by $\mathcal{A}_j$ for the dichotomic class H_0. The idea behind such models is to apply successively several BDAs to the set of codons and hence partition it into a disjoint union of subsets. For instance, Fig. 1(b) shows the model generated by the three BDAs for Rumer, Parity and Complementary. In particular, only for $k \geq 6$ it is possible (but not guaranteed) that a mapping representing a BDA-generated model is bijective, hence classifies the codons uniquely, or for $k = 5$ is the minimum number of BDAs required to create 20 or 21 classes for an amino acid mapping. We will show in the sequel that indeed there are plenty of combinations of six BDAs such that a codon can be uniquely identified just by telling to which dichotomic classes generated by the six BDAs it belongs and we will also show a mapping for amino acids.

Remark 4. It is easy to see that for a given $k \in \mathbb{N}$ there are $c(k) = \binom{432}{k}$ different BDA-generated models of grade k. For instance, $c(2) = 93096$, $c(3) = 13343760$ and $c(6) \simeq 9 \cdot 10^{13}$.

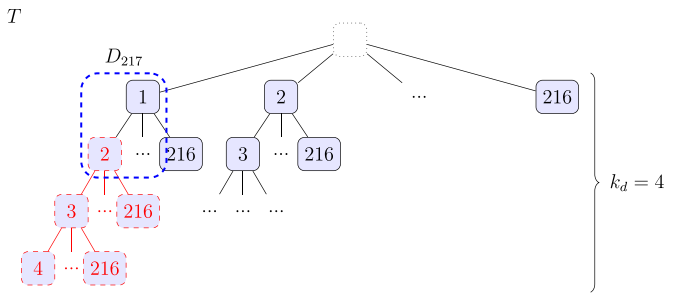


Fig. 2. Example of a tree T with a maximum BDA-set size of $k_d = 4$. Path D_{217} consisting of the nodes (BDAs) 1, 2 may have an invalid model. Thus, the subtree (red, dashed line) can be truncated.

We now define an equivalence relation on the set of BDA-generated models of the genetic code since obviously permuting the order of the BDAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$ that generate a model formally gives a new model but essentially contains the same information about the codons.

Let us consider the symmetric group $(S_k, \circ)$ where

$$S_k := \{\pi : \{1, \ldots, k\} \to \{1, \ldots, k\} : \pi \quad \text{is} \quad \text{bijective}\}$$

and $\circ$ denotes the composition of mappings.

Lemma 2.5. *Let $M : \mathcal{B}^3 \to \{0,1\}^k$ be a BDA-generated model of the genetic code of grade k, generated by the BDAs $\mathcal{A}_1, \ldots, \mathcal{A}_k$, and let $\pi \in S_k$. Then the mapping $M_\pi : \mathcal{B}^3 \to \{0,1\}^k$ with*

$$M_\pi(c) = (\mathcal{A}_{\pi(1)}(c), \mathcal{A}_{\pi(2)}(c), \ldots, \mathcal{A}_{\pi(k)}(c))$$

is also a BDA-generated model of the genetic code of the same grade k.

Proof. Let M be generated by the BDAs $\mathcal{A}_j, j \in \{1, \ldots, k\}$. Then M_π is generated by the BDAs $\mathcal{A}_{\pi(j)}, j \in \{1, \ldots, k\}$. □

By the above Lemma 2.5 the symmetric group defines an equivalence relation on the set of BDA-generated models by saying that two BDA-generated models M and N of the same grade k are **equivalent** if there exists a permutation $\pi \in S_k$ such that $M = N_\pi$. For our purposes, it will be mostly enough to restrict ourselves to equivalence classes of BDA-generated models and all the algorithms developed are designed under this assumption. We are now interested in the redundancy of BDA-generated models or equivalently, in the number of classes (subsets) into which they partition the set of codons.

Definition 2.6. Let $M : \mathcal{B}^3 \to \{0,1\}^k$ be a BDA-generated model of the genetic code of grade k. We will say that two codons c_1 and c_2 belong to the same **class** if their images under M are equal, i. e. $M(c_1) = M(c_2)$. The class of a codon c, i. e. the set of all codons that belong to the same class as c, is denoted by **class(c)**. Moreover, by $|M|$ we will denote the number of classes of M, i. e. $|M| = |\{class(c) : c \in \mathcal{B}^3\}|$.

3. Scan algorithm as implemented in Beady-A

A purely mathematical analysis of the BDA-generated models appears to be too difficult by the large variety of such models. Instead, we have developed the software Beady-A for the analysis.

3.1. Description of the algorithm

Our software is implemented in Scala Odersky et al. (2010) and uses the BioJava framework Holland et al. (2008), e. g. for the genetic code tables integration. The overall idea of this so-called scan algorithm is to iterate over all possible BDA-sets $S = D_1, D_2, \ldots, (D_i \subseteq \mathcal{D})$ where the size $k_i = |D_i|$ of any BDA-set must not exceed an intended maximal size k_d. For each BDA-set D_i its model M_i is computed. Any *interesting* model out of the S is called a solution which is reported. What is considered to be a solution can be defined. For example, a solution could be a model that maps the 64 codons to the 20 amino acids or to 64 dichotomic partitions. Let $P = \{1, 2, \ldots, 216\}$ where each number represents a BDA $\mathcal{A}_i$ and where complementary BDAs are **left out**. All possible models of grade $k \leq k_d$ can be written as $S = \{\wp(P) : |\wp(P)| \leqq k_d\}$ with $\wp$ as the power set. Thus, our algorithm computes the power set of all subsets of maximal size k_d.

Algorithm 1. Scan algorithm.

```
function create(D, h, w)
        ▷Arguments: D: A set of BDAs, h: begin of BDA-range, w: scan parameter
  if |D| < k_d then                    ▷Max. scan depth reached?
    for i ← h...|D*| do                ▷Ignore already computed BDAs
      A ← D*_i                         ▷A (new) candidate
      if A ∉ D then
            ▷Check if it is not already in the bda set.
          ▷Might happen as we can pass in fixed BDAs.
        D' ← D ∪ A
        (valid, wn)← createModel(D', w)
        if valid then create(D', i+1, wn)
        end if                         ▷Configuration valid: search is continued
      end if
    end for
  end if
end function
```

S is created by calling recursively the *create*-function as shown in listing 1 (see also listings 2 and 3 in appendix). The first call of *create* is either initialized with an empty BDA-set or with a list of fixed BDAs. Fixed BDAs are BDAs that should always be included in the model because they are supposed to be of interest. With each recursion a new BDA $\mathcal{A}$ is added to D. Each BDA is numbered from 1 to 216 and is represented as an array index (details see listing 2 in appendix). On recursion level 1, the BDA-numbers run from $i = 1, 2, \ldots, 216$, on level 2 from $i = 2, 3, \ldots, 216$, etc. in order to get a set and thus to avoid a redundant assignment of BDAs. While creating S within the function *create*, it is checked that any fixed BDA is skipped and not included twice in S. The recursion always terminates if the maximum BDA-set size k_d has been reached. The construction of S can be considered as a tree T where each node represents a BDA and each path from the root element to an inner node or leaf is a BDA-set D. The root node represents an empty BDA-set. The height of T is $k_d + 1$. An example is shown in Fig. 2.

As seen in Remark 4, the number of models generated by a BDA-set of size k is extremely huge.

Lemma 5. *The scan algorithm has a run-time complexity of $O(k_d^{32})$.*

Proof. The run-time depends exclusively on the maximum size of the BDA-set (k_d). Each node can be considered as a path representing a BDA-set D and the root note represents an empty set. T includes all lists of size 0, 1, 2, ... k_d. In total we have $N(k_d) = \sum_{k=0}^{k_d} c(k)$ nodes (see Remark 4). A simple calculation shows that $N(k_d) \leq 216 \cdot k_d^{32}$ holds for all $1 \leq k_d \leq 216$. □

The algorithm has a polynomial run-time complexity but this search problem is in real life hard to tackle because of the huge exponent. Luckily, in practice the run-time can be improved: A current BDA-set D of size $k < k_d$ may lead to an invalid model, i. e. the model may violate an intended structure. If so, D can be dropped and more recursive calls are not necessary. Note that the function *createModel* in listing 3 in the appendix returns true or false to indicate whether a D is a valid configuration or not. The computation of the subtree in T starting with the path D can be skipped. Thus, the list of all possible BDAs S can be shortened when these 'illegal' combinations are not considered while creating all the valid combinations.

3.2. Types of scan algorithms

Depending on the addressed question and the solution in mind, the behavior of the scan may differ. From now on, c_d denotes the intended number of classes for a solution (i. e. $2 \leq c_d \leq 64$). A so-called scan parameter w is passed in into every recursive *create* function. It can be used to decide whether a configuration is valid or not. Function *createModel* can be parametrized with the functions *isValidConfig* and *isSolution*. *isValidConfig*(M, w) indicates whether the model M is a valid configuration. If not, the recursive scan is terminated and the current BDA of the BDA-set is skipped. Parameter w can be used to decide if the configuration is valid. *isSolution*(M) indicates whether the model M is a solution or not.

3.2.1. IncreaseScan

IncreaseScan is the most generic scan type. However, even in the generic case, subtrees of T can be pruned. A configuration D can always be left out if the number of classes c has not increased relative to its previous BDA-set D^*. The scan parameter w is the number of classes c. The scanning is continued when the following condition is true:

$$isValidConfig(M, w) := |M| > w$$

An intended solution could be a model with a specific number of classes c_d:

$$isSolution(M) := (|M| = c_d)$$

3.2.2. Power2Scan

Power2Scan can be used when the number of classes $|M|$ should be a power-of-2-number depending on the size k_d. A solution is:

$$isSolution(M) := (|M| = 2^{k_d})$$

For instance, $k_d = 6$ can be used when 64 classes should be generated. The scanning is continued when the following condition is true:

$$isValidConfig(M, w) := (|M| = 2^k)$$

Thus, with every step (or for each $k = 1, 2, \ldots, k_d$) the number of classes must be doubled. Note that the number of codons of a class is not considered here. Also, the scan parameter w is not used. Clearly, *Power2Scan* truncates invalid subtrees very efficiently.

3.2.3. AminoCompatibilityScan (error measurement)

AminoCompatibilityScan scans for solutions where the classes are to some extent compatible with amino acids. I. e. a perfect solution contains 21 classes and all codons of the same class refer to the same amino acid or the stop signal.

In Section 4.2 we will show (compare Proposition 7) that it is impossible to find a BDA-generated model which maps all amino acids exactly. For this reason we will define an error E that indicates how good the codon's classes of a BDA-generated model match the 20 amino acid classes

Let $C \subseteq \{0, 1\}^k$ be a subset of all BDA-classes of length k and $\alpha : A \to C$ be a mapping of amino acids to the BDA-classes generated by a BDA-set of size k. Example: $\alpha(\text{Pro}) = \{11, 10\}$ and $\alpha(\text{Phe}) = \{10\}$ for $\mathcal{A}_1 = \text{Rumer}$, $\mathcal{A}_2 = \text{Parity}$. Here a Proline (Pro) is mapped to two classes whereas Phenylalanine (Phe) is only mapped to one class. Clearly, a BDA-set can be ignored if an amino acid is coded in more than one class, i. e. $\exists a : |\alpha(a)| > 1$. Two distinct BDA-classes can never be merged when applying more BDAs. Hence, the amino acid a (like Proline in the example above) will always be coded by two different classes which is a contradiction to the intended genetic code table. We like to tolerate some errors with the compatibility and introduce an error E ($0 \leq E < 1$) which indicates how compatible a code table is compared to the amino acid mappings. $E = 0$ means no error and thus full compatibility.

Given a model M, we list for every amino acid a the list C_a of codons associated with it. The length of C_a, written as $|C_a|$, is the degeneracy of a. Note that, if the model M would map the correspondence between the codons and their amino acids exactly, C_a would only contain elements of the same class. However, because of the impossibility to construct such a model (compare Proposition 7 again), C_a may contain elements of different classes. We will consider in each case all classes which contain the elements of C_a, identify the class with the biggest group of the elements of C_a and label this class as the correct one for the given amino acid a. Let u_a be the number of elements of C_a in the correct class for the amino acid a. We define the error for the amino acid a in the model M as

$$E_a = \frac{|C_a| - u_a}{64}$$

and the overall error for all codons in the model M as

$$E = \sum_a E_a.$$

Let E_d ($0 \leq E_d$) be the maximal tolerated compatibility error. A model is called E_d-compatible or short compatible if $E \leq E_d$. For instance, if we set $E_d := 0$ it means that only solutions which are perfectly compatible are searched.

The scan algorithm can be constrained like this. A solution is:

$$isSolution(M) := (M\text{iscompatible} \quad \wedge \quad |M| = c_d)$$

The scanning is continued when the following condition is true:

$$isValidConfig(M, w) := (M\text{iscompatible} \quad \wedge \quad w < |M| \leq c_d)$$

Thus, as long as any "interim" model ($|M| < c_k$) is compatible, the search is continued.

3.2.4. Search space

The search space for BDA-generated models is extremely large. We have limited the bit-length in our analysis mostly to values $k \leq 10$. Yet, the number of interesting models which were detected is still quite high. Scan methods based on the power-2-approach have a good run-time performance (in the scale of seconds to minutes on a regular work station) whereas others are very time consuming (days to weeks). We have also implemented a concurrent version of the scan-algorithm which is not described in detail in this paper. Depending on the scan-type, the search can be accelerated linearly by the number of processors or cores available.

4. Results and discussion

4.1. Possible number of classes $|M|$

As a first analysis we are interested in the power of BDA-sets. In particular, it was analyzed if it is possible to create models with any number of classes ranging from 2 to 64. For instance, the model shown in Fig. 1(b), which uses the Rumer, Parity and Complementary BDAs, divides the 64 codons into 8 classes. *IncreaseScan* was modified in the following way: The algorithm maintains a set R of numbers that was initialized with $R = \{2, 3, \ldots, 64\}$. If a model has a number of classes $|M| \in R$ a solution is found and $|M|$ is removed from R.

A scan with a maximum BDA-set-size of $k_d = 6$ revealed that all class-numbers $|M|$ are possible except for $|M| \in \{3, 51, 53, 54, 55, 57, 58, 59, 61, 62, 63\}$. When allowing an additional BDA, i. e. setting $k_d = 7$, all class numbers except for $|M| \in \{3, 63\}$ could be generated. This led to the question whether 3 and 63 are special class-numbers which can never be achieved. Indeed, we now show that

Proposition 6. *It is not possible to create 3 or 63 classes with BDAs.*

The proof is available in the appendix. An important outcome of this analysis is that it is possible to divide the set of codons into $|M| = 21$, $|M| = 20$, $|M| = 24$ or $|M| = 64$ classes. 21 classes could be used to classify the 20 amino acids and the stop signal. If we assume that the class of a stop codon does not matter as the tRNA is missing, 20 classes could also be considered. As we will see, 24 classes are also of interest as the model by Gonzalez (2008) also uses 24 classes for the amino acids. Here, amino acids of degeneracy 6 are split into two classes of size 4 and 2. Finally, a bijective mapping between 64 codons and their 64 classes is also relevant since this gives an algorithmic way to determine a codon uniquely by using BDAs. We will discuss later whether a BDA mechanism can actually be performed by the ribosome. Moreover, an exact matching of codons and classes of the model will provide new mathematical models that allow applying results and techniques from coding theory, for instance.

4.2. Mapping 64 codons to 20 amino acids and one stop signal with $k \geq 5$ BDAs

AminoCompatibilityScan was used to look for models that map the codons to 20 amino acids and the stop signal. The number of BDAs has to be $k \geq 5$ in order to get 21 or more generally 20 to 24 classes. Interestingly, our search never came across a solution for a class number of $|M| = 21$ that only used $k_d = 5$ BDAs. Instead, $k = 6$ is the minimum number of BDAs required. Our analysis shows that **no solutions** could be found for a compatibility error $E_d = 0$ and a class number $|M| = 21$, no matter what genetic code table was used. A further analysis revealed that the nature of a BDA does not allow us to correctly create classes that respect the amino acid distribution of the last quartet row (codons of the type *Gxx*) in the code table. Indeed, the following proposition shows that

Proposition 7. *There is no BDA-generated model such that codons of each class map a unique amino acid and codons from different classes different amino acids.*

Proof. Let us assume that there is a BDA-generated model such that codons of each class correspond to the same amino acid and codons from different classes correspond to different amino acids. We consider all codons of the form G*XY* with $X, Y \in \mathcal{B}$. For a given $X \neq$ A, the four codons of the form G*XY* with $Y \in \mathcal{B}$ code the same amino acid (Val, Ala and Gly). Thus, there must exist one BDA $\mathcal{A}$ from the model producing a dichotomic partition (H_0, H_1) such that $\mathcal{A}$ classifies the codons corresponding to the amino acids Val, Ala and Gly to one of the classes H_0 or H_1, respectively but distributes the codons of the form GA*Y* equally into both classes. Now assume that $i_1 = 1$ for $\mathcal{A}$. Then question Q_1 can not contain G since then all codons of the form GA*Y* would be classified to the same class – a contradiction. However, in the other case question Q_2 then either classifies all codons of the form GA*Y* to the same class (if $i_2 = 2$) or does not classify the codons corresponding to the amino acids Val, Ala and Gly to the same class, respectively (if $i_2 = 3$) - a contradiction. Now, let us assume that $i_1 = 2$. Then question Q_1 can not contain A since otherwise the four codons of the form GA*Y* would be classified to the same class - a contradiction. If A is not in Q_1, then similar arguments as above lead to a contradiction. Finally, if $i_1 = 3$, then question Q_1 already divides the codons corresponding to the amino acids Val, Ala and Gly to two classes – a contradiction. □

When the maximal compatibility error E_d was set to a value greater than 0 and the number of classes were allowed to be in a range from 20 to 24, we obtained BDA generated models that are almost able to partition the amino acids correctly. For a class number of $|M| = 24$ the minimum error is $E = 0.171875 = 11/64$, indicating that 11 codons do not correctly match their amino acids or stop signal. Here several different models exist and Table 1 depicts an example of such a model that has 24 classes generated by 7 BDAs. Note that this partition includes the Rumer dichotomy.

Table 1
24 classes generated by seven BDAs including Rumer with $E = 11/64$.

	U		C		A		G		
U	Phe	1111010	Ser	0111101	Tyr	1111001	Cys	1111111	U
U	Phe	1111010	Ser	0111101	Tyr	1111001	Cys	1111111	C
U	Leu	1111010	Ser	0111101	!	1111000	!	1111110	A
U	Leu	1111010	Ser	0111101	!	1111000	Trp	1111110	G
C	Leu	0010000	Pro	0100001	His	1001001	Arg	0111001	U
C	Leu	0010000	Pro	0100001	His	1001001	Arg	0111001	C
C	Leu	0010000	Pro	0100001	Gln	1001000	Arg	0111000	A
C	Leu	0010000	Pro	0100001	Gln	1001000	Arg	0111000	G
A	Ile	1000010	Thr	0000101	Asn	1000001	Ser	1000111	U
A	Ile	1000010	Thr	0000101	Asn	1000001	Ser	1000111	C
A	Ile	1000010	Thr	0000101	Lys	1000000	Arg	1000110	A
A	Met	1000010	Thr	0000101	Lys	1000000	Arg	1000110	G
G	Val	0010110	Ala	0100111	Asp	1001111	Gly	0111111	U
G	Val	0010110	Ala	0100111	Asp	1001111	Gly	0111111	C
G	Val	0010110	Ala	0100111	Glu	1001110	Gly	0111110	A
G	Val	0010110	Ala	0100111	Glu	1001110	Gly	0111110	G

BDA	(i_1, i_2)	Q_1	Q_2
Rumer	(2, 1)	(C, A)	$\{C, G\}$
$\mathcal{A}_2$	(1, 2)	(A, U)	$\{A, U\}$
$\mathcal{A}_3$	(1, 2)	(A, U)	$\{A, C\}$
$\mathcal{A}_4$	(1, 2)	(A, U)	$\{U, C\}$
$\mathcal{A}_5$	(1, 2)	(C, G)	$\{A, U\}$
$\mathcal{A}_6$	(1, 2)	(C, G)	$\{A, C\}$
$\mathcal{A}_7$	(2, 3)	(U, C)	$\{A, G\}$

All codons of the second and third column, e. g. those of type *x*C*x* or *x*A*x* are correctly mapped to their amino acids. The code table is incorrect for Glycine (Gly, codons GG*x*) as this 4-degeneracy amino acid is split into two classes 0111111 and 0111110. This is an example of Proposition 7. Although their classes are different, they only differ in one digit. Also, the 6-degeneracy amino acids Leucine (Leu) and Serine (Ser) are incorrectly assigned to two classes of size 2 and 4: Leucine is divided into the classes 1111010 and 0010000 and Serine is divided into the classes 0111101 and 1000111. The 6-degeneracy amino acid Arginine (Arg, codons CG*x* and AG*x*) is even split into three classes 0111001, 0111000 and 1000110 each of size 2. Finally, Methionine (Met) was assigned to the same class as Isoleucine (Ile). These exceptions of having four more classes and one merged lead to a class number of 24 instead of 21. The class for the stop signal UGA is the same class as Tryptophan (Trp). However, this incorrect classification could be ignored as there is no tRNA for the stop signal codon UGA.

We have analyzed how likely such an optimal model is. As the overall number of models is extremely huge, we have randomly picked 5000 models out of the set of models that generate $|M| = 24$ classes and have $k = 5$ to 9 different BDAs. The distribution of the E values is shown in Fig. 3 on the left. The likelihood P observed for a model with $E = 11/64$ was $P < 0.0005$. However, when Rumer is included as a fixed BDA and the size is set to $k = 7$ (as in Table 1), optimal models are observed (cf. 3, right). Note that the mean E value is shifted from 0.5312 towards 0.3281 when Rumer is applied. This again indicates that Rumer might play a special role.

The dichotomic partition Parity was never selected by our scan algorithm while looking for solutions with $E_d = 0.2$. That is because Parity partitions the codons in a way that is not suitable for the amino acid assignment. As seen in Fig. 1(b), Parity splits the quartets of codons that have the same first and second base into two classes of size 3 and 1. However, the standard genetic code mostly maps those codons to the same amino acid or to two species but only one time to 3 and 1 (Isoleucine and Methionine). Thus,

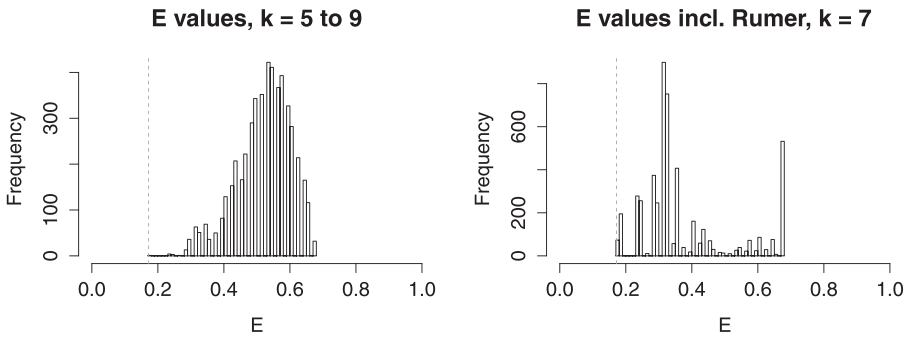


Fig. 3. Distributions of the compatibility error E for 5000 random models that have $|M| = 24$ classes. The bin size is 1/64. Left: $k = 5$, 6, 7, 8 or 9 randomly chosen BDAs are used. Right: Rumer-BDA is always included in a model plus 6 randomly chosen BDAs. The dashed line shows the value for the optimal model $E = 11/64 = 0.171875$.

Table 2
$|M| = 64$ classes generated by six BDAs including Rumer.

	U		C		A		G		
U	Phe	110111	Ser	010111	Tyr	100111	Cys	111111	U
U	Phe	110100	Ser	010100	Tyr	100100	Cys	111100	C
U	Leu	110001	Ser	010001	!	100001	!	111001	A
U	Leu	110101	Ser	010101	!	100101	Trp	111101	G
C	Leu	000110	Pro	011110	His	101110	Arg	001110	U
C	Leu	000010	Pro	011010	His	101010	Arg	001010	C
C	Leu	000000	Pro	011000	Gln	101000	Arg	001000	A
C	Leu	000011	Pro	011011	Gln	101011	Arg	001011	G
A	Ile	110110	Thr	010110	Asn	100110	Ser	111110	U
A	Ile	110010	Thr	010010	Asn	100010	Ser	111010	C
A	Ile	110000	Thr	010000	Lys	100000	Arg	111000	A
A	Met	110011	Thr	010011	Lys	100011	Arg	111011	G
G	Val	000111	Ala	011111	Asp	101111	Gly	001111	U
G	Val	000100	Ala	011100	Asp	101100	Gly	001100	C
G	Val	000001	Ala	011001	Glu	101001	Gly	001001	A
G	Val	000101	Ala	011101	Glu	101101	Gly	001101	G

BDA	(i_1, i_2)	Q_1	Q_2
Rumer	(2, 1)	(C, A)	$\{C, G\}$
$\mathcal{A}_2$	(2, 1)	(A, C)	$\{G, C\}$
$\mathcal{A}_3$	(2, 1)	(U, G)	$\{A, U\}$
$\mathcal{A}_4$	(3, 1)	(A, U)	$\{A, C\}$
$\mathcal{A}_5$	(3, 1)	(A, U)	$\{U, G\}$
$\mathcal{A}_6$	(3, 1)	(C, G)	$\{A, C\}$

Table 3
64 classes generated by 7 BDAs that include Rumer, Parity and Complementary.

	U		C		A		G		
U	Phe	1111010	Ser	0011010	Tyr	1011010	Cys	1111110	U
U	Phe	1111011	Ser	0011001	Tyr	1011001	Cys	1111111	C
U	Leu	1101001	Ser	0101000	!	1101000	!	1101101	A
U	Leu	1001011	Ser	0001000	!	1001000	Trp	1001111	G
C	Leu	0100010	Pro	0001110	His	1000110	Arg	0101110	U
C	Leu	0100011	Pro	0001101	His	1000101	Arg	0101111	C
C	Leu	0100001	Pro	0101100	Gln	1100100	Arg	0101101	A
C	Leu	0000011	Pro	0001100	Gln	1000100	Arg	0001111	G
A	Ile	1110010	Thr	0010010	Asn	1010010	Ser	1110110	U
A	Ile	1110011	Thr	0010001	Asn	1010001	Ser	1110111	C
A	Ile	1100001	Thr	0100000	Lys	1100000	Arg	1100101	A
A	Met	1000011	Thr	0000000	Lys	1000000	Arg	1000111	G
G	Val	0110010	Ala	0011110	Asp	1010110	Gly	0111110	U
G	Val	0110011	Ala	0011101	Asp	1010101	Gly	0111111	C
G	Val	0110001	Ala	0111100	Glu	1110100	Gly	0111101	A
G	Val	0010011	Ala	0011100	Glu	1010100	Gly	0011111	G

BDA	(i_1, i_2)	Q_1	Q_2
Rumer	(2, 1)	(C, A)	$\{C, G\}$
Parity	(3, 2)	(G, A)	$\{C, A\}$
Complementary	(1, 3)	(C, G)	$\{G, A\}$
$\mathcal{A}_4$	(1, 2)	(A, U)	$\{A, U\}$
$\mathcal{A}_5$	(2, 1)	(U, G)	$\{A, U\}$
$\mathcal{A}_6$	(3, 2)	(A, U)	$\{A, C\}$
$\mathcal{A}_7$	(3, 2)	(U, C)	$\{A, C\}$

Parity leads to an increase of the compatibility error E. Our scan also never reported a solution where Rumer-BDA and Complementary-BDA were included as fixed BDAs and the compatibility error was $E \leq 11/64$.

4.3. Mapping 64 codons to 64 classes with $k \geq 6$ BDAs

As seen in Section 4.1, it is possible to create 64 classes, i. e. each codon can match a class uniquely. The analysis can be done quite quickly with *Power2Scan*. The scan showed that there are 53856 solutions for a BDA-set of size $k = 6$. An example that includes the Rumer-BDA is shown in Table 2.

As the BDAs Rumer, Parity and Complementary play a special role in the mathematical modeling of the genetic code, it was analyzed how many solutions exist that contain one of the three specific BDAs or even combinations of them. There exist no solutions for any combinations of two or more specific BDAs when the BDA-set size is restricted to $k_d = 6$. Interestingly, there are always only 36 BDAs used (out of $|\mathcal{D}^*| = 216$ possible) when one of the special BDAs is included. The BDA approach seems to generate symmetrical solutions.

It is not possible to create 64 classes that include Rumer, Parity and Complementary with $k_d = 6$ BDAs. However, it is possible with $k_d = 7$ as shown in Table 3. It remains to be shown why adding redundancy enables the integration of more fixed BDA.

4.4. Reading head compatible BDAs

Another approach to screen for interesting BDAs is to consider functional aspects of the decoding process occurring in the decoding center of the ribosome.

A correct Watson-Crick base-pairing between the codon and the tRNA anticodon in the first two bases of the codon are essential for the decoding process (Schmeing and Ramakrishnan, 2009). Functional and structural evidences indicate that in these two positions the interactions made by universally conserved bases of the 16S ribosomal RNA closely monitor base-pairing geometry during decoding (Ogle et al., 2001). In particular, A1492 and A1493 form a reading-head structure able to monitor the correct base-pairing of the first two bases of the codon, through readout of the minor-groove of the codon-anticodon mini-helix, forming locally a triple-helix. These interactions appear to govern domain closure of the 30S subunit (Ogle et al., 2002), accelerating the forwards steps in decoding (Gromadski and Rodnina, 2004).

The evidence of minor-groove inspection of the codon-anticodon mini-helix by the A1492-A1493 reading head bears interesting implications: because of nucleoside biochemistry, only two possible hydrogen-bonding profiles are formed in the minor groove out of the four different possible base pairs for each position. In particular, weak base-pairs (either A-U or U-A) are indistinguishable one from another in the minor groove. Strong base-pairs (either C-G or G-C) are also indistinguishable one from another in the minor groove, but display a different profile of hydrogen bonds compared to weak base pairs (Masliah et al., 2013).

From a theoretical point of view, the reading head formed by A1492-A1493 could dichotomically discriminate between weak (A, U) and strong (C,G) base-pairs both in the first and in the second position of codon-anticodon mini-helix. Thus, we can consider BDAs that inquire about the weak or strong nature of the codon in position 1 and/or 2 as being biologically meaningful in decoding.

The flexibility of spatial organization in the third position of the codon constitutes another key ingredient in decoding, determining acceptance of degeneracy and wobble-pairing (Demeshkina et al., 2012, 2013). Chemical modifications of the anticodon stem loop and in the wobble position consistently affect maintenance of the translational reading frame (Atkins and Björk, 2009), as well as codon selection (Agris et al., 2007). Isoaccepting tRNAs that are able to decode different codons, regardless of standard Watson-Crick base-pairing rules, are an illuminating example to date (Voorhees et al., 2013). This indicates that the decoding process is influenced by conformational read-out of the third codon base, rather than depending on a strict complementarity of the codon-anticodon pairing. In fact, all amino acids with degeneracy two are discriminated based on purine or pyrimidine in the third codon position. Purines (A, G) and pyrimidines (U, C) can neither be reciprocally distinguished from the energy of hydrogen bonding with the complementary base (weak/strong), nor from the hydrogen bond acceptor/donor properties (keto/amino). Thus, together with the BDAs mimicking the reading-head properties of the A-triple minihelix described before, we can regard BDAs that ask for the conformation of the nucleoside (purine/pyrimidine R/Y) in the third position of the codon as functionally purposeful and ribosome-compatible. We will refer jointly to the latter as reading-head BDAs.

Interestingly, we were able to map all codons of the genetic code to 64 classes using only reading-head BDAs (see Table 4), suggesting that the algorithm may be useful to faithfully reproduce and further explore the logic behind the decoding process.

Table 4
64 classes generated by 7 reading-head-compatible BDAs.

	U		C		A		G		
U	Phe	1111111	Ser	1010110	Tyr	1101111	Cys	1011110	U
U	Phe	1111001	Ser	1010000	Tyr	1101001	Cys	1011000	C
U	Leu	1111010	Ser	1010010	!	1101010	!	1011010	A
U	Leu	1111011	Ser	1010011	!	1101011	Trp	1011011	G
C	Leu	1010111	Pro	0000110	His	1000111	Arg	0001110	U
C	Leu	1010001	Pro	0000000	His	1000001	Arg	0001000	C
C	Leu	1010100	Pro	0000100	Gln	1000100	Arg	0001100	A
C	Leu	1010101	Pro	0000101	Gln	1000101	Arg	0001101	G
A	Ile	0111111	Thr	0010110	Asn	0101111	Ser	0011110	U
A	Ile	0111001	Thr	0010000	Asn	0101001	Ser	0011000	C
A	Ile	0111010	Thr	0010010	Lys	0101010	Arg	0011010	A
A	Met	0111011	Thr	0010011	Lys	0101011	Arg	0011011	G
G	Val	1110111	Ala	0100110	Asp	1100111	Gly	0101110	U
G	Val	1110001	Ala	0100000	Asp	1100001	Gly	0101000	C
G	Val	1110100	Ala	0100100	Glu	1100100	Gly	0101100	A
G	Val	1110101	Ala	0100101	Glu	1100101	Gly	0101101	G

BDA	(i_1, i_2)	Q_1	Q_2
$\mathcal{A}_1$	(1, 2)	(A, U)	$\{C, G\}$
$\mathcal{A}_2$	(1, 2)	(C, G)	$\{C, G\}$
$\mathcal{A}_3$	(2, 1)	(A, U)	$\{C, G\}$
$\mathcal{A}_4$	(2, 1)	(C, G)	$\{C, G\}$
$\mathcal{A}_5$	(3, 1)	(C, U)	$\{A, U\}$
$\mathcal{A}_6$	(3, 1)	(C, U)	$\{C, G\}$
$\mathcal{A}_7$	(3, 2)	(A, G)	$\{C, G\}$

5. Conclusions

In this work a new approach for a mathematical description of the structure of the genetic code is presented, based on the concept of binary dichotomic algorithms (BDAs) as introduced in Fimmel et al. (2013). The analysis shows that through BDAs it is possible to divide 64 codons into different classes by binary decisions. This is not naturally a result you may expect as BDAs are only able to partition the 64 codons in a very limited way compared to the overall possible dichotomic partitions (see Remark 3). A binary yes-no decision is the most parsimonious classification nature can apply. In this respect a BDA-set represents a very concise and efficient way for classification: in fact, an overlapping of binary decisions can create 21 (or with tolerated errors: 24) classes. Moreover, although there exists no BDA-generated model that exactly classifies codons according to their amino acids, BDAs are able to create an informational structure which is relatively close to the universal genetic code table.

It is remarkable how universal and flexible BDA-generated models are. These models may help to understand the structure of the genetic code and its singular degeneracy. For instance, Shaul and co-workers (Shaul et al., 2010) have analyzed, by means of classification and regression trees (CART), the acceptor stems of tRNAs and their role in classifying the amino acid. A similar analysis could be performed for the genetic code table to derive classifiers similar to BDAs, or even BDA-compatible classifiers.

The scans revealed that there are relevant models which have to have more BDAs than necessary in order to divide the codons into classes which represent amino acids or 64 classes. This leads us to the question if such redundancy may be used within the genetic code for some sort of error detection or even error correction. In analogy, a similar use of the redundancy has been proposed for circular codes, putative remnants of primeval comma-free codes detected in coding sequences (Michel, 2012; Fimmel et al., 2013). Similar considerations on the circularity of the rRNA nucleotide sequences in the decoding center of the ribosome were recently elaborated (Soufi and Michel, 2014). Although a perfect mapping to amino acids is not possible we speculate that codons may be grouped to amino acids such that codons of each group have a smaller Hamming distance than codons between the groups. If we, for example, assume a greater BDA-set of size, let us say, $k = 12$ for the classification of the 20 amino acids, it might happen that the model clusters amino acids whose binary representations have a considerably smaller Hamming distance within this group than between different amino acids.

In conclusion, we have developed an algorithm that scans for significant models of the genetic code generated by BDAs. The findings show that there are models that describe the degeneracy of the code with only little error. There are also models that map all 64 codons uniquely to 64 classes, including only BDAs that may reflect biological processes taking place in the ribosome. This indicates that BDAs could serve as a basis for further theoretical analyses or implementations of coding theory to study of the genetic code.

Competing interests

The authors declare that they have no competing interests.

Author's contributions

MG developed the Beady-A tool along with the scan algorithm, did the search for relevant models and drafted the main manuscript. EF and LS introduced the concept of a BDA-generated model and wrote the mathematics part. AD performed the analysis of BDAs applied on the ribosome and wrote the biological parts. All authors read and approved the final manuscript.

Acknowledgments

We would like to thank Ivo Wolf, Simone Giannerini and Diego Luis Gonzalez for stimulating discussions.

Appendix

Pseudo-Code: Listings 2 and 3 show additional pseudo code of the scan-algorithm.

Algorithm 2. Declaration of all BDAs that are used.

```
nucs ← Array(Adenine, Uracil, Cytosine, Guanine)      ▷Four nucleotides

Q1Nucs ← List((0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3))
      ▷ Index for array nucs. Just half of all combinations as the missing
          nucleotide-tuples would lead to a complement partition.

Q2Nucs ← List((0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3))
▷Index for array nucs. Only have half of all elements as they define a set.

positions ← List((0, 1), (0, 2), (1, 2), (1, 0), (2, 0), (2, 1))

    ▷Positions of base within codon where Q1 and Q2 is asked

D* ← createAllBdas()      ▷216 different BDAs

function createAllBdas
  bdaList← empty list of BDAs
  for all (i1, i2) ← positions and Q1 ← Q1Nucs and Q2 ← Q2Nucs do
    A ← BDA with positions (i1, i2) and questions Q1 and Q2
    Add A to bdaList
  end for
  return bdaList as array
end function
```

Algorithm 3. Creation of a model M. Details are omitted.

```
function createModel(D, w)
  M← Create new model using the BDA set D
  valid← isValidConfig(M, w)          ▷Check if model is valid
  if valid and isSolution(M) then
    output(M)          ▷Report this solution
  end if
  return (valid, scanParameter(M))
end function
```

Proof for Proposition 6

Proof. Assume that we have applied a list $(A_1, \ldots, A_k)$ of k BDAs to obtain the partition $\mathcal{B}^3 = \bigcup_{i=1}^{n} X_i$ which is different from the one produced by the list $(A_1, \ldots, A_{k-1})$. Let us first deal with the case of three classes, i. e. $n = 3$. W. l. o. g. the list $(A_1, \ldots, A_{k-1})$ produces the partition $\mathcal{B}^3 = Y_1 \cup Y_2$ and assume that the last BDA A_k has the dichotomic partition (H_0, H_1). It follows that

$$\mathcal{B}^3 = X_1 \cup X_2 \cup X_3 = (Y_1 \cap H_0) \cup (Y_1 \cap H_1) \cup (Y_2 \cap H_0) \cup (Y_2 \cap H_1)$$

and hence we must have that one of the intersections $(Y_i \cap H_j)$ is the empty set, say $Y_1 \cap H_0 = \emptyset$. But then $Y_1 \subseteq H_1$ and since both sets have size 32 we conclude $Y_1 = H_1$ and consequently $Y_2 = H_0$ which means that the list $(A_1, \ldots, A_k)$ produces the partition $Y_1 \cup Y_2$ - a contradiction. Now assume that $n = 63$. Necessarily, the list $(A_1, \ldots, A_{k-1})$ must then produce a partition $\mathcal{B}^3 = \bigcup_{i=1}^{32} Y_i$ of 32 disjoint subsets Y_i since each BDA can at most divide a set into two sets. Clearly, each set Y_i must then have 2 elements by cardinality reasons. Thus only one of the sets Y_i does not get split when applying the last BDA A_k, say Y_1. It follows that each set $H_0 \cap Y_i$ and $H_1 \cap Y_i$ for $i = 2, \ldots, 32$ has exactly 1 element. W. l. o. g. assume that $H_0 \cap Y_1 = \emptyset$ and hence $H_1 \cap Y_1 = Y_1$. But then $H_1 = \bigcup_{i=2}^{32} (H_1 \cap Y_i) \cup Y_1$ has 33 elements - a contradiction! □

References

Agris, P.F., Vendeix, F.A.P., Graham, W.D., 2007. tRNA's wobble decoding of the genome: 40 years of modification. J. Mol. Biol. 366 (1), 1–13.
Arquès, D.G., Michel, C.J., 1996. A complementary circular code in the protein coding genes. J. Theor. Biol. 182 (1), 45–58.
Atkins, J.F., 2009. A gripping tale of ribosomal frameshifting: extragenic suppressors of frameshift mutations spotlight p-site realignment. Microbiol. Mol. Biol. Rev. 73 (1), 178–210.
Crick, F.H., Griffith, J.S., Orgel, L.E., 1957. Codes without commas. Proc. Natl. Acad. Sci. U. S. A. 43 (5), 416–421.
Crick, F.H., 1968. The origin of the genetic code. J. Mol. Biol. 38 (3), 367–379.
Demeshkina, N., Jenner, L., Westhof, E., Yusupov, M., Yusupova, G., 2012. A new understanding of the decoding principle on the ribosome. Nature 484 (7393), 256–259.
Demeshkina, N., Jenner, L., Westhof, E., Yusupov, M., Yusupova, G., 2013. New structural insights into the decoding mechanism: translation in-delity via a G–U pair with Watson-Crick geometry. FEBS Lett. 587 (13), 1848–1857.
Fimmel, E., Danielli, A., Strüngmann, L., 2013. On dichotomic classes and bijections of the genetic code. J. Theor. Biol. 336 (0), 221–230.
Freeland, S.J., Wu, T., Keulmann, N., Oct 2003. The case for an error minimizing standard genetic code. Orig. Life Evol. Biosph. 33 (4-5), 457–477.
Gamow, G., 1954. Possible relation between deoxyribonucleic acid and protein structures. Nature 173, 318.
Giannerini, S., Gonzalez, D.L., Rosa, R., Jun 2012. DNA, dichotomic classes and frame synchronization: a quasi-crystal framework. Philos. Trans. A: Math. Phys. Eng. Sci. 370 (1969), 2987–3006.
Di Giulio, M., 2008. An extension of the coevolution theory of the origin of the genetic code. Biol. Direct 3, 37.
Gonzalez, D.L., Giannerini, S., Rosa, R., Nov 2008. Strong short-range correlations and dichotomic codon classes in coding DNA sequences. Phys. Rev. E: Stat. Nonlin. Soft Matter Phys. 78 (5 Pt 1), 051918.
Gonzalez, D.L., 2008. The mathematical structure of the genetic code. In: Codes of Life: The Rules of Microevolution. Springer.
Gromadski, K.B., Rodnina, M.V., 2004. Kinetic determinants of high-fidelity tRNA discrimination on the ribosome. Mol. Cell 13 (2), 191–200.
Guilloux, A., Jestin, J.-L., 2012. The genetic code and its optimization for kinetic energy conservation in polypeptide chains. Biosystems 109 (2), 141–144.
Haig, D., Hurst, L.D., 1991. A quantitative measure of error minimization in the genetic code. J. Mol. Evol. 33 (5), 412–417.
Seligmann, H., Aug 2014. Species radiation by DNA replication that systematically exchanges nucleotides? J. Theor. Biol. 363C, 216–222.
Holland, R.C.G., Down, T.A., Pocock, M., Prlic, A., Ardelluen, D.H., James, K., Foisy, S., Dräger, A., Yates, A., Heuer, M., Schreiber, M.J., Sep 2008. BioJava: an open-source framework for bioinformatics. Bioinformatics 24 (18), 2096–2097.
Jukes, T.H., Osawa, S., 1993. Evolutionary changes in the genetic code. Comp. Biochem. Physiol. B 106 (3), 489–494.
Koonin, E.V., Novozhilov, A.S., 2009. Origin and evolution of the genetic code: the universal enigma. IUBMB Life 61 (2), 99–111.
Masliah, G., Barraud, P., Allain, F.H., 2013. RNA recognition by double-stranded RNA binding domains: a matter of shape and sequence. Cell. Mol. Life Sci. 70 (11), 1875–1895.
Michel, C.J., Apr 2012. Circular code motifs in transfer and 16s ribosomal RNAs: a possible translation code in genes. Comput. Biol. Chem. 37, 24–37.
Odersky, M., Spoon, L., Venners, B., 2010. Programming in Scala, 2nd ed. Artima Developer.
Ogle, J.M., Brodersen, D.E., Clemons Jr., W.M., Tarry, M.J., Carter, A.P., Ramakrishnan, V., 2001. Recognition of cognate transfer RNA by the 30s ribosomal subunit. Science 292 (5518), 897–902.
Ogle, J.M., Murphy, F.V., Tarry, M.J., Ramakrishnan, V., 2002. Selection of tRNA by the ribosome requires a transition from an open to a closed form. Cell 111 (5), 721–732.
Osawa, S., Jukes, T.H., Watanabe, K., Muto, A., 1992. Recent evidence for evolution of the genetic code. Microbiol. Rev. 56 (1), 229–264.
Rumer, I.B., 1966. Codon systematization in the genetic code. Dokl. Akad. Nauk. SSSR 167 (6), 1393–1394.
Schmeing, T.M., Ramakrishnan, V., 2009. What recent ribosome structures have revealed about the mechanism of translation. Nature 461 (7268), 1234–1242.
Seligmann, H., Nov 2007. Cost minimization of ribosomal frameshifts. J. Theor. Biol. 249 (1), 162–167.
Seligmann, H., Oct 2014. Mitochondrial swinger replication: DNA replication systematically exchanging nucleotides and short 16s ribosomal DNA swinger inserts. Biosystems 125C, 22–31.
Sella, G., Ardell, D.H., 2006. The coevolution of genes and genetic codes: Crick's frozen accident revisited. J. Mol. Evol. 63 (3), 297–313.

Shaul, S., Berel, D., Benjamini, Y., Graur, D., 2010. Revisiting the operational RNA code for amino acids: ensemble attributes and their implications. RNA 16 (1), 141–153.
El Soufi, K., Michel, C.J., Oct 2014. Circular code motifs in the ribosome decoding center. Comput. Biol. Chem. 52, 9–17.
Voorhees, R.M., Mandal, D., Neubauer, C., Köhrer, C., RajBhandary, U.L., Ramakrishnan, V., 2013. The structural basis for specific decoding of AUA by isoleucine tRNA on the ribosome. Nat. Struct. Mol. Biol. 20 (5), 641–643.
Wong, J.T., 1975. A co-evolution theory of the genetic code. Proc. Natl. Acad. Sci. U. S. A. 72 (5), 1909–1912.