Gritstone Oncology Coding Assignment: Targeted Mutagenesis

Background: the goal is to introduce targeted mutations in a protein by using expanded genetic code, where a mixture of bases is present at a certain position. For example, codon ATB where B is a mixture of C, G and T encodes for both Ile and Met. One can incorporate such codons in a synthesized gene or oligo to get a low-complexity library. We want to modify all positions independently from each other.

Unfortunately, some expanded triplets code for more amino acids than necessary. Say, we want an amino acid to be either Ala, Ile or Val – there are many degenerate codons that can encode for all three of them. For example, RYA (R = A or G, Y = C or T), however the same codon will also encode for Thr. There are additional possibilities, such as RYT or RYC, however they will again encode for Thr.

Let's define "codon efficiency" as the number of codons translated into desired amino acids divided by the total number of all codons encoded by the degenerate codon. E.g. RYA = (ACA, ATA, GCA, GTA), i.e. 4 total. ACA is Thr, ATA is Ile, GCA is Ala and GTA is Val. If Ile, Ala and Val are desired and Thr is not, then the efficiency is 3/4 = 75%. Turns out that other possibilities such as RYT, RYC and a few others have the same 75% efficiency, we can use any of them. In the real life the choice would be driven by codon usage frequency in the target organism (yeast), but this is outside of the scope of this assignment.

Task:

1) Write a function that takes a set of amino acids (say, {'A','V','I'}) and returns the set of most efficient codons and the achieved efficiency
    a. For {A','V','I'} the answer would be
    ```
    ({'RYA', 'RYH', 'RYC', 'RYW', 'RYM', 'RYY', 'RYT'}, 0.75)
    ```

2) Write a function that takes a set of amino acids and:
    a. if they can be encoded by codons with 100% efficiency, returns the input set
    b. if they cannot be encoded by a codon with 100% efficiency, removes the minimal number of amino acids from the list such as the resulting list can be encoded with 100% efficiency. If there are multiple possibilities, return all of them.

    I.e. from {'A','V'} one would get {'A','V'} since they can be both encoded by the a number of codons with 100% efficiency (say, GYT)

    From {'A','I','V'} one would get {'I','V'} and {'A','V'} (both 100% efficient), but not {'A','I'} since {'A','I'} is only 50% efficient.

Things to consider:

1) Write everything in Python2 or 3, make use of standard libraries such as collections and itertools. To save time, we're providing a Python file with genetic code already typed in
2) In part (1), how do you want to handle situations when the resulting codon also encodes for stop? No single right answer here.

3) Assume that this function will be called >100 times from a parent routine. Think about precomputing or caching things, i.e. some of the results you computed for position 1 might be reused in calculating things for position 2.

Thank you for your time and considering Gritstone Oncology!