ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB406

SEARCH FOR NEW ISOTOPE PRODUCTION PATHWAYS*

L. F. Dabill, Coe College, Cedar Rapids, United States
A. Hutton, Jefferson Lab, Newport News, United States

Abstract

The isotope group at Jefferson Lab is carrying out R&D for producing medically interesting radioisotopes, especially those with theranostic (therapeutic and diagnostic) attributes. Here the search for viable production mechanisms has been expanded to multi-step reactions where a daughter is produced from the target and decays into a medically interesting granddaughter radioisotope. It is difficult to find efficient production routes when investigating both the initial excitation reaction as well as the decay routes leading to medically interesting isotopes. The overall goal of this project is to create a structured code in python to find these decay routes by automatically exploring the large number of isotopes and their possible decay modes. The program structure is described, and preliminary results are presented.

INTRODUCTION

Background

Radioisotope production through a gamma ray production route must consider the likelihood and percentage of a reaction before choosing an appropriate target and method. One such consideration is the Q-value of a reaction. A reaction route that has a low Q-value is desirable because the reaction needs a lower amount of energy to produce the desired isotope from the target isotope. Having a low energy requirement for the reaction is especially interesting for medical applications because the desired medical radioisotope will be produced in a higher quantity. A graph that depicts the desirable zone of production in terms of Q-value is depicted in Fig. 1.

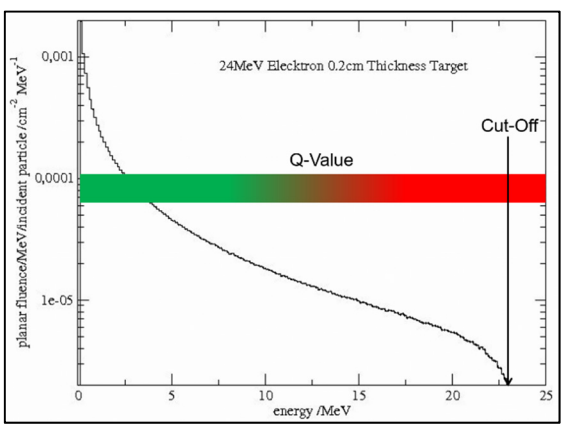

Figure 1: Q-value chart.

*Work supported by Old Dominion University REU through NSF Grant, and by the U.S. DOE, Office of Science, Office of Nuclear Physics under contract DE-AC05-06OR23177.

The reaction route that commonly has the lowest Q-value is through a gamma-neutron reaction in which a neutron is kicked out of the target isotope to produce the desired isotope. However, while this method of production is desired for the low energy required, separation is difficult because an isotopic separation is needed, which is much more complex than a chemical process. Therefore, it is interesting to consider granddaughter reactions where the daughter is produced via a gamma-neutron reaction and the granddaughter is a result of decay of the daughter. Since the desired isotope will be a different element from the target isotope, a chemical separation process can be used to extract the desired radioisotope.

A program that considers the isotopes that are of interest and provides a list of possible decay chains that contain the desired radioisotopes is thus desired. This type of approach allows the user to search all possible daughter isotopes in an efficient and timely manner, enabling an exhaustive search for new production pathways.

Focus of the Project

This project began via hand calculations using the online interactive nuclides list as the main source of decay information. However, following decay routes for specific desired isotope granddaughters is a both tedious and not very efficient. Thus, creating a program that would replicate the isotope decay data and then search for the desired set of isotopes within that data was developed to find suitable reaction routes for producing any set of desired isotopes.

METHODS

Initial Process

The program to track the decay paths of the current existing radioisotopes was coded in a python Jupyter Notebook. The data of the current existing isotopes was taken from the Berkeley spreadsheet which is based on the Nuclear Wallet Cards data [1]. This spreadsheet provided the list of isotopes along with their decay modes and branch percentages.

At first, the main functions were established to read the data files and present the information in a user-friendly and understandable manner. The main idea of the program is to create a search function which will go through the list of isotopes and follow their branches of decay. Along the way, the search function was programmed to check if either the decay mode of an isotope led to a stable isotope or if the decay mode led to an isotope of interest. If it led to a stable isotope, the loop would break and go to a new branch, and if it led to an isotope of interest, that information was printed. Initially, the approach involved using a recursive function which would read the decay mode of an isotope,

ISBN: 978-3-95450-214-1 ISSN: 2673-5490 doi:10.18429/JACoW-IPAC2021-TUPAB406

follow the decay chain, and start again until it had gone through all the isotope data. However, this approach ultimately was not sufficient since python has a low recursion limit causing the function to crash.

Thus, a new approach was required for the search function to use loops instead of recursion. The use of *while* loops and *for* loops in python do not hit a limit as quickly as recursion does, so this type of function would be able to run without crashing with the large data set of isotopes.

```
def loop_funct(x):
    element = x[0]
    A = str(int(x[1]) + int(x[2]))
    iso = A+element
    if iso in decay:
        decay_list = decay[iso]
    else:
        return "no decay modes found"
    new_iso = []
    for i in decay_list:
        temp_iso = combo_decay(x,i)
        if temp_iso != -1:
            new_iso.append(temp_iso)
    if len(new_iso) == 0:
        return "no decay modes found"
    iso_stack = [new_iso]
    f = open("decay_chains.txt", "w")


    while len(iso_stack) > 0:
        if len(iso_stack[0]) == 0:
            break
        elif len(iso_stack[-1]) == 0:
            iso_stack.pop()
            iso_stack[-1].pop(0)
        else:
            x = iso_stack[-1][0]
            element = x[0]
            A = str(int(x[1]) + int(x[2]))
            iso = A+element
            if iso in decay:
                decay_list = decay[iso]
            else:
                decay_list = []
            new_iso = []
            for i in decay_list:
                temp_iso = combo_decay(x,i)
                if temp_iso != -1:
                    new_iso.append(temp_iso)
            if len(new_iso) > 0:
                iso_stack.append(new_iso)
            elif len(new_iso) == 0:
                f.write("\ndecay chain:  ")
                for i in iso_stack:
                    f.write(str(i[0]))
                iso_stack[-1].pop(0)
```

Figure 2: Loop function code where input *x* is the isotope.

Change in Direction

The layout of the search function involves a loop function with a *while* loop as shown in Fig. 2, which creates stacks of data for each decay branch that the code follows. Once the branch is complete, the function exits the *while* loop and returns to the top of the function which sets up the next isotope that the search function will track. The decay chain is recorded in a document and rewritten for each isotope so that the overall search function examines the decay tree of only one isotope at a time.

This process continues until all the isotopes in the data set have been read and searched, and then the isotopes which led to an isotope of interest are printed so that the user knows which branches led to the list of desirable isotopes. Therefore, any list of desirable isotopes can be entered by the user so that the search looks for those specific isotopes in the decay chains.

RESULTS

When comparing the results of the decay chains from the code with the calculations done by hand, it is evident that the code is functioning properly. Additionally, because there is no parameter set to produce just one generation of decay, the function produces the full tree of decay which means that the results could be multi-generation granddaughters. Figure 3 provides an example of a medically interesting radioisotope and a possible decay chain that it is produced both from the code as well as what was calculated by hand.

```
In [25]: 1 A.search_funct(["NB", "41", "48"])
Out[25]: {"['ZR', '40', '49']": ['NB', '41', '48']}
```

Medically Needed Isotope (Granddaughter)	Half-Life of Grandaughter	Nearby stable Isotopes	Percent naturally occuring	Natural State	Prossible Photonuclear Reaction Production Mechanism
89Zr	3.27 d	93Nb	100	Solid	93Nb(γ, 4n)89Nb
		92Mo	14.53	Solid	92Mo(γ, p+2n)89Nb
		94Mo	9.15	Solid	
		96Ru	5.54	Solid	

Figure 3: Results from code compared to results by hand.

DISCUSSION

Moving forward on this project will involve finishing the debugging process of the “search all” function in the code since there are still a few entries that remain stuck in the while loop. Additionally, it would be beneficial to polish the decay chain search function code to be more user friendly with easily readable outputs.

Considering next steps in this project, it would be interesting to write a code that takes in the possible daughter isotopes and searches for the most efficient reaction mechanism. This could be expanded beyond just gamma reactions and would provide information such as any other isotopes produced via the chosen reaction route. These two programs would work well in tandem with each other as it would communicate both the most ideal daughter isotopes as granddaughter candidates as well as the best reaction route to produce that daughter.

REFERENCES

[1] Berkeley Nuclear Forensic Search Project, https://metadata.berkeley.edu/nuclear-forensics/Data.html