Plegamiento 2D: Un enfoque informático

Pablo Marín Gómez
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
pabmargom3@alum.us.es (pabmargom3)

David Zamora Fernández
dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Sevilla, España
davzamfer@alum.us.es (davzamfer)

Resumen—

En este trabajo se ha propuesto la implementación de diversas funciones, en el lenguaje de programación Python, orientadas a crear una implementación de un modelo de plegamiento de proteínas básico.

Con estas funciones implementadas, se ha seguido una metodología que permite implementar el algoritmo desde los cálculos más triviales hasta una función las cual aprovechará todos estos cálculos aditivos para obtener finalmente el resultado deseado.

Después de obtener una versión básica del algoritmo, se han realizado pruebas sobre tal implementación y diversas reescrituras para obtener un rendimiento eficiente. Tal cantidad de trabajo sobre el algoritmo ha hecho al equipo llegar a la conclusión de que la generación de plegamientos óptimos es muy costosa y compleja, además de sumamente delicada, ya que un pequeño fragmento de código no optimizado apropiadamente puede aumentar de forma significativa los tiempos de espera.

Junto con la implementación de código, se ha realizado una pequeña investigación acerca del funcionamiento de DeepMind y la historia detrás del problema de predicción de la estructura de proteínas. En esa breve investigación también se ha comentado acerca de trabajos de libre acceso derivados de DeepMind.

Palabras Clave—Plegamiento, Proteínas, Aminoácidos, Optimización, AlphaFold, Algoritmo.

I. Introducción

Este algoritmo ha sido elaborado comenzando desde las funciones más triviales hasta poder obtener todas las herramientas necesarias para poder construir el algoritmo.

Para ello, inicialmente se ha utilizado el conocimiento básico que se había otorgado en el propio documento de entrega, donde se expone de forma muy general el funcionamiento de plegamiento de proteínas y las distintas dificultades que conlleva el cálculo de su aproximación.

Con esa información básica, se pudo realizar las funciones triviales de los primeros ejercicios, pero para poder avanzar más en la entrega, requerimos de tener un conocimiento mayor acerca de cómo funcionan los plegamientos y acerca del método de enfriamiento simulado en sí, por lo que para ello, recurrimos a dos fuentes principales; un documento enlazado en la bibliografía sugerida [1] y una explicación audiovisual[2] de un profesor de la Universidad de Missouri—St. Louis.

Con esas fuentes anteriores, se logró tener un fundamento más amplio del funcionamiento del algoritmo, pero aún no se comprendía bien el funcionamiento del plegamiento en sí, por lo que se tuvo que entrar en contacto con alumnos especializados en la materia, lo cual, junto a una breve explicación más extensa de cómo debería comportarse la función, permitió al equipo finalmente lograr la salida deseada.

Una vez comprendido el funcionamiento del plegamiento, no fue muy complejo lograr una primera versión del algoritmo, el cual, funciona mejor con proteínas pequeñas (una secuencia de 200 aminoácidos es recomendada) y si se decide utilizar alguna proteína mayor, a base de diversos experimentos, se descubrió que los tiempos de esperas podían llegar a ser altos.

Para poder obtener mejores resultados, se decidió experimentar con el funcionamiento del algoritmo, modificando diversas partes de la implementación, hasta que finalmente, se consiguió obtener una versión más eficiente.

Esta última versión ha sido probada con proteínas de distintos tamaños, y a diferencia de su primera iteración, esta ofrece unos tiempos mucho más deseables; siendo el tiempo más extenso que se ha encontrado alrededor de 30 min y con una proteína de más de 1000 aminoácidos.

Este documento ha sido estructurado en distintos apartados para poder explicar de forma más precisa el trabajo que ha sido realizado por el equipo. Este se ha dividido en diversos puntos;

I. **Introducción:** Se explica de forma general el trabajo realizado por el equipo y cómo ha evolucionado.
II. **Preliminares:** Información que el equipo ha recopilado antes y durante la realización del algoritmo, como ayuda para su comprensión.
III. **Metodología:** Exposición del flujo de trabajo del equipo.
IV. **Resultados:** Muestra de las pruebas realizadas por el equipo sobre el algoritmo.
V. **Conclusiones:** Conocimiento obtenido por el equipo tras realizar completamente todo el proyecto.

II. Preliminares

A continuación explicamos cómo hemos llevado a cabo el trabajo y algunos trabajos relacionados con el problema en cuestión, que nos han ayudado a desarrollar la solución:

A. Métodos empleados

Para abordar este problema se ha optado por utilizar algoritmos de optimización, principalmente el algoritmo de Enfriamiento Simulado [2], con el objetivo de poder predecir el plegamiento de proteínas.

A continuación se detalla el funcionamiento básico del algoritmo y como ha sido adaptado en el contexto de la herramienta desarrollada:

1. Inicialmente se parte de una estructura inicial (denominada Cinit), definida como C y de una temperatura inicial máxima (Tmax), la cual vamos a reducir por cada iteración.
2. Por cada iteración, se va a calcular la energía del estado actual y se va a iterar al estado siguiente para calcular también su energía.
3. Una vez obtenidas ambos valores de las energías, se realiza la diferencia entre ambas y se estudia:
   a. Si la diferencia entre la nueva energía y la original es mayor a 0, se redefine el estado nuevo como C.
   b. Si en cambio, no se de lo anterior, pero si se da que $e^{dE/T}$ es mayor que un número aleatorio entre 0 y 1, se definirá también el estado nuevo N, como C. Este condicional servirá como parte de la condición explorativa del algoritmo.

Esta no es la única interpretación del algoritmo existente; por ejemplo, en la figura 2, podemos apreciar una versión del algoritmo ligeramente distinta.

En el contexto desarrollado, ha sido necesario realizar diversos cambios para lograr adaptar el algoritmo. A continuación se detallan esos cambios:

A diferencia del algoritmo original, en este contexto obtenemos todos los nuevos estados posibles obteniendo la longitud total de la estructura, para, a continuación, calcular los pliegues (90º y -90º) desde cada posición.

Una vez calculado los dos pliegues posibles de una posición, se obtienen sus deltas correspondientes y se realizan los mismos pasos que el punto 3.

B. Trabajo Relacionado

Antes de la existencia de DeepFold, para poder conocer la estructura era necesario usar o bien la técnica de Cristalografía de rayos X, Criomicroscopía electrónica o resonancia magnética nuclear, técnicas que permiten conocer la estructura interna de los materiales, pero que son muy costosos tanto económicamente como temporal.

Los resultados de estas técnicas después eran usados (y siguen siendo) usados para verificar el funcionamiento de los algoritmos, los cuales no sobrepasaban el 40% de puntuación media en sus evaluaciones.

De hecho, sus predicciones sólo eran fiables sobre pequeñas proteínas.

Más adelante, llegaría **AlphaFold2**[8], un programa de inteligencia artificial (IA) desarrollado por DeepMind de Alphabets/Google que realiza predicciones de la estructura de las proteínas mediante Deep Learning.

Existen dos versiones mayores de este programa, la versión original (1.0), la cual fue lanzada en 2018, el cual obtuvo el primer puesto en la treceava edición de “Critical Assessment of Techniques for Protein Structure

Prediction", abreviado "CASP", un experimento mundial realizado para evaluar la predicción de la estructura de proteínas, que tuvo lugar en diciembre de 2018.

La puntuación general que obtuvo fue de 68,5, siendo capaz de predecir correctamente las estructuras consideradas "más difíciles" por los organizadores.

A pesar de haber logrado ser un programa tan excelente, el equipo de DeepMind descubrió tiempo después un defecto en AlphaFold; este hacia una sobrevaloración entre algunas interacciones de aminoácidos, lo que provocaba que en sus predicciones aparecieran frecuentemente hélices alfa y hojas beta en sus estructura secundaria.

Debido a este descubrimiento, el equipo de DeepMind decidió refinar el programa mediante el desarrollo de otra versión mayor, AlphaFold 2, la cual introdujo cambios internos considerables para mejorar sus predicciones al mismo tiempo que solventaba las carencias de su antecesora.

Esta versión, al igual que su predecesora, se presentó al experimento CASP de noviembre de 2020, obteniendo una vez más la victoria, pero con una clara mejoría; esta vez obtuvo una puntuación media de 92.4/100.

Además, sus predicciones sirvieron de ayuda para cuatro estructuras.

Fue tal el impacto que realizó AlphaFold 2 en la comunidad científica que se considera de gran importancia en el campo de la biología computacional y la predicción de estructuras a partir de la secuencia de aminoácidos.

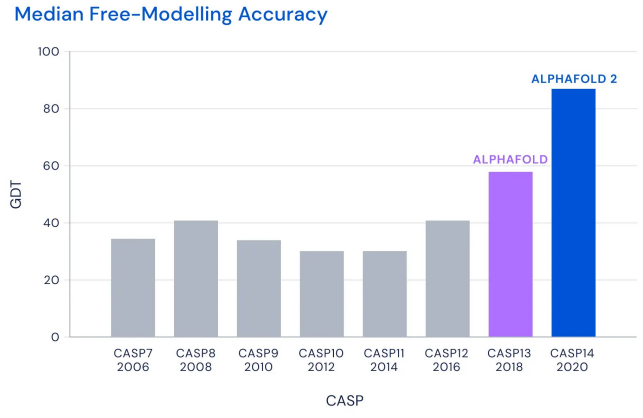


Fig. 1. Impacto de AlphaFold2 en el experimento CASP[10]

También es relevante mencionar que la versión original de AlphaFold inspiró a la creación de otras implementaciones, algunas incluso de código abierto, por ejemplo, **ProSPr**[11], cuyo código es accesible para todos desde su repositorio público de GitHub.

C. *Multidisciplinariedad*

Este complejo problema abarca diferentes campos de la ciencia, por lo que es indudable la necesidad de colaborar entre varias disciplinas para lograr resultados satisfactorios.

Por ejemplo, un ingeniero software es capaz de diseñar algoritmos complejos, pero sin la ayuda de un bioquímico, le resultará bastante complicado llegar a comprender y profundizar en temas como el plegamiento de las proteínas, o simplemente el hecho de averiguar el resultado de sus algoritmos tienen sentido o son realistas.

Por ello, resolver este problema requiere de colaboración entre diversas disciplinas, como pueden ser, especialistas en inteligencia artificial para el desarrollo del algoritmo o bioquímicos que guíen el proyecto y aporten la información y los conocimientos necesarios para llevar a cabo los problemas planteados.

En el caso de DeepMind con AlphaFold, contaban con equipos de ciencias computacionales, ciencias cognitivas, machine learning, human-centred AI neurociencia y ética.

Además cuentan con el apoyo de Google para llevar a cabo la aplicación de sus sistemas en el mundo real.

III. METODOLOGÍA

Se ha seguido una metodología que buscaba maximizar al máximo el trabajo en equipo, esto es, haciendo uso de herramientas que permiten colaborar en tiempo real, tales como:

- Se ha usado Deepnote[3] para poder construir un notebook de Python totalmente compatible con Jupyter Notebook otorgando la ventaja de ser colaborativo en tiempo real.
- Google Docs[4], como plataforma para poder redactar de forma colaborativa esta memoria.
- Discord[5] ha sido usada para la comunicación por voz además del intercambio de archivos.
- Uniprot[6] es la base de datos donde se han extraído algunas secuencias de proteínas para los experimentos.

También se ha abordado la programación del algoritmo desde un enfoque más concreto a uno más amplio del algoritmo en sí; iniciando principalmente en pequeñas funciones que después formarán parte de un

algoritmo más complejo. A continuación se detallan tal funciones:

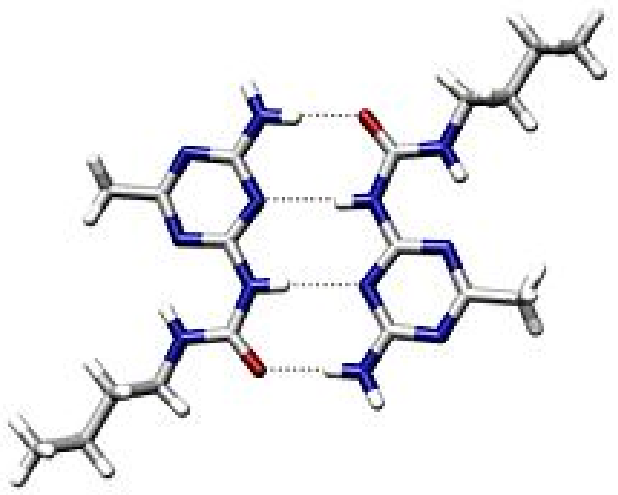


Fig. 2. Ilustración 2D del plegamiento de proteínas

a) *Función "get_spatial_dic"*: Se trata de una función que nos permite obtener el espacio 2D en el que se ubica cada aminoácido.

.

b) *Función "is_hydrophobic"*: Identifica el tipo de aminoácido en función de su variación de energía libre.

c) *Función "get_score"*: Nos proporciona una puntuación a posibles plegamientos de la proteína, de forma que optimizaremos a la más probable, y así, predecir el mismo.

d) *Función "fold":* Esta función obtiene una estructura de una proteína y produce un plegamiento, devolviendo una estructura nueva.

e) *Función "get_successors"*: Introduciendo la información de una proteína, producirá de resultado todos los posibles plegamientos de dicha proteína y sus respectivos espacios 2D. Formaba parte del algoritmo original, pero fue descartado.

Durante el desarrollo del algoritmo y de estas funciones, fue evidente para el equipo que se requeriría de más funciones no especificadas inicialmente, por lo tanto se decidió implementarlas como objetivo de asistir a una o varias funciones en su tarea correspondiente.

A continuación, se detallan tales funciones:

a) Función "get_energy": Esta función es similar a *is_hydrophobic*, diferenciándose únicamente en la salida, ya que en lugar de devolver un valor booleano indicando si es polar o no, devuelve el valor de su energía.

b) Función "structure_to_plot"(Pyplot): herramienta importada para la producción de gráficas 2D con las que se representa el propio plegamiento de la proteína los experimentos del algoritmo.

c) Función "generate_structure": es utilizada para generar una estructura simple a partir de una secuencia de aminoácidos dada. Esta no forma una estructura óptima, sino genera una rápidamente para después ser usada en algoritmos de optimización.

- Sea $s = s_0$
- Para $k = 0$ hasta k_{max} (exclusive):
  - $T \leftarrow$ temperatura(k/k_{max})
  - $s_{nue} \leftarrow$ vecino(s)
  - Si $P(E(s), E(s_{nue}), T) \geq$ azar(0, 1):
    - $s \leftarrow s_{nue}$
- Salida: estado final $\boldsymbol{s}$

La probabilidad de aceptación originalmente propuesta es

- si $e' < e$, entonces $P(e, e', T) = 1$
- si $e' > e$, entonces $P(e, e', T) = \exp(-(e' - e)/T)$

Fig. 3. Algoritmo de Enfriamiento Simulado [7]

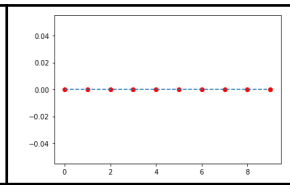
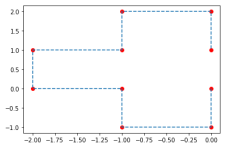
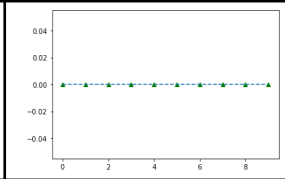
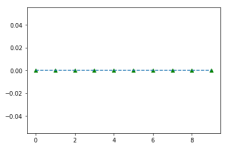
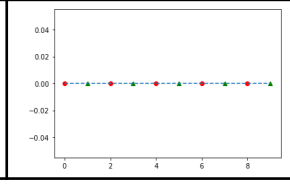
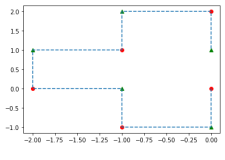
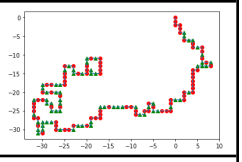
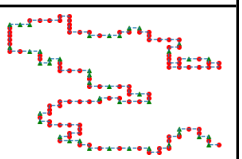
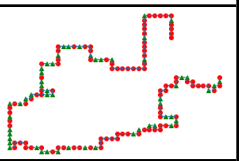
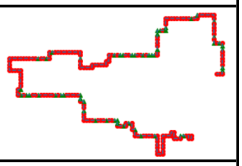
IV. RESULTADOS

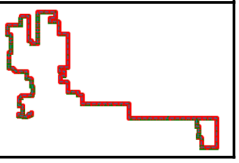
En esta sección se detalla tanto los experimentos realizados como los resultados conseguidos:

Se ha probado el algoritmo con 3 proteínas básicas registradas en UniProt (secuencia menor de 200 aminoácidos). Además, a partir de su secuencia de aminoácidos, se ha creado una función auxiliar que genera una estructura 2D aleatoria.

A continuación se muestra un análisis de los resultados, haciendo comparativas y obteniendo conclusiones a través de una tabla.

Table 1. Comparativa de resultados

Protein	Results		
	Original Plot	*Time*	*Result Plot*
Mismos parámetros, diferente tipo de aminoácidos			
AAAAA		<1s	
KKKKK		<1s	
AKAKA		<1s	
MENOR A 200			
40S ribosomal protein S18 (Ke-3 (Ke3))		<10s	
Cysteine-rich tail protein 1		<10s	
T cell receptor delta constant		<10s	
ENTRE 200 Y 400			
Clarin-2		13s	
ENTRE 400 Y 600			
POTE ankyrin domain family member B3		2,5 min	
ENTRE 600 Y 800			

Protein	Results		
	Original Plot	*Time*	*Result Plot*
Rho GTPase-activating protein 10		6 min	
MAYOR A 800			
UHRF1-binding protein 1-like		33 min	

A. Parámetros de los tests:

Para todos los casos de prueba hemos decidido introducir los siguientes valores para los parámetros del algoritmo:

- temperatura inicial (tInit) = 200

  (Excepto para los tres primeros, en los que hemos cambiado la temperatura inicial a 50 ya que eran más simples y su función original era probar el funcionamiento del algoritmo.)

- enfriamiento (a) = 10

Además, la estructura inicial de la proteína es siempre la misma en cualquier caso:

- IEEEE..E (tantas E como hagan falta para igualar el número de aminoácidos introducidos)

B. Objetivo de los tests:

Los primeros experimentos tratan de casos de prueba aleatorios para probar el algoritmo recién diseñado, de ahí su simpleza de estructura. La principal diferencia entre estos tres casos es la secuencia de aminoácidos, de esta forma comprobamos cómo influye realmente a la hora de plegarse si son hidrofílicos o hidrofóbicos.

A continuación se decidió probar el algoritmo con secuencias reales para comprobar su funcionamiento en casos prácticos. Para ello hemos realizado pruebas escaladas, donde inicialmente probamos el algoritmo con 3 proteínas con menos de 200 aminoácidos que se podían encontrar mediante la búsqueda por filtrado de UniProt.

Una vez que se obtenían resultados satisfactorios, se aumentó el número de aminoácidos para comprobar cómo aumentaba el tiempo en relación con la cantidad de aminoácidos.

Con una proteína de 232 aminoácidos[12] no se obtuvo gran diferencia debido a su cercanía con las

anteriores, obteniendo un tiempo de 14 seg, algo mayor que los casos anteriores pero tampoco provocaba un gran impacto.

Más adelante, al utilizar una proteína de 581 aminoácidos[13], el tiempo de espera, aumentó hasta los 2 minutos y medio, lo cual, era previsible debido a que, a diferencia del caso de prueba anterior, este si se encuentra a una mayor distancia de los casos de prueba originales.

Para la siguiente prueba, se usó una proteína de 786 aminoácidos[14], aumentando a su vez el tiempo de espera hasta los 6 minutos.

Finalmente se realizó una última prueba usando una de las proteínas con más aminoácidos que se pueden encontrar en UniProt, con 1,464 aminoácidos. Para esta prueba se necesitó de alrededor de 30 min para poder obtener un pliegue satisfactorio con los mismos paramentos que los anteriores.

Por último hemos añadido otros tres casos de prueba con secuencias de la base de datos de Uniprot, esta vez con secuencias que no superen a la insulina en la longitud de secuencia, pero que sean algo mayores que las anteriores.

Todo esto se realiza para poder comprobar de forma creciente el equilibrio entre longitud y tiempo necesario del algoritmo implementado.

Los resultados inicialmente no fueron satisfactorios, ya que apenas se apreciaban las diferencias tras el plegamiento en cada uno de los casos de prueba ejecutado, lo que hacía pensar que el algoritmo no estuviera del todo refinado para solucionar el problema de predicción.

Con esa afirmación en mente, el grupo se propuso reescribir el algoritmo, eliminado elementos que se creían causantes de la ineficiencia, hasta que finalmente se llegó a la conclusión de que el uso de la función get_succesors en el algoritmo era sumamente inadecuado, provocado que se calculen estructuras muy poco útiles.

Por ello, se decidió prescindir de esta y reimplantar el algoritmo haciendo uso directamente de la función fold, consiguiendo así, finalmente, unos resultados bastante buenos tanto en tiempo como en el plieuge de la estructura en sí.

V. Conclusiones

Se puede concluir que es extremadamente costoso conseguir una estructura óptima, por lo que hay que tener una precaución extrema a la hora de implantar un algoritmo, ya que como se ha comentado anteriormente, si se elige una mala implementación puede dar lugar a unos resultados muy pobres tal y como pasó al equipo anteriormente.

Esta suma fragilidad a la hora de calcular una estructura es entendible teniendo en cuenta la historia detrás del problema de predicción de los pliegues, a conciencia de que es un problema de más de 50 años de antigüedad (el cual se considera por parte de ciertos sectores resultó ya debido a que superó la puntuación general de 90).

Una vez reestructurado el algoritmo, se realizaron inicialmente las mismas pruebas para corroborar su correcto funcionamiento, y una vez pasadas con éxito, se continuó usando proteínas cada vez más grandes, con la intención de probar el rendimiento al que se puede aspirar.

Con estas pruebas de rendimiento, junto con las estadísticas de UniProt, el equipo considera que el algoritmo es bastante eficiente, dado que la mayoría de las proteínas registradas se encuentran entre 200 y 600 proteínas, siendo el tiempo de espera máximo de 6 minutos en el peor caso es bastante bueno.

En definitiva, haciendo uso tanto de la historia del problema como de los retos que se ha encontrado el equipo en el camino, se puede llegar fácilmente a la comprensión de que obtener un algoritmo óptimo es bastante complejo.

Además, el equipo es plenamente consciente de que el algoritmo implementado a pesar de poder predecir una estructura óptima, esta es lejana de ser la real, ya que a la hora de comprobar el algoritmo no se han realizado comprobaciones sobre cuánto porcentaje de similitud tiene con la estructura apreciada en el mundo real, por lo que no se han realizado ajustes teniendo en cuenta tal métrica, tal y como se realiza en los algoritmos de estas características.

Referencias

[1] Alas-Guardado, Salomón de J, Arturo Rojo, and Gabriel Merino. 2010. “La paradoja de Levinthal: cuando una contradicción se vuelve lógica.” Educación Química 22, no. 1 (Diciembre): 51-54. https://www.elsevier.es/es-revista-educacion-quimica-78-articulo-la-paradoja-levinthal-cuando-una-S0187893X18301149

[2] Adhikari, Badri. n.d. “The simulated annealing algorithm explained with an analogy to a toy.”

YouTube. Accessed June 8, 2022. https://www.youtube.com/watch?v=eBmU1ONJ-os.

[3] Deepnote. n.d. Deepnote - Data science notebook for teams. Accessed June 6, 2022. https://deepnote.com/.

[4] Alphabet, Inc. n.d. Google Docs. Accessed June 6, 2022. https://docs.google.com.

[5] Discord Inc. n.d. Discord | Your Place to Talk and Hang Out. Accessed June 6, 2022. https://discord.com/.

[6] UniProt Consortium. n.d. UniProt. Accessed June 7, 2022. https://www.uniprot.org/

[7] Wikimedia Commons. n.d. "Algoritmo de recocido simulado." Wikipedia. Accessed June 7, 2022. https://es.wikipedia.org/wiki/Algoritmo_de_recocido_simulado.

[8] "AlphaFold." n.d. DeepMind. Accessed June 8, 2022. https://www.deepmind.com/research/highlighted-research/alphafold.

[9] Wikimedia Commons. n.d. "AlphaFold." Wikipedia. Accessed June 10, 2022. https://es.wikipedia.org/wiki/AlphaFold.

[10] DeepMind. 2020. "AlphaFold: a solution to a 50-year-old grand challenge in biology." DeepMind. https://www.deepmind.com/blog/alphafold-a-solution-to-a-50-year-old-grand-challenge-in-biology.

[11] dellacortelab. n.d. "ProSPr: Protein Structure Prediction." GitHub. Accessed June 10, 2022. https://github.com/dellacortelab/prospr.

[12] "Clarin-2." 2020. UniProt. https://beta.uniprot.org/uniprotkb/A0PK11/entry#sequences.

[13] "POTE ankyrin domain family member B3." 2020. UniProt. https://beta.uniprot.org/uniprotkb/A0JP26/entry#sequences.

[14] "Rho GTPase-activating protein 10." 2020. UniProt. https://beta.uniprot.org/uniprotkb/A1A4S6/entry#sequences.