

# PROTAgENTS: PROTEIN DISCOVERY VIA LARGE LANGUAGE MODEL MULTI-AGENT COLLABORATIONS COMBINING PHYSICS AND MACHINE LEARNING

**Alireza Ghafarollahi**

Laboratory for Atomistic and Molecular  
Mechanics (LAMM)  
Massachusetts Institute of Technology  
77 Massachusetts Ave.  
Cambridge, MA 02139, USA

**Markus J. Buehler**

Laboratory for Atomistic and Molecular  
Mechanics (LAMM)  
Center for Computational Science and Engineering  
Schwarzman College of Computing,  
Massachusetts Institute of Technology  
77 Massachusetts Ave.  
Cambridge, MA 02139, USA

Correspondence: mbuehler@MIT.EDU

## ABSTRACT

Designing *de novo* proteins beyond those found in nature holds significant promise for advancements in both scientific and engineering applications. Current methodologies for protein design often rely on AI-based models, such as surrogate models that address end-to-end problems by linking protein structure to material properties or vice versa. However, these models frequently focus on specific material objectives or structural properties, limiting their flexibility when incorporating out-of-domain knowledge into the design process or comprehensive data analysis is required. In this study, we introduce ProtAgents, a platform for *de novo* protein design based on Large Language Models (LLMs), where multiple AI agents with distinct capabilities collaboratively address complex tasks within a dynamic environment. The dynamic collaboration between agents, empowered by LLMs, provides a versatile approach to tackling protein design and analysis problems, as demonstrated through diverse examples in this study. The problems of interest encompass designing new proteins, analyzing protein structures and obtaining new first-principles data – natural vibrational frequencies – via physics simulations. The concerted effort of the system allows for powerful automated and synergistic design of *de novo* proteins with targeted mechanical properties. The flexibility in designing the agents, on one hand, and their capacity in autonomous collaboration through the dynamic LLM-based multi-agent environment on the other hand, unleashes great potentials of LLMs in addressing multi-objective materials problems and opens up new avenues for autonomous materials discovery and design.

## 1 INTRODUCTION

Proteins, the building blocks of life, serve as the fundamental elements of many biological materials emerging from natural evolution over the span of 300 million years. The underlying sequences of amino acids (AAs) in a protein determines its unique three-dimensional structure, which, in turn, dictates its specific biological activity and associated outstanding properties. The diversity in protein design is immense, with over  $20^{100}$  possible AA sequences for just a relatively small 100-residue protein. However, the natural evolutionary process has sampled only a fraction of this vast sequence space. This leaves a substantial portion uncharted, presenting a significant opportunity for the *de novo* design of proteins with potentially remarkable properties Huang et al. (2016). Despite this potential, the extensive design space, coupled with the costs

associated with experimental testing, poses formidable challenges in *de novo* protein design. Navigating this intricate landscape necessitates the development of a diverse set of effective tools enabling the targeted design of *de novo* proteins with specific structural features or properties.

LLMs (Wei et al., 2022; Chang et al., 2023) have represented a paradigm shift in modeling problems across a spectrum of scientific and engineering domains (Jablonka et al., 2023; Buehler, 2023a;b; Boyko et al., 2023; Madani et al., 2020; Ni & Buehler, 2023; Brodnik et al., 2023; Tinn et al., 2023; Hu & Buehler, 2023; Buehler, 2023d; Luu & Buehler, 2023; Peng et al., 2021; Brown et al., 2020; Bates, 1995; Thoppilan et al., 2022; Chowdhery et al., 2023). Such models, built upon attention mechanism and transformer architectures Vaswani et al. (2017), have emerged as powerful tools recently in the field of materials science and related areas, contributing to various aspects ranging from knowledge retrieval to modeling, design, and analysis (Buehler, 2023a;d;b; Lu et al., 2023).

However, LLMs are typically not best equipped to solve specific physics-based forward and inverse design tasks, and are often focused on leveraging their conversational capabilities. Recently, LLMs have been instrumental in powering conversable AI agents through a multi-agent strategy, facilitating the transition from AI-human conversations to AI-AI or AI-tools interactions for increased autonomy (Ni & Buehler, 2023; Buehler, 2023d; Zhang et al., 2023; Wang et al., 2023a; Xi et al., 2023).

In this paper, we propose a multi-agent strategy to the protein design problems by introducing ProtAgents, a multi-agent modeling framework to solve protein-related analysis and design problems by leveraging customized functions across domains and disciplines. The core underpinning concept of the multi-agent systems is the use state-of-the-art LLMs-based agents combined with a series of tools. The LLM backbone exhibits exceptional analytical, rational, and strategic planning capabilities, which are crucial for solving complex problems. By leveraging these abilities and utilizing various tools, the proposed model seeks to minimize the necessity for human intervention and intelligence at different stages of protein design.

**Contributions:** the main contribution of our work is summarized as follows

- We propose ProtAgents, a pioneering multi-agent modeling framework that combines state-of-the-art LLMs with diverse tools to tackle protein design and analysis problems
- Our model harnesses the collective capabilities of agents with specialized expertise that interact autonomously and nonlinearly to solve the protein-related task.
- Equipped with various tools and functions, the model demonstrates an advanced ability to integrate new physical data from different disciplines, surpassing conventional deep learning models in versatility and problem-solving capacity in protein science.
- Our model significantly minimizes the need for human input throughout different stages of the problem-solving process.

## 2 RELATED WORK

**machine learning for protein design** In the field of *de novo* protein design, data-driven and machine learning methods have emerged as powerful tools, offering valuable insights and accelerating the discovery of novel proteins with desired properties (Notin et al., 2022; Ingraham et al., 2019; Wu et al., 2022; Anand & Achim, 2022; Eguchi et al., 2022; Rives et al., 2021; Madani et al., 2020; Anand et al., 2022; Greener et al., 2018; Riesselman et al., 2018; Alley et al., 2019; Watson et al., 2023; Anishchenko et al., 2021; Ingraham et al., 2023). These methods have opened great avenues for predicting structure, properties, and functions of proteins solely based on their underlying AA sequence (Yu et al., 2022; Elnaggar et al., 2022; Mirabello & Pollastri, 2013; Pollastri et al., 2002; Zhang et al., 2018; Pollastri & McLysaght, 2005). As such, the development of such DL models significantly reduces the cost of screening the vast sequence space to target proteins with improved or optimized mechanical performance.

A frontier, however, that still exists is how we can automate the design process of *de novo* proteins by creating intelligent systems that leverage a diverse set of knowledge and abilities, combining data-driven ML tools with physics-based modeling, important for accurate predictions.

**LLM-based protein design** LLMs have demonstrated significant potential in mimicking human intelligence, generating text across a broad spectrum of topics and sentiments. Increasingly, this concept is being applied to the domain of protein sequences, which, akin to human languages, can be represented as strings of letters corresponding to amino acids. Recent efforts have pivoted towards developing LLMs specifically for *de novo* protein design. These models, trained on extensive datasets of natural protein sequences, are capable of predicting *de novo* sequences with varied functionalities Unsal et al. (2022); Ferruz & Höcker (2022); Ferruz et al. (2022); Nijkamp et al. (2023); Madani et al. (2023); Ruffolo & Madani (2024).

**LLM-based multi-agent strategy** Concurrently, LLM-based agents have garnered substantial interest in various disciplines Wang et al. (2023b); Guo et al. (2024). These agents enhance the capabilities of LLMs by integrating tools from across domains and disciplines including pre-trained ML and DL models. The concept of a multi-agent strategy has been investigated in the fields of materials science and mechanical engineering in previous studies Buehler (2023b), and has been further explored within the realm of molecular modeling Buehler (2023d). The use of AI scientists in a multi-agent strategy to enhance biomedical research and scientific discovery is comprehensively addressed in Gao et al. (2024), highlighting its potential to revolutionize the field.

### 3 METHOD

Our multi-agent framework consists of a team of agents, each powered by a state-of-the-art general-purpose LLM, GPT-4, and characterized by a unique profile that details its role as shown in Figure 1a. Furthermore, agents are given access to a set of tools with various functionalities across domains. As shown in Figure 1b each function is characterized by a descriptive profile and input parameters. A key part of developing our multi-agent system is selecting the appropriate types and numbers of agents and carefully defining their profiles and the functions they perform. We have optimized these agents through a process of trial and error, conducting experiments to ensure they deliver optimal performance in critical areas of the problem-solving process. These areas include planning, seamless collaboration between agents, function execution, error management, and informed decision-making based on the given task and past/current observations.

The outline of the proposed multi-agent framework is shown in Figure 1c, illustrating the collaborative efforts of a team of four agents organized into a team structure, overseen by a manager who coordinates overall communication among the agents. A generic structure illustrating how the team of agents collaborate in the current study is depicted in 2. Moreover, **Table B1** in Appendix B lists the full profile for the agents recruited in our multi-agent framework. Further details can be found in A.

The tools are incorporated into the model via the "Assistant" agent who oversees executing the functions and returning the results. ProtAgents consists of a rich library of functions each with special powers in solving the protein problems. Each function has a distinct profile describing its focus and takes

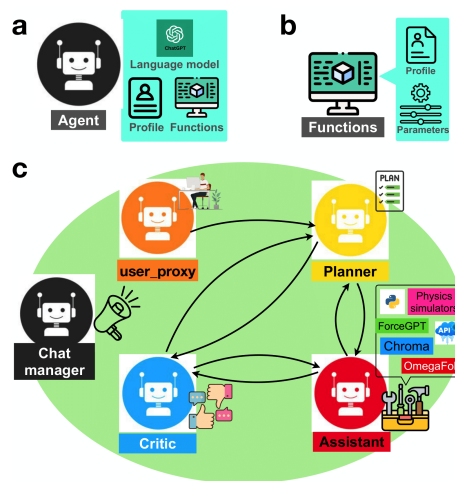


Figure 1: Automated multi-agent AI framework for protein discovery and analysis, featuring: **a**, agents with language communication, defined focuses, and custom functions; **b**, functions tailored by profiles and parameters; **c**, specialized agent teams for problem description (*user\_proxy*), planning (*Planner*), execution (*Assistant*), and evaluation (*Critic*), coordinating through a dynamic group chat (*Chat manager*) for efficient, human-independent problem-solving in protein design.”

one or more required entities as the input. The functions provide the ability to, for instance, retrieve knowledge, perform protein folding, analyze the secondary structure, and predict some mechanical properties through a pre-trained autoregressive language model (see A). Additionally, a function can carry out simulations to compute the protein natural frequencies, thus allowing the model to integrate the new physics-based data. A full list of functions implemented in the current study is provided in **Table B2** in Appendix B.

## 4 EXPERIMENTS

We present a series of experiments aimed at evaluating the effectiveness and potential of a multi-agent modeling framework for various challenges within the domain of protein design and analysis.

### 4.1 *De novo* PROTEIN DESIGN USING CHROMA

In this example, we assess the model’s performance in using the state-of-the-art models in *de novo* protein design, ChromaIngraham et al. (2023), and protein folding, OmegaFoldWu et al. (2022). To this end, we formulate a complex multi-step task with the objective of comparing the two generative models based on various structural and physical features derived from the folded structures obtained through Chroma and OmegaFold2. We pose the following task through the “*user\_proxy*” agent which is subsequently followed by a plan developed by the “*planner*”:

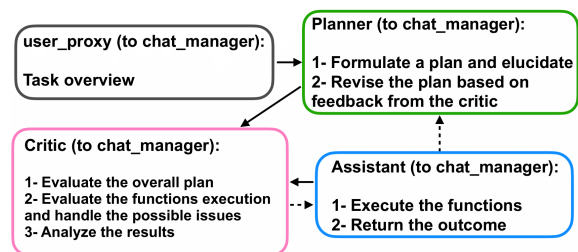
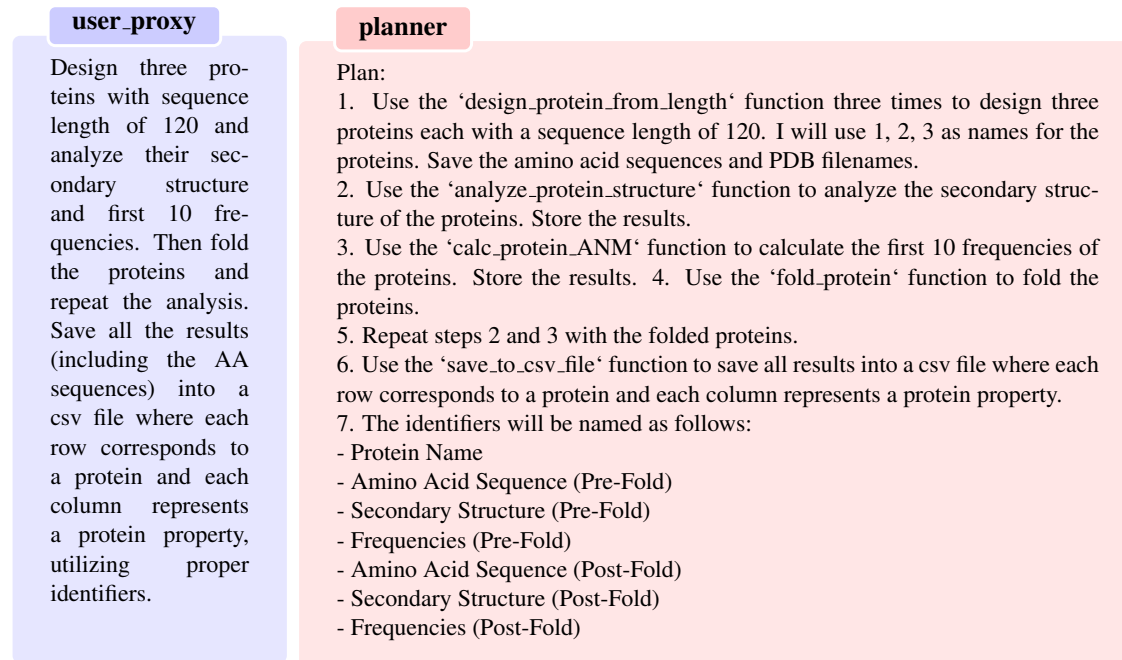


Figure 2: A flowchart outlining the dynamic interaction between the multi-agent team members to solve protein design and analysis problems.



At first glance, the plan seems to cover all the details necessary to accomplish the tasks included in the problem statement. However, the “*critic*” agent who is responsible in giving feedback about the plan spots a minuscule error in the saving part of the plan as follows:

**critic**

Your plan is almost perfect. However, when a protein is folded, the protein sequence doesn’t change, only the structure does, which then affects the secondary structure and possible frequencies you can calculate. So you need to only save Amino Acid Sequence once after designing the proteins. Each protein should have the following column properties:

- Protein Number
- Amino Acid Sequence
- Secondary Structure (Pre-Fold)
- Frequencies (Pre-Fold)
- Secondary Structure (Post-Fold)
- Frequencies (Post-Fold)

Other than that, your plan works perfectly. So steps will be as follows:

...

The correction made by the “*critic*” concerning the sequence length underscores its notable proficiency in comprehending how diverse functions and parameters influence various aspects within the realm of protein design.

The “*user-proxy*” agent is then asked to confirm the plan. The “*assistant*” then takes the stage and starts following the plan by calling and executing the functions until all the steps have been undertaken. An overview of the work performed by the “*assistant*” is depicted in **Figure B2** in Appendix B. At the end of the computations, the results are formatted into a JSON dictionary to be fed into the “*save\_to\_csv\_file*” function. However, an error related to the JSON dictionary format occurs when executing the function as follows:

```
***** Response from calling function "save_to_csv_file" *****
Error: Expecting property name enclosed in double quotes: line 2 column 5 (char 6)
*****
```

The “*critic*” then steps in by making the following comment and suggesting a plan to fix the error as follows:

**critic**

I apologize for the mistake. It seems there was an issue with the formatting of the input JSON dictionary for saving to the CSV file. Let’s try again with properly formatted JSON.

*Plan to fix the error:*

1. Correctly format JSON with double quotes instead of single quotations.
2. Call ‘*save\_to\_csv\_file*’ function again with correctly formatted JSON dictionary.

I’ll directly proceed with this plan as the error was due to syntax not plan.

The critic makes the necessary corrections and suggests the corrected JSON dictionary for the “*assistant*” to execute the “*save\_to\_csv\_file*” function. This time, the function is successfully executed and the results are saved into a csv file as shown in **Table B3** in Appendix B.

This experiment showcases a good example of multi-agent collaboration in solving complex tasks in the context of *de novo* protein design and analysis. It specially shows the great capability of the “*critic*” agent

in providing valuable feedback to other working agents at different stages of the problem solving endeavor, further assisting the team of agents in handling possible errors without the need for human involvement. Figure 3 shows the plots of the generated results including the 3D folded structures. The full conversations can be found in **Table B4** in Appendix B.

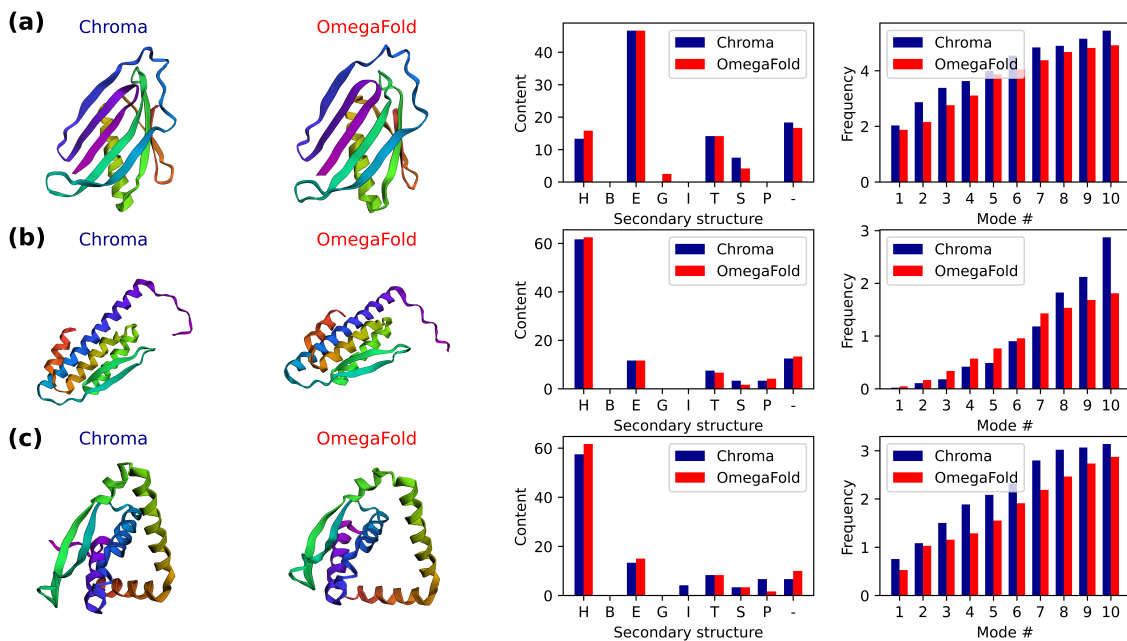


Figure 3: **The results generated by the multi-agent collaboration for the first experiment, Section 4.1.** The first and second columns depict the 3D folded structures of proteins generated by Chroma and OmegaFold2, respectively, while the third and fourth columns represent the fractional content of secondary structures, and first ten natural frequencies for the generated proteins.

#### 4.2 PROTEIN DESIGN CONDITIONED ON THE PROTEIN CATH CLASS

CATH is a hierarchical classification system for protein structures that consists of four main levels. The highest level in this hierarchy is the “Class” which primarily characterizes the secondary structure content of the protein. For example,  $C_1$ ,  $C_2$ , and  $C_3$  correspond to proteins predominantly composed of  $\alpha$ -helix, mainly  $\beta$ -sheet, and a combination of  $\alpha$  and  $\beta$  secondary structures. Consequently, designing proteins based on the CATH class number, i.e.  $C_1$ ,  $C_2$ ,  $C_3$ , can be understood as creating proteins with a specific fractional content of the secondary structure. Previous studies have demonstrated the importance of the protein secondary structures content, specially  $\alpha$ -helix/ $\beta$ -sheet ratio, on the mechanical properties of the protein materials (Sikora et al., 2009; Nakajo et al., 2007; Ackbarow et al., 2007; Knowles & Buehler, 2011; Xu & Buehler, 2010; Qin & Buehler, 2010). Chroma Ingraham et al. (2023) has the potential to conditionally generate proteins with specified folds according to CATH class annotations at three levels

In this example, we task the multi-agent team with generating proteins based on their CATH class and subsequently performing computational and structural analysis tasks. Specifically, in addition to secondary structure analysis and natural frequency calculations, as covered in previous examples, we instruct the team to compute the maximum unfolding force (maximum force in the unfolding force-separation curve) and unfolding energy (the area under the unfolding force-separation curve) for each generated protein. To accomplish the latter, we have equipped the multi-agent team with a custom function that utilizes a trained

autoregressive transformer generative AI model, *ForceGPT*. Detailed information about the training of the model can be found in **Materials and Methods** Appendix A. The task given is:

**user\_proxy**

Design two proteins of length 100 for each family of mainly beta, mainly alpha, and alpha beta proteins. Then compute the maximum force and unfolding energy, analyze the secondary structure and compute the first 10 frequencies for these proteins.

The planner outlines a detailed plan comprising all the required sub-tasks to be accomplished step by step. Moreover, the planner has identifies and suggests the correct functions and corresponding input parameters for each sub-task. The plan is further supported by the “*critic*” who provides positive feedback as follows:

**planner**

Yes, this plan sounds good. It covers each step required for the task. The parameter inputs for each function match with the required parameters as stated in their function definitions. Thus, this plan should successfully accomplish the task.

The multi-agent team then proceeds to execute the different steps outlined in the plan by calling and executing the functions. Specifically, the function ‘design\_protein\_from\_CATH’ is executed with the appropriate ‘CATH\_ANNOTATION’ for a specific protein structure design, as outlined in the plan. Following the generation of all proteins, the executions are followed by structural analysis and force and energy computations. It’s noteworthy that the model exhibits good performance in restoring and memorizing the sequences of the generated proteins, which are essential for the force and energy calculations. Finally, the team successfully completes the task by computing the first 10 frequencies for each protein. An overview of the computations performed by the team of agents for this experiment is shown in **Figure S3** of the supplementary material.

Given the complexity of the problem involving numerous computational tasks, a decent number of results have been generated in the first round of the conversation. In the next round, to evaluate the team’s ability to memorize and restore the results, we present the following task:

**user\_proxy**

Could you save the results in a CSV file named “protein\_analysis.csv,” where each row corresponds to a protein, and each column represents a specific property? Include the AA sequence in the results and use suitable identifiers for the columns.

In this task, we not only request the team to save the data but also require them to adhere to a customized format when storing the results. The model is proficient in creating a JSON dictionary that satisfies the specified format and saving the results to a CSV file, as illustrated in **Table B5** in Appendix B.

The plots of the obtained results are shown in Figure 4. The results indicate that Chroma has done a poor performance in creating  $\beta$ -rich protein named *mainly\_beta.protein\_2* which its structure is dominant in  $\alpha$ -helix. As an attempt to test the capability of the multi-agent model in analyzing the results, in the last round of the conversation, we ask the model to assess Chroma’s performance in generating the proteins conditioned on the secondary structure by posing the following question:

**user\_proxy**

Based on the results of this example, can you check if the protein generator (Chroma) has been successful in creating proteins with desired structure?

The “*critic*” agent conducts a thorough evaluation of Chroma’s performance in generating proteins with targeted secondary structure content. Through a detailed analysis of each CATH structure, it reveals the inher-

ent strengths and weaknesses in Chroma’s capabilities. Specifically, addressing the limitations of Chroma’s performance, the critic’s evaluation provides the following observations for the mainly beta proteins:

**critic**

- The mainly beta proteins showed higher percentages of extended strand/beta-sheet secondary structure ('E'). Though, the percentages varied quite a bit (64% for mainly\_beta\_protein\_1 and only 8% for mainly\_beta\_protein\_2), which could be due to the complex nature of beta-structures.

This illustration not only highlights the multi-agent model’s proficiency in computational tasks but also underscores its intelligence in handling intricate data analyses—an aspect traditionally reserved for human. The full conversations for this experiment can be found in **Table B6** in Appendix B.

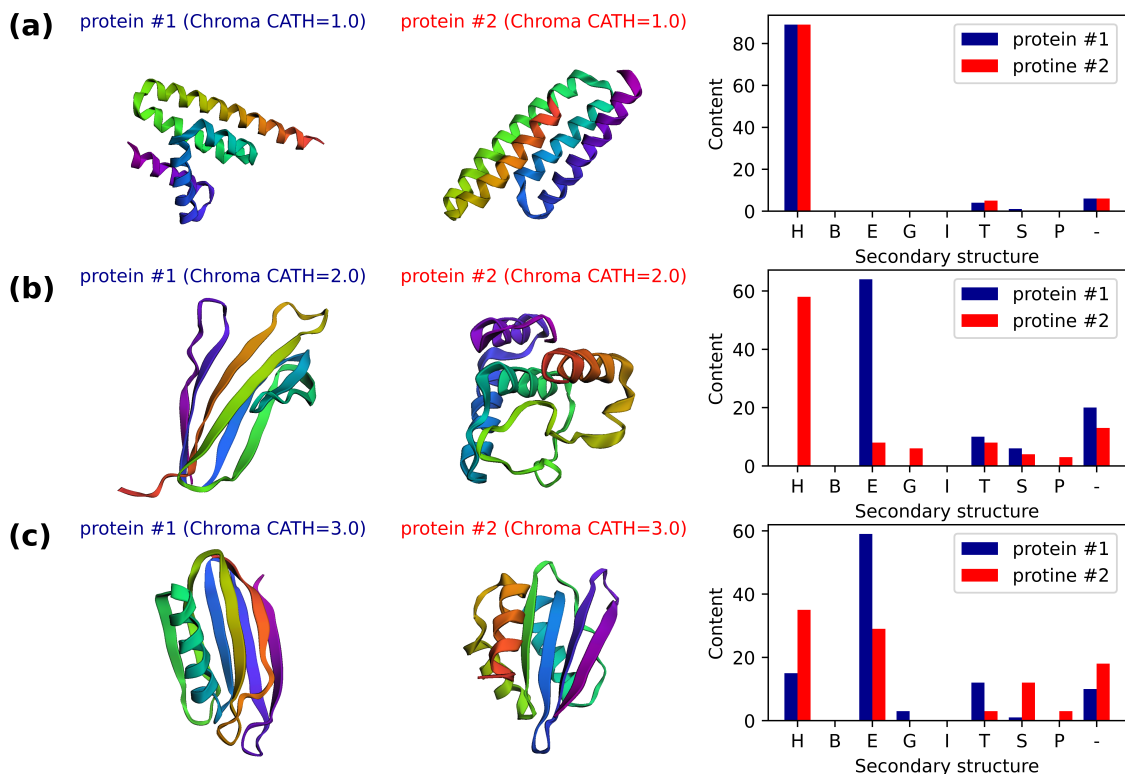


Figure 4: **The results generated by the multi-agent collaboration for the second experiment, Section 4.2.** The first and second columns depict the 3d folded structures and the last column represents the fractional content of secondary structures for the two proteins generated by Chroma conditioned on the CATH class of (a) 1: mainly alpha protein, (b) 2: mainly beta protein, and (c) 3: alpha beta protein.

## 5 CONCLUSIONS

Large Language Models (LLMs) have made remarkable strides, revealing their immense potential to potentially replicate human-like intelligence across diverse domains and modalities, demonstrating proficiency in comprehending extensive collective knowledge and proving adept at effectively applying this information. However, to reach intelligent problem-solving systems, these types of models are not yet sufficient and require integration with other methods and the LLM-intelligence driven multi-agent systems have emerged as a

powerful approach to fill this gap. These AI systems are composed of a team of agents with distinct expertise that have been powered by a general purpose LLM model, such as GPT-4, which allows them to communicate via conversation. More importantly, such systems are not limited to mere linguistic interactions between agents; they have the capacity to incorporate a variety of special-purpose modeling and simulation tools, human input, tools for knowledge retrieval, and even deep learning-based surrogate models to solve particular tasks. Furthermore, additional tools can be integrated into the multi-agent system with popular external APIs, and up-to-date knowledge about special topics can be retrieved by searching and browsing the web through specialized API interfaces. By harnessing the collective abilities of agents, including reasoning, tool usage, criticism, mutual correction, adaptation to new observations, and communication this framework has proven highly effective in navigating intricate challenges including protein design.

In this study, we have explored the protein design problem through the lens of a multi-agent strategy by developing ProtAgents. Our aim was to create an autonomous system composed of expert agents capable of designing proteins *de novo*, and performing tasks like secondary structure analysis, performing physical simulations, and predicting the protein’s unfolding energy and maximum force. To achieve this goal, we assembled a group of agents, each assigned a unique profile through initial prompts, to dynamically interact in a group chat via conversations and make decisions and take actions based on their observations. The agents profile outlines their attributes, roles, and functionalities within the system and describes communication protocols to exchange information with other agents in the system. Our team of agents include a *user\_proxy* to pose the query, a *planner* to formulate a plan, an function-baked *assistant* to execute the functions and return the results, and a *critic* to evaluate the outcome and criticize the performance. Through a series of experiments, we unleashed the power of agents in not only conducting the roles they were assigned to, but to autonomously collaborate by discussion powered by the all-purpose LLM, GPT-4. Our experiments have showcased the great potential of the multi-agent modeling framework in tackling complex tasks as well as integrating AI-agents into physics-based modeling.

Our work represents a foundational step towards the ambitious goal of developing an intelligent system that could function as a "protein scientist," capable of making significant discoveries in the field of protein science. We envision that such models could play a crucial role in generating scientific hypotheses in various domains related to *de novo* protein design. These models could also outline a series of tasks to test these hypotheses, refining them based on the results obtained. The model we have developed could provide the necessary framework for planning and conducting the validation phase, by initiating and executing appropriate functions. For example, it could evaluate the secondary structure of proteins or conduct molecular dynamics simulations.

Venturing into the development of systems that push the boundaries of current scientific capabilities uncovers several challenges and shortcomings that must be meticulously addressed before these systems can be accurately and safely deployed. Language models (LMs), serving as the core of the multi-agent system, are susceptible to generating unreliable predictions, including the fabrication of non-factual information and the presence of systematic biases. Consequently, specialized evaluation mechanisms are necessary to accurately assess the performance of these models. Additionally, as more agents are integrated into the system, the complexity of agent interactions is likely to increase, potentially complicating the planning and function-execution stages and thus impacting overall model performance. The implementation of precise communication protocols may be necessary to overcome this hurdle. Another challenge relates to the iterative nature of the hypothesis generation-validation-refinement process, essential for discovery in protein science, which might provoke extended discussions due to its iterative nature. Enabling agents to recall prior steps in such complex, prolonged processes and to integrate comprehensive observational data into their decision-making introduces an additional layer of complexity to the model’s performance. In summary, while the potential of intelligent systems to revolutionize protein science is immense, the path to their realization is fraught with significant technical and operational challenges, and addressing these challenges head-on is crucial for the successful deployment and application of such systems in scientific discovery.

#### AUTHOR CONTRIBUTIONS

MJB and AG conceived the study and developed the multi-agent models. AG performed the tests for various problems, analyzed the results and prepared the first draft of the paper. MJB supported the analysis, revised and finalized the paper with AG.

#### ACKNOWLEDGEMENTS

We acknowledge support from USDA (2021-69012-35978), DOE-SERDP (WP22-S1-3475), ARO (79058LSCSB, W911NF-22-2-0213 and W911NF2120130) as well as the MIT-IBM Watson AI Lab, MIT’s Generative AI Initiative, and Google. Additional support from NIH (U01EB014976 and R01AR077793) ONR (N00014-19-1-2375 and N00014-20-1-2189) is acknowledged. AG gratefully acknowledges the financial support from the Swiss National Science Foundation (#P500PT\_214448).

#### REFERENCES

- Openai api. URL <https://openai.com/blog/openai-api>.
- LlamaIndex (formerly GPT index), a data framework for LLM applications. URL [https://github.com/run-llama/llama\\_index](https://github.com/run-llama/llama_index).
- Theodor Ackbarow, Xuefeng Chen, Sinan Keten, and Markus J. Buehler. Hierarchies, multiple energy barriers, and robustness govern the fracture mechanics of  $\alpha$ -helical and  $\beta$ -sheet protein domains. *Proceedings of the National Academy of Sciences of the United States of America*, 104:16410–16415, 10 2007. ISSN 00278424. doi: 10.1073/PNAS.0705759104/SUPPL\_FILE/0705759FIG6.JPG. URL <https://www.pnas.org/doi/abs/10.1073/pnas.0705759104>.
- Ethan C. Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M. Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature Methods* 2019 16:12, 16:1315–1322, 10 2019. ISSN 1548-7105. doi: 10.1038/s41592-019-0598-1. URL <https://www.nature.com/articles/s41592-019-0598-1>.
- Namrata Anand and Tudor Achim. Protein structure and sequence generation with equivariant denoising diffusion probabilistic models. <https://arxiv.org/abs/2205.15019v1>, 5 2022. URL <https://arxiv.org/abs/2205.15019v1>.
- Namrata Anand, Raphael Eguchi, Irimpan I. Mathews, Carla P. Perez, Alexander Derry, Russ B. Altman, and Po Ssu Huang. Protein sequence design with a learned potential. *Nature Communications* 2022 13:1, 13: 1–11, 2 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-28313-9. URL <https://www.nature.com/articles/s41467-022-28313-9>.
- OpenAI and: Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florenzia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter

- Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. 3 2023. URL <https://arxiv.org/abs/2303.08774v4>.
- Ivan Anishchenko, Samuel J. Pellock, Tamuka M. Chidyausiku, Theresa A. Ramelot, Sergey Ovchinnikov, Jingzhou Hao, Khushboo Bafna, Christoffer Norn, Alex Kang, Asim K. Bera, Frank DiMaio, Lauren Carter, Cameron M. Chow, Gaetano T. Montelione, and David Baker. De novo protein design by deep network hallucination. *Nature* 2021 600:7889, 600:547–552, 12 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-04184-w. URL <https://www.nature.com/articles/s41586-021-04184-w>.
- A R Atilgan, S R Durell, R L Jernigan, M C Demirel, O Keskin, and I Bahar. Anisotropy of fluctuation dynamics of proteins with an elastic network model. *Biophys. J.*, 2001. doi: 10.1016/S0006-3495(01)76033-X.
- Ahmet Bakan, Lidio M. Meireles, and Ivett Bahar. Prody: Protein dynamics inferred from theory and experiments. *Bioinformatics*, 27:1575–1577, 6 2011. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTR168. URL <https://dx.doi.org/10.1093/bioinformatics/btr168>.
- Madeleine Bates. Models of natural language understanding. *Proceedings of the National Academy of Sciences*, 92:9977–9982, 10 1995. ISSN 00278424. doi: 10.1073/PNAS.92.22.9977. URL <https://www.pnas.org/doi/abs/10.1073/pnas.92.22.9977>.
- James Boyko, Joseph Cohen, Nathan Fox, Maria Han Veiga, Jennifer I-Hsiu Li, Jing Liu, Bernardo Modenesi, Andreas H. Rauch, Kenneth N. Reid, Soumi Tribedi, Anastasia Visheratina, and Xin Xie. An interdisciplinary outlook on large language models for scientific research. 11 2023. URL <https://arxiv.org/abs/2311.04929v1>.
- Neal R. Brodник, Samuel Carton, Caelin Muir, Satanu Ghosh, Doug Downey, McLean P. Echlin, Tresa M. Pollock, and Samantha Daly. Perspective: Large language models in applied mechanics. *Journal of*

- Applied Mechanics*, 90, 10 2023. ISSN 0021-8936. doi: 10.1115/1.4062773. URL <https://dx.doi.org/10.1115/1.4062773>.
- Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020. URL <https://commoncrawl.org/the-data/>.
- Markus J. Buehler. Melm, a generative pretrained language modeling framework that solves forward and inverse mechanics problems. *Journal of the Mechanics and Physics of Solids*, 181:105454, 12 2023a. ISSN 0022-5096. doi: 10.1016/J.JMPS.2023.105454.
- Markus J. Buehler. Mechgpt, a language-based strategy for mechanics and materials modeling that connects knowledge across scales, disciplines and modalities. *Applied Mechanics Reviews*, pp. 1–82, 10 2023b. ISSN 0003-6900. doi: 10.1115/1.4063843. URL <https://dx.doi.org/10.1115/1.4063843>.
- Markus J. Buehler. Generative pretrained autoregressive transformer graph neural network applied to the analysis and discovery of novel proteins. *Journal of Applied Physics*, 134, 5 2023c. doi: 10.1063/5.0157367. URL <https://arxiv.org/abs/2305.04934v2>.
- Markus J. Buehler. Generative retrieval-augmented ontologic graph and multi-agent strategies for interpretive large language model-based materials design. *ACS Engineering AU*, 10 2023d. URL <https://doi.org/10.1021/acseengineeringau.3c00058>.
- Yupeng Chang, X U Wang, Yuan Wu, Hao Chen, Wei Ye, Yue Zhang, Y I Chang, Philip S Yu, Xiaoyuan Yi, Yi Chang, Qiang Yang, Hong Kong, Xu Wang, Jindong Wang, Linyi Yang, Kaijie Zhu, Cunxiang Wang, Yidong Wang, and Xing Xie. A survey on evaluation of large language models. *J. ACM*, 37:42, 7 2023. URL <https://arxiv.org/abs/2307.03109v8>.
- Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24:1–113, 2023. ISSN 1533-7928. URL <http://jmlr.org/papers/v24/22-1144.html>.
- Peter J.A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J.L. De Hoon. Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25:1422, 6 2009. ISSN 13674803. doi: 10.1093/BIOINFORMATICS/BTP163. URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2682512/>.  
URL <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2682512/?report=abstract>
- Pemra Doruker, Ali Rana Atilgan, and Ivet Bahar. Dynamics of proteins predicted by molecular dynamics simulations and analytical approaches: Application to-amylase inhibitor. *PROTEINS: Structure, Function, and Bioinformatics*, 2000. doi: 10.1002/1097-0134. URL <https://onlinelibrary.wiley.com/terms-and-conditions>.



- Raphael R. Eguchi, Christian A. Choe, and Po Ssu Huang. Ig-vae: Generative modeling of protein structure by direct 3d coordinate generation. *PLOS Computational Biology*, 18:e1010271, 6 2022. ISSN 1553-7358. doi: 10.1371/JOURNAL.PCBI.1010271. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1010271>.
- Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, Debsindhu Bhowmik, and Burkhard Rost. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:7112–7127, 10 2022. ISSN 19393539. doi: 10.1109/TPAMI.2021.3095381.
- Noelia Ferruz and Birte Höcker. Controllable protein design with language models. *Nature Machine Intelligence*, 4(6):521–532, 2022.
- Noelia Ferruz, Steffen Schmidt, and Birte Höcker. Protgpt2 is a deep unsupervised language model for protein design. *Nature communications*, 13(1):4348, 2022.
- Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. Empowering biomedical discovery with ai agents. *arXiv preprint arXiv:2404.02831*, 2024.
- Joe G. Greener, Lewis Moffat, and David T. Jones. Design of metalloproteins and novel protein folds using variational autoencoders. *Scientific Reports 2018 8:1*, 8:1–12, 11 2018. ISSN 2045-2322. doi: 10.1038/s41598-018-34533-1. URL <https://www.nature.com/articles/s41598-018-34533-1>.
- Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. *arXiv preprint arXiv:2402.01680*, 2024.
- Yiwen Hu and Markus J. Buehler. Deep language models for interpretative and predictive materials science. *APL Machine Learning*, 1:10901, 3 2023. doi: 10.1063/5.0134317. URL [/aip/aml/article/1/1/010901/2878738/Deep-language-models-for-interpretative-and](https://aip/aml/article/1/1/010901/2878738/Deep-language-models-for-interpretative-and).
- Po Ssu Huang, Scott E. Boyken, and David Baker. The coming of age of de novo protein design. *Nature 2016 537:7620*, 537:320–327, 9 2016. ISSN 1476-4687. doi: 10.1038/nature19946. URL <https://www.nature.com/articles/nature19946>.
- John Ingraham, Vikas K Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in Neural Information Processing Systems*, 32, 2019.
- John B. Ingraham, Max Baranov, Zak Costello, Karl W. Barber, Wujie Wang, Ahmed Ismail, Vincent Frappier, Dana M. Lord, Christopher Ng-Thow-Hing, Erik R. Van Vlack, Shan Tie, Vincent Xue, Sarah C. Cowles, Alan Leung, João V. Rodrigues, Claudio L. Morales-Perez, Alex M. Ayoub, Robin Green, Katherine Puentes, Frank Oplinger, Nishant V. Panwar, Fritz Obermeyer, Adam R. Root, Andrew L. Beam, Frank J. Poelwijk, and Gevorg Grigoryan. Illuminating protein space with a programmable generative model. *Nature 2023 623:7989*, 623:1070–1078, 11 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06728-8. URL <https://www.nature.com/articles/s41586-023-06728-8>.
- Kevin Maik Jablonka, Qianxiang Ai, Alexander Al-Feghali, Shruti Badhwar, Joshua D. Bocarsly, Andres M. Bran, Stefan Bringuier, L. Catherine Brinson, Kamal Choudhary, Defne Circi, Sam Cox, Wibe A. de Jong, Matthew L. Evans, Nicolas Gastellu, Jerome Genzling, María Victoria Gil, Ankur K. Gupta, Zhi Hong, Alishba Imran, Sabine Kruschwitz, Anne Labarre, Jakub Lála, Tao Liu, Steven Ma, Sauradeep Majumdar, Garrett W. Merz, Nicolas Moitessier, Elias Moubarak, Beatriz Mouriño, Brenden Pelkie, Michael Pieler, Mayk Caldas Ramos, Bojana Ranković, Samuel G. Rodrigues, Jacob N. Sanders, Philippe Schwaller, Marcus Schwarting, Jiale Shi, Berend Smit, Ben E. Smith, Joren Van Herck, Christoph Völker, Logan Ward, Sean Warren, Benjamin Weiser, Sylvester Zhang, Xiaoqi Zhang, Ghezal Ahmad Zia, Aristana Scourtas, K. J. Schmidt, Ian Foster, Andrew D. White, and Ben Blaiszik. 14 examples of how LLMs

- can transform materials science and chemistry: a reflection on a large language model hackathon. *Digital Discovery*, 2:1233–1250, 10 2023. ISSN 2635098X. doi: 10.1039/D3DD00113J. URL <https://pubs.rsc.org/en/content/articlehtml/2023/dd/d3dd00113j><https://pubs.rsc.org/en/content/articlelanding/2023/dd/d3dd00113j>.
- Wolfgang Kabsch and Christian Sander. Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features. *Biopolymers*, 22(12):2577–2637, 12 1983. ISSN 0006-3525. doi: 10.1002/bip.360221211. URL <http://www.ncbi.nlm.nih.gov/pubmed/6667333><http://doi.wiley.com/10.1002/bip.360221211>.
- Tuomas P.J. Knowles and Markus J. Buehler. Nanomechanics of functional and pathological amyloid materials. *Nature Nanotechnology* 2011 6:8, 6:469–479, 7 2011. ISSN 1748-3395. doi: 10.1038/nnano.2011.102. URL <https://www.nature.com/articles/nnano.2011.102>.
- Wei Lu, David L. Kaplan, and Markus J. Buehler. Generative modeling, design, and analysis of spider silk protein sequences for enhanced mechanical properties. *Advanced Functional Materials*, pp. 2311324, 12 2023. ISSN 1616-3028. doi: 10.1002/ADFM.202311324. URL <https://onlinelibrary.wiley.com/doi/full/10.1002/adfm.202311324><https://onlinelibrary.wiley.com/doi/abs/10.1002/adfm.202311324><https://onlinelibrary.wiley.com/doi/10.1002/adfm.202311324>.
- Rachel K. Luu and Markus J. Buehler. Materials informatics tools in the context of bio-inspired material mechanics. *Journal of Applied Mechanics*, 90, 9 2023. ISSN 0021-8936. doi: 10.1115/1.4062310. URL <https://dx.doi.org/10.1115/1.4062310>.
- Ali Madani, Bryan McCann, Nikhil Naik, Nitish Shirish Keskar, Namrata Anand, Raphael R. Eguchi, Po-Ssu Huang, and Richard Socher. Progen: Language modeling for protein generation. <https://arxiv.org/abs/2004.03497v1>, 3 2020. URL <https://arxiv.org/abs/2004.03497v1>.
- Ali Madani, Ben Krause, Eric R Greene, Subu Subramanian, Benjamin P Mohr, James M Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z Sun, Richard Socher, et al. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, 2023.
- Claudio Mirabello and Gianluca Pollastri. Porter, paleale 4.0: high-accuracy prediction of protein secondary structure and relative solvent accessibility. *Bioinformatics*, 29:2056–2058, 8 2013. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTT344. URL <https://dx.doi.org/10.1093/bioinformatics/btt344>.
- Arata Nakajo, Zacharie Wullemmin, Patrick Metzger, al, Yuxiu Liu, Michael W Murphy, Daniel R Baker, Ana Marija Damjanovi, Burak Koyutürk, Yan-Sheng Li, Joanna I Sułkowska, and Marek Cieplak. Mechanical stretching of proteins—a theoretical survey of the protein data bank. *Journal of Physics: Condensed Matter*, 19:283201, 6 2007. ISSN 0953-8984. doi: 10.1088/0953-8984/19/28/283201. URL <https://iopscience.iop.org/article/10.1088/0953-8984/19/28/283201><https://iopscience.iop.org/article/10.1088/0953-8984/19/28/283201/meta>.
- Bo Ni and Markus J. Buehler. Mechagents: Large language model multi-agent collaborations can solve mechanics problems, generate new data, and integrate knowledge. 2023. URL <https://arxiv.org/abs/2311.08166v1>.
- Bo Ni, David L. Kaplan, and Markus J. Buehler. Forcegen: End-to-end de novo protein generation based on nonlinear mechanical unfolding responses using a protein language diffusion model. *Science Advances*, 2023. doi: 10.1126/sciadv.adl4000.
- Erik Nijkamp, Jeffrey A Ruffolo, Eli N Weinstein, Nikhil Naik, and Ali Madani. Progen2: exploring the boundaries of protein language models. *Cell systems*, 14(11):968–978, 2023.

- Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena Hurtado, Aidan N Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval, 6 2022. ISSN 2640-3498. URL <https://proceedings.mlr.press/v162/notin22a.html>.
- Grace C.Y. Peng, Mark Alber, Adrian Buganza Tepole, William R. Cannon, Suvranu De, Salvador Durabernal, Krishna Garikipati, George Karniadakis, William W. Lytton, Paris Perdikaris, Linda Petzold, and Ellen Kuhl. Multiscale modeling meets machine learning: What can we learn? *Archives of Computational Methods in Engineering*, 28:1017–1037, 5 2021. ISSN 18861784. doi: 10.1007/S11831-020-09405-5/FIGURES/7. URL <https://link.springer.com/article/10.1007/s11831-020-09405-5>.
- Gianluca Pollastri and Aoife McLysaght. Porter: a new, accurate server for protein secondary structure prediction. *Bioinformatics*, 21:1719–1720, 4 2005. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTI203. URL <https://dx.doi.org/10.1093/bioinformatics/bti203>.
- Gianluca Pollastri, Darisz Przybylski, Burkhard Rost, and Pierre Baldi. Improving the prediction of protein secondary structure in three and eight classes using recurrent neural networks and profiles. *Proteins: Structure, Function, and Bioinformatics*, 47:228–235, 5 2002. ISSN 1097-0134. doi: 10.1002/PROT.10082. URL <https://onlinelibrary.wiley.com/doi/full/10.1002/prot.10082><https://onlinelibrary.wiley.com/doi/abs/10.1002/prot.10082><https://onlinelibrary.wiley.com/doi/10.1002/prot.10082>.
- Zhao Qin and Markus J. Buehler. Cooperative deformation of hydrogen bonds in beta-strands and beta-sheet nanocrystals. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 82:061906, 12 2010. ISSN 15393755. doi: 10.1103/PHYSREVE.82.061906/FIGURES/5/MEDIUM. URL <https://journals.aps.org/pre/abstract/10.1103/PhysRevE.82.061906>.
- Nicholas Rego and David Koes. 3dmol.js: molecular visualization with webgl. *Bioinformatics*, 31:1322–1324, 4 2015. ISSN 1367-4803. doi: 10.1093/BIOINFORMATICS/BTU829. URL <https://dx.doi.org/10.1093/bioinformatics/btu829>.
- Adam J. Riesselman, John B. Ingraham, and Debora S. Marks. Deep generative models of genetic variation capture the effects of mutations. *Nature Methods* 2018 15:10, 15:816–822, 9 2018. ISSN 1548-7105. doi: 10.1038/s41592-018-0138-4. URL <https://www.nature.com/articles/s41592-018-0138-4>.
- Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C. Lawrence Zitnick, Jerry Ma, and Rob Fergus. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences of the United States of America*, 118:e2016239118, 4 2021. ISSN 10916490. doi: 10.1073/PNAS.2016239118/SUPPL\_FILE/PNAS.2016239118.SAPP.PDF. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2016239118>.
- Jeffrey A Ruffolo and Ali Madani. Designing proteins with language models. *Nature Biotechnology*, 42(2): 200–202, 2024.
- Mateusz Sikora, Joanna I. Sułkowska, and Marek Cieplak. Mechanical strength of 17 134 model proteins and cysteine slipknots. *PLOS Computational Biology*, 5:e1000547, 2009. ISSN 1553-7358. doi: 10.1371/JOURNAL.PCBI.1000547. URL <https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000547>.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Reformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

- Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, YaGuang Li, Hongrae Lee, Huaixiu Steven Zheng, Amin Ghafouri, Marcelo Menegali, Yanping Huang, Maxim Krikun, Dmitry Lepikhin, James Qin, Dehao Chen, Yuanzhong Xu, Zhifeng Chen, Adam Roberts, Maarten Bosma, Vincent Zhao, Yanqi Zhou, Chung-Ching Chang, Igor Krivokon, Will Rusch, Marc Pickett, Pranesh Srinivasan, Laichee Man, Kathleen Meier-Hellstern, Meredith Ringel Morris, Tulsee Doshi, Renelito Delos Santos, Toju Duke, Johnny Soraker, Ben Zevenbergen, Vinodkumar Prabhakaran, Mark Diaz, Ben Hutchinson, Kristen Olson, Alejandra Molina, Erin Hoffman-John, Josh Lee, Lora Aroyo, Ravi Rajakumar, Alena Butryna, Matthew Lamm, Viktoriya Kuzmina, Joe Fenton, Aaron Cohen, Rachel Bernstein, Ray Kurzweil, Blaise Aguera-Arcas, Claire Cui, Marian Croak, Ed Chi, and Quoc Le. Lamda: Language models for dialog applications. 1 2022. URL <https://arxiv.org/abs/2201.08239v3>.
- Robert Tinn, Hao Cheng, Yu Gu, Tristan Naumann, Jianfeng Gao, Hoifung Poon Correspondence, Naoto Usuyama, Xiaodong Liu, and Hoifung Poon. Fine-tuning large neural language models for biomedical natural language processing highlights d systematic exploration of fine-tuning stability in biomedical nlp d domain-specific vocabulary and pretraining facilitate robust models for fine-tuning d pubmedbert-large and pubmedelectra models advance state-of-the-art in biomedical nlp in brief fine-tuning large neural language models for biomedical natural language processing. *Patterns*, 4:100729, 2023. doi: 10.1016/j.patter.2023.100729. URL <https://doi.org/10.1016/j.patter.2023.100729>.
- Serbulent Unsal, Heval Atas, Muammer Albayrak, Kemal Turhan, Aybar C Acar, and Tunca Doğan. Learning functional properties of proteins with language models. *Nature Machine Intelligence*, 4(3):227–245, 2022.
- Ashish Vaswani, Google Brain, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Ji-Rong Wen. A survey on large language model based autonomous agents. 8 2023a. URL <https://arxiv.org/abs/2308.11432v2>.
- Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. A survey on large language model based autonomous agents. *arXiv preprint arXiv:2308.11432*, 2023b.
- Joseph L. Watson, David Juergens, Nathaniel R. Bennett, Brian L. Trippe, Jason Yim, Helen E. Eisenach, Woody Ahern, Andrew J. Borst, Robert J. Ragotte, Lukas F. Milles, Basile I.M. Wicky, Nikita Hanikel, Samuel J. Pellock, Alexis Courbet, William Sheffler, Jue Wang, Preetham Venkatesh, Isaac Sappington, Susana Vázquez Torres, Anna Lauko, Valentin De Bortoli, Emile Mathieu, Sergey Ovchinnikov, Regina Barzilay, Tommi S. Jaakkola, Frank DiMaio, Minkyung Baek, and David Baker. De novo design of protein structure and function with rfdiffusion. *Nature* 2023 620:7976, 620:1089–1100, 7 2023. ISSN 1476-4687. doi: 10.1038/s41586-023-06415-8. URL <https://www.nature.com/articles/s41586-023-06415-8>.
- Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. Emergent abilities of large language models. 6 2022. URL <https://arxiv.org/abs/2206.07682v2>.
- Kevin E. Wu, Kevin K. Yang, Rianne van den Berg, James Y. Zou, Alex X. Lu, and Ava P. Amini. Protein structure generation via folding diffusion. <https://arxiv.org/abs/2209.15611v2>, 9 2022. URL <https://arxiv.org/abs/2209.15611v2>.

- Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Awadallah, Ryan W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation. *https://arxiv.org/abs/2308.08155*, 2023. doi: <https://arxiv.org/abs/2308.08155>.
- Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. The rise and potential of large language model based agents: A survey. 9 2023. URL <https://arxiv.org/abs/2309.07864v3>.
- Zhiping Xu and Markus J. Buehler. Mechanical energy transfer and dissipation in fibrous beta-sheet-rich proteins. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 81:061910, 6 2010. ISSN 15393755. doi: 10.1103/PHYSREVE.81.061910/FIGURES/4/MEDIUM. URL <https://journals.aps.org/pre/abstract/10.1103/PhysRevE.81.061910>.
- Chi Hua Yu, Wei Chen, Yu Hsuan Chiang, Kai Guo, Zaira Martin Moldes, David L. Kaplan, and Markus J. Buehler. End-to-end deep learning model to predict and design secondary structure content of structural proteins. *ACS Biomaterials Science and Engineering*, 8:1156–1165, 3 2022. ISSN 23739878. doi: 10.1021/ACSBBIOMATERIALS.1C01343/SUPPL\_FILE/AB1C01343\_SI\_001.ZIP. URL <https://pubs.acs.org/doi/full/10.1021/acsbiomaterials.1c01343>.
- Buzhong Zhang, Jinyan Li, and Qiang Lü. Prediction of 8-state protein secondary structures by a novel deep learning architecture. *BMC Bioinformatics*, 19:1–13, 8 2018. ISSN 14712105. doi: 10.1186/S12859-018-2280-5/TABLES/13. URL <https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2280-5>.
- Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. 7 2023. URL <https://arxiv.org/abs/2307.02485v1>.
- Chenxu Zhu, Bo Chen, Huifeng Guo, Hang Xu, Xiangyang Li, Xiangyu Zhao, Weinan Zhang, Yong Yu, and Ruiming Tang. Autogen: An automated dynamic model generation framework for recommender system. *WSDM 2023 - Proceedings of the 16th ACM International Conference on Web Search and Data Mining*, pp. 598–606, 2 2023. doi: 10.1145/3539597.3570456. URL <https://dl.acm.org/doi/10.1145/3539597.3570456>.

## A MATERIALS AND METHODS

### AGENT DESIGN

As shown in Figure 1a, we design AI agents using the state-of-the-art all-purpose LLM GPT-4 and dynamic multi-agent collaboration is implemented in AutoGen framework Zhu et al. (2023), an open-source ecosystem for agent-based AI modeling. Additional agents are introduced as described below, including some based on generative AI as well as physics modeling.

In our multi-agent system, the human *user\_proxy* agent is constructed using `UserProxyAgent` class from `Autogen`, and *Assistant*, *Planner*, *Critic* agents are created via `AssistantAgent` class from `Autogen`; and the group chat manager is created using `GroupChatManager` class. Each agent is assigned a role through a profile description listed in **Table B1**, included as *system\_message* at their creation.

### FUNCTION AND TOOL DESIGN

All the tools implemented in this work are defined as python functions. Each function is characterized by a name, a description, and input properties with a description as tabulated in **Table B2** in Appendix B. The list of functions is incorporated into the multi-agent system, included as the *function\_map* parameter in the `AssistantAgent` at its creation.

### AUTOREGRESSIVE TRANSFORMER MODEL TO PREDICT PROTEIN UNFOLDING FORCE-EXTENSION FROM SEQUENCES

We use a special-purpose GPT-style model denoted as `ProteinForceGPT`, similar as in Buehler (2023c), here trained to predict force-extension curves from sequences along with other mechanical properties, and vice versa (<https://huggingface.co/lamm-mit/ProteinForceGPT>). The protein language model is based on the NeoGPT-X architecture and uses rotary positional embeddings (RoPE) Su et al. (2024). The model has 16 attention heads, 36 hidden layers and a hidden size of 1024, an intermediate size of 4096 and uses GeLU activation functions.

Pre-training was conducted based on a dataset of  $\sim 800,000$  amino acid sequences, using next-token predictions using a “Sequence” task (<https://huggingface.co/datasets/lamm-mit/GPTProteinPretrained>):

```
Sequence<GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
```

The `ProteinForceGPT` model was then fine-tuned bidirectionally, to predict the mechanical properties of proteins from their sequence, as well as sequence candidates that meet a required force-extension behavior and various other properties. Fine-tuning is conducted using a dataset derived from molecular dynamics (MD) simulations Ni et al. (2023). Sample tasks for the model include:

```
CalculateForce<GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN> [0.262]
CalculateEnergy<GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN> [0.220]
CalculateForceEnergy<GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN> [0.262, 0.220]
CalculateForceHistory<GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN>
[0.004, 0.034, 0.125, 0.142, 0.159, 0.102, 0.079, 0.073, 0.131, 0.105, 0.071, 0.058, 0.072, 0.060, 0.049, 0.114,
0.122, 0.108, 0.173, 0.192, 0.208, 0.153, 0.212, 0.222, 0.244]
GenerateForce<0.262> [GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
GenerateForce<0.220> [GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
GenerateForceEnergy<0.262, 0.220> [GEECDGSPSNPCDAATCKLRPGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
GenerateForceHistory<0.004, 0.034, 0.125, 0.142, 0.159, 0.102, 0.079, 0.073, 0.131, 0.105, 0.071, 0.058, 0.072,
0.060, 0.049, 0.114, 0.122, 0.108, 0.173, 0.192, 0.208, 0.153, 0.212, 0.222, 0.244> [GEECDGSPSNPCDAATCKLR
PGAQCADGLCCDQCRFKKKRTICRIARGDFPDDRCTGQSADCPRWN]
```

Sample results from validation of the model are shown in **Figure B3** of Appendix B. We only use forward predictions for use in the agent model reported here. Lastly, the normalized constants to non-dimensionalize the energy and force values are provided in **Table B7** in Appendix B.

#### SOFTWARE VERSIONS AND HARDWARE

We develop our multi-agent models using local workstations with NVIDIA GPUs. We use Python 3.10 and pyautogen-0.2.2Wu et al. (2023). Additional implementation details are included in the code.

#### VISUALIZATION

We use Py3DMolRego & Koes (2015) for visualization of the protein structures.

#### SECONDARY STRUCTURE ANALYSIS

We use the dictionary of protein secondary structure (DSSP)Kabsch & Sander (1983) module via BioPythonCock et al. (2009) to analyze the secondary structure content of the proteins from its geometry.

#### NATURAL VIBRATIONAL FREQUENCY CALCULATIONS

We perform Anisotropic Network Model (ANM)Atilgan et al. (2001); Doruker et al. (2000) calculations as implemented in ProDyBakan et al. (2011) for normal mode analysis. The problem is solved by considering the protein as a network of interactions, defined within a cutoff distance for which spring-like potentials are assumed to define molecular interactions.

#### RETRIEVAL AUGMENTED GENERATION

We use Llama Indexlla as a tool to implement RAG where the full text of papers cited as references Sikora et al. (2009); Nakajo et al. (2007) are used as external sources from which information can be retrieved by the system in real-time.

#### CONFLICT OF INTEREST

The author declares no conflict of interest.

#### DATA AND CODE AVAILABILITY

All data and codes are available on GitHub at <https://github.com/lamm-mit/ProtAgents>. Alternatively, they will be provided by the corresponding author based on reasonable request.

## B SUPPLEMENTARY MATERIAL

Table B1: The profiles of the agents implemented in the current study to solve multi-objective tasks in the context of protein design and analysis.

<b>Agent #</b>	<b>Agent role</b>	<b>Agent profile</b>
1	user_proxy	user_proxy. Plan execution needs to be approved by user_proxy
2	Planner	Planner. You develop a plan. Begin by explaining the plan. Revise the plan based on feedback from the critic and user_proxy, until user_proxy approval. The plan may involve calling custom function for retrieving knowledge, designing proteins, and computing and analyzing protein properties. You include the function names in the plan and the necessary parameters. If the plan involves retrieving knowledge, retain all the key points of the query asked by the user for the input message.
3	Assistant	Assistant. You have access to all the custom functions. You focus on executing the functions suggested by the planner or the critic. You also have the ability to prepare the required input parameters for the functions.
4	Critic	user_proxy. You double-check the plan, especially the functions and function parameters. Check whether the plan included all the necessary parameters for the suggested function. You provide feedback.
5	Group chat manager	You repeat the following steps: Dynamically selecting a speaker, collecting responses, and broadcasting the message to the group.



Table B2: The library of functions implemented in the current study to perform a wide range of tasks in the context of protein design and analysis. Each function is described by a unique profile and takes at least one parameter as input. Similar to the functions, the input parameters are also described by a profile.

#	Function name	Function profile	Function parameters	Parameters profile
1	save_to_csv_file	With a JSON dictionary as input, saves the data to a csv file with a provided name	input_JSON_dictionary output_csv_name	The input JSON dictionary. The output name for the csv file
2	analyze_protein_structure	Given the protein structure file as input, analyzes and returns the secondary structure of the protein. The function returns a JSON dictionary with % content of the 8 secondary structure types. The 8 secondary structures are ['H': alpha-helix], ['B': isolated beta bridge], ['E': Extended strand or beta-sheet], ['G': 3-helix (3/10 helix)], ['I': 5 helix (pi-helix)], ['T': Hydrogen bonded turn], ['S': Bend], ['P': Poly-proline helices] and ['-': None]	protein_structure	Protein structure file
3	calculate_force_energy_from_seq	Calculates the unfolding energy and maximum force of a protein. The function requires the amino acid sequence of a protein structure in string format.	sequence	Amino acid sequence in single-letter FASTA code.
4	fold_protein	Fold a protein with required amino acid sequence which creates a protein structure file. The function also returns the output protein structure name.	sequence name	Amino acid sequence in single-letter FASTA code. Name of the folded output protein structure.
5	retrieve_content	An expert in retrieving knowledge about protein, their mechanical properties, structures, and PDB names.	message	Message to be used to retrieve detailed knowledge.
6	design_protein_from_length	With an optional caption and required length of the protein (number of amino acids) and a name as input, designs a new protein. Returns a PDB name and the amino acid sequence.	length name	Length of the protein to be designed. Name of the protein to be saved.

7	design_protein_from_CATH	With a required CATH_ANNOTATION domain (1 is mainly alpha, 2 is mainly beta, 3 is alpha beta), required output protein name, and length of the protein (number of amino acids) as input, designs a protein and creates a protein structure file. It returns the PDB file name and the amino acid sequence.	name	Name of the protein to be saved.
			length	Length of the protein to be designed.
			CATH_ANNOTATION	CATH_ANNOTATION that describes the protein CATH domain (1 is mainly alpha, 2 is mainly beta, 3 is alpha beta) that will be designed. Default is 2 (mainly beta)
8	calc_protein_ANM.	With input of a protein structure file, calculates the frequencies of the first n_modes eigenmodes.	protein_structure	Protein structure file.
			n_modes	Number of modes to be calculated.
9	fetch_protein_structure_from_PDBID	With input of a protein PDB id, fetches the protein structure file and stores it in the directory. It also returns the name of the file stored in the directory. The input must be a PDB id of the protein not a generated protein name.	PDB_id	Protein PDB id
10	analyze_protein_CATH_from_PDBID	With input of a protein PDB id, you collect info about the CATH domain or structure classification of the protein.	PDB_id	PDB id
11	analyze_protein_length_from_PDB	With input of a protein PDB id or protein name, you give the length of the amino-acid sequence of the protein.	PDB_id	Protein PDB id
12	analyze_protein_seq_from_PDB	With input of a protein PDB id or protein name, you give the sequence of the amino-acid sequence of the protein.	PDB_id	Protein PDB id

Table B3: The final results generated by the group chat in the form of a CSV file, without human intervention, for the first experiment, Section 4.1.

Protein Number #	Amino Acid Sequence	Secondary Structure (Pre-Fold)	Frequencies (Pre-Fold)	Secondary Structure (Post-Fold)	Frequencies (Post-Fold)
1	MIINIKTENGLSIT YNSDEKKLELKY TPVKSPEDFKFPE DAKATISEVEYK KKVIKIDAKLYVS PDLKAKLTIEVN ADISQEEADKIIDE FIKLLLESLGNIKLK VTKDGNKYTIEV E	{'H': 13.33, 'B': 0.0, 'E': 46.67, 'G': 0.0, 'I': 0.0, 'T': 14.17, 'S': 7.5, 'P': 0.0, '-': 18.3333333333}	[2.0337, 2.8678, 3.3843, 3.6263, 3.9904, 4.5381, 4.8373, 4.8956, 5.1492, 5.4416]	{'H': 15.8333333333, 'B': 0.0, 'E': 46.67, 'G': 2.5, 'I': 0.0, 'T': 14.17, 'S': 4.17, 'P': 0.0, '-': 16.67}	[1.8739, 2.1563, 2.7611, 3.1086, 3.8712, 4.0481, 4.3759, 4.6717, 4.8183, 4.9126]
2	GSPLRPPLSPEEQ EALRKKAEKYN EFVSKIKELLRRA ADRVRRGEPVEL IEKTIKIGDYEYK IVATSPEEAKELE NLIKEMIDLGFKP SKEFSDKLVEAAR LIREGRVDEALRL LDEM	{'H': 61.67, 'B': 0.0, 'E': 11.67, 'G': 0.0, 'I': 0.0, 'T': 7.5, 'S': 3.33, 'P': 3.33, '-': 12.5}	[0.0207, 0.1058, 0.1782, 0.4189, 0.49, 0.9015, 1.1832, 1.8257, 2.1212, 2.8726]	{'H': 62.5, 'B': 0.0, 'E': 11.67, 'G': 0.0, 'I': 0.0, 'T': 6.67, 'S': 1.67, 'P': 4.17, '-': 13.33}	[0.0444, 0.1641, 0.3379, 0.5724, 0.765, 0.9568, 1.4306, 1.5344, 1.6834, 1.8099]
3	APLDPDDLSAQLR AAIDELVRLGYEE EVSKPEFIEALRL YALDLGLKEVVL RRVTPAPASQPGV YTVEDVTVDLEA LRKQELSPEEQAR LEKIRAKYDEML ADPEFQALLDEVL ARARAA	{'H': 57.5, 'B': 0.0, 'E': 13.33, 'G': 0.0, 'I': 4.17, 'T': 8.33, 'S': 3.33, 'P': 6.67, '-': 6.67}	[0.7546, 1.0836, 1.5026, 1.8874, 2.0844, 2.3192, 2.7975, 3.0199, 3.0669, 3.1382]	{'H': 61.67, 'B': 0.0, 'E': 15.0, 'G': 0.0, 'I': 0.0, 'T': 8.33, 'S': 3.33, 'P': 1.67, '-': 10.0}	[0.5256, 1.0278, 1.1566, 1.2877, 1.5521, 1.9111, 2.1887, 2.4664, 2.734, 2.8731]

Table B4: The complete chat records between the agents for the first experiment, Section 4.1.

<p>user_proxy (to chat_manager):  Design three proteins with sequence length of 120 and analyze their secondary structure and first 10 frequencies. Then fold the proteins and repeat the analysis. Save all the results (including the AA sequences) into a csv file where each row corresponds to a protein and each column represents a protein property, utilizing proper identifiers.</p>
<p>Planner (to chat_manager):</p> <p>Plan:</p> <ol style="list-style-type: none"> <li>1. Use the 'design_protein_from_length' function three times to design three proteins each with a sequence length of 120. I will use 1, 2, 3 as names for the proteins. Save the amino acid sequences and PDB filenames.</li> <li>2. Use the 'analyze_protein_structure' function to analyze the secondary structure of the proteins. Store the results.</li> <li>3. Use the 'calc_protein_ANM' function to calculate the first 10 frequencies of the proteins. Store the results.</li> <li>4. Use the 'fold_protein' function to fold the proteins.</li> <li>5. Repeat steps 2 and 3 with the folded proteins.</li> <li>6. Use the 'save_to_csv_file' function to save all results into a csv file where each row corresponds to a protein and each column represents a protein property.</li> <li>7. The identifiers will be named as follows: <ul style="list-style-type: none"> <li>- Protein Name</li> <li>- Amino Acid Sequence (Pre-Fold)</li> <li>- Secondary Structure (Pre-Fold)</li> <li>- Frequencies (Pre-Fold)</li> <li>- Amino Acid Sequence (Post-Fold)</li> <li>- Secondary Structure (Post-Fold)</li> <li>- Frequencies (Post-Fold)</li> </ul> </li> </ol> <p>Please confirm if this plan works for you.</p>
<p>Critic (to chat_manager):</p> <p>Your plan is almost perfect. However, when a protein is folded, the protein sequence doesn't change, only the structure does, which then affects the secondary structure and possible frequencies you can calculate. So you need to only save Amino Acid Sequence once after designing the proteins. Each protein should have the following column properties:</p> <ul style="list-style-type: none"> <li>- Protein Number</li> <li>- Amino Acid Sequence</li> <li>- Secondary Structure (Pre-Fold)</li> <li>- Frequencies (Pre-Fold)</li> <li>- Secondary Structure (Post-Fold)</li> <li>- Frequencies (Post-Fold)</li> </ul> <p>Other than that, your plan works perfectly. So steps will be as follows:</p> <ol style="list-style-type: none"> <li>1. Use the 'design_protein_from_length' function three times to design three proteins each with a sequence length of 120. I will use 1, 2, 3 as numbers for the proteins. Save the amino acid sequences and PDB filenames.</li> <li>2. Use the 'analyze_protein_structure' function to analyze the secondary structure of the proteins. Store the results.</li> <li>3. Use the 'calc_protein_ANM' function to calculate the first 10 frequencies of the proteins. Store the results.</li> </ol>

```
4. Use the 'fold_protein' function to fold the proteins. Get the new structure names.
5. Repeat steps 2 and 3 with the new folded protein structure.
6. Use the 'save_to_csv_file' function to save all results into a csv file where each row corresponds to a
protein and each column represents a protein property.
Please note that in step 3, 'calc_protein_ANM' will require 'n_modes: 10' to calculate the first 10 fre-
quencies.
```

---

```
Provide feedback to chat_manager. Press enter to skip and use auto-reply, or type 'exit' to end the
conversation: Excellent! user_proxy (to chat_manager):
Excellent!
```

---

```
assistant (to chat_manager):
**** Suggested function Call: design_protein_from_length ****
Arguments:
{
"length": 120,
"name": 1
}
*****
```

---

```
>>>>>>> EXECUTING FUNCTION design_protein_from_length...
assistant (to chat_manager):
**** Response from calling function "design_protein_from_length" ****
('./code_protein/1.pdb', 'MIIINIKTENGLSITYNSDEKKLELKYTPVKSPEDFKFPEDAKATISEVEYKGGKVIKIDAKLYV
SPDLSKAKLTIEVNADISQEEADKIIDEFIKLLLESLGNIKLVTKDGNKYTIEVE')
*****
```

---

```
assistant (to chat_manager):
**** Suggested function Call: design_protein_from_length ****
Arguments:
{
"length": 120,
"name": 2
}
*****
```

---

```
>>>>>>> EXECUTING FUNCTION design_protein_from_length...
assistant (to chat_manager):
**** Response from calling function "design_protein_from_length" ****
('./code_protein/2.pdb', 'GSPLPRPPLSPEEQEALRKKAEKYNEFVSKIKELLRRAADRVRGEP
VELIEKTIKIGDYEYKIVATSPEEAKELLENLIKEMIDLGFKPSKEFSDKLVEAARLIREGRVDEA
LRLLED') *****
```

---

```
assistant (to chat_manager):
**** Suggested function Call: design_protein_from_length ****
Arguments:
{
"length": 120,
"name": 3
}
*****
```

---

```
>>>>>>> EXECUTING FUNCTION design_protein_from_length...
assistant (to chat_manager):
**** Response from calling function "design_protein_from_length" ****
```

```
( './code_protein/3.pdb', 'APLDPDDLSQLRAAIDELVRLGYEEEVSKPEFIEALRLYALDLGLKE  
VVLRRVTPAPASQPGVYTVEDVTVDLEALRKQELSPEEQARLEKIRAKYDEMLADPEFQALL  
DEVLARARAA')
```

```
*****
```

Critic (to chat\_manager):

```
***** Suggested function Call: analyze_protein_structure *****
```

Arguments:

```
{  
"protein_structure": "./code_protein/1.pdb"  
}
```

```
*****
```

```
>>>>>>> EXECUTING FUNCTION analyze_protein_structure...
```

Analyzing the secondary structure of this protein: ./code\_protein/1.pdb

The protein analyzed has 120 residues.

assistant (to chat\_manager):

```
***** Response from calling function "analyze_protein_structure" *****
```

```
{  
  "H": 13.333333333333334,  
  "B": 0.0,  
  "E": 46.666666666666664,  
  "G": 0.0,  
  "I": 0.0,  
  "T": 14.166666666666666,  
  "S": 7.5,  
  "P": 0.0,  
  "-": 18.333333333333332  
}
```

```
*****
```

assistant (to chat\_manager):

```
***** Suggested function Call: analyze_protein_structure *****
```

Arguments:

```
{  
"protein_structure": "./code_protein/2.pdb"  
}
```

```
*****
```

```
~~~~~ EXECUTING FUNCTION analyze_protein_structure...
```

Analyzing the secondary structure of this protein: ./code\_protein/2.pdb

The protein analyzed has 120 residues.

assistant (to chat\_manager):

```
***** Response from calling function "analyze_protein_structure" *****
```

```
{  
  "H": 61.666666666666667,  
  "B": 0.0,  
  "E": 11.666666666666666,  
  "G": 0.0,  
  "I": 0.0,  
  "T": 7.5,  
  "S": 3.3333333333333335,  
  "P": 3.3333333333333335,  
  "-": 12.5  
}
```

```
*****
assistant (to chat_manager):
**** Suggested function Call: analyze_protein_structure ****
Arguments:
{
"protein_structure": "/code_protein/3.pdb"
}
*****

>>>>>>> EXECUTING FUNCTION analyze_protein_structure...
Analyzing the secondary structure of this protein: /code_protein/3.pdb
The protein analyzed has 120 residues.
assistant (to chat_manager):
**** Response from calling function "analyze_protein_structure" ****
{
  "H": 57.49999999999999,
  "B": 0.0,
  "E": 13.333333333333334,
  "G": 0.0,
  "I": 4.166666666666666,
  "T": 8.333333333333332,
  "S": 3.333333333333335,
  "P": 6.666666666666667,
  "-": 6.666666666666667
} *****

assistant (to chat_manager):
**** Suggested function Call: calc_protein_ANM ****
Arguments:
{
"protein_structure": "/code_protein/1.pdb",
"n_modes": 10
}
*****

lllllll EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
**** Response from calling function "calc_protein_ANM" ****
[
  2.0337,
  2.8678,
  3.3843,
  3.6263,
  3.9904,
  4.5381,
  4.8373,
  4.8956,
  5.1492,
  5.4416
]
*****

assistant (to chat_manager):
**** Suggested function Call: calc_protein_ANM ****
Arguments:
```

```
{
"protein_structure": "/code_protein/2.pdb",
"n_modes": 10
}
*****
```

```
>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
***** Response from calling function "calc_protein_ANM" *****
[
  0.0207,
  0.1058,
  0.1782,
  0.4189,
  0.49,
  0.9015,
  1.1832,
  1.8257,
  2.1212,
  2.8726
]
*****
```

```
assistant (to chat_manager):
***** Suggested function Call: calc_protein_ANM *****
Arguments:
{
"protein_structure": "/code_protein/3.pdb",
"n_modes": 10
}
*****
```

```
~~~~~ EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
***** Response from calling function "calc_protein_ANM" *****
[
  0.7546,
  1.0836,
  1.5026,
  1.8874,
  2.0844,
  2.3192,
  2.7975,
  3.0199,
  3.0669,
  3.1382
]
*****
```

```
Planner (to chat_manager):
***** Suggested function Call: fold_protein ***** Arguments:
{
"sequence": "MIIINIKTENGLSITYNSDEKKLELKYTPVKSPEDFKFPEDAKATISEVEYKGGKVI
KIDAKLYVSPDLSKAKLTIEVNADISQEEADKIIDEFIKLLLESLGNIKLKVTKDGNKYTIEVE",
"name": "1_folded"
}
*****
```



```
*****
>>>>>>> EXECUTING FUNCTION fold_protein...
assistant (to chat_manager):
**** Response from calling function "fold_protein" ****
1_folded.pdb *****

assistant (to chat_manager):
**** Suggested function Call: fold_protein ****
Arguments:
{
"sequence": "GSPLRPPLSPEEQEALRKKAEKYNEFVSKIKELLRRAADRVRGEPVELIEKTI
KIGDYEYKIVATSPEEAKELLENLIKEMIDLGFKPSKEFSDKLVEAARLIREGRVDEALRLDEM
",
"name": "2_folded"
}
*****

>>>>>>> EXECUTING FUNCTION fold_protein...
assistant (to chat_manager):
**** Response from calling function "fold_protein" ****
2_folded.pdb
*****

assistant (to chat_manager):
**** Suggested function Call: fold_protein **** Arguments:
{
"sequence": "APLDPDDLSQLRAAIDELVRLGYEEEVSKPEFIEALRLYALDLGLKEVLRRT
PAPASQPGVYTVEDVTVDLEALRKQELSPPEQARLEKIRAKYDEMLADPEFQALLDEV LARA
RAA",
"name": "3_folded"
}
*****

>>>>>>> EXECUTING FUNCTION fold_protein...
assistant (to chat_manager):
**** Response from calling function "fold_protein" ****
3_folded.pdb
*****

assistant (to chat_manager):
**** Suggested function Call: analyze_protein_structure **** Arguments:
"protein_structure": "1_folded.pdb" *****

>>>>>>> EXECUTING FUNCTION analyze_protein_structure...
Analyzing the secondary strucute of this protein: code_protein/1_folded.pdb
The protein analyzed has 120 residues.
assistant (to chat_manager):
**** Response from calling function "analyze_protein_structure" ****
{
"H": 15.833333333333332,
"B": 0.0,
"E": 46.666666666666664,
"G": 2.5,
"I": 0.0,
"T": 14.166666666666666,
"S": 4.166666666666666,
```

<pre>"P": 0.0, "-": 16.666666666666664 } *****</pre>
<pre>assistant (to chat_manager): ***** Suggested function Call: analyze_protein_structure ***** Arguments: { "protein_structure": "2_folded.pdb" } *****</pre>
<pre>&gt;&gt;&gt;&gt;&gt;&gt;&gt; EXECUTING FUNCTION analyze_protein_structure... Analyzing the secondary strucute of this protein: code_protein/2_folded.pdb The protein analyzed has 120 residues. assistant (to chat_manager): ***** Response from calling function "analyze_protein_structure" ***** {   "H": 62.5,   "B": 0.0,   "E": 11.666666666666666,   "G": 0.0,   "I": 0.0,  "T": 6.666666666666667,   "S": 1.666666666666667,   "P": 4.166666666666666,   "-": 13.333333333333334 } *****</pre>
<pre>assistant (to chat_manager): ***** Suggested function Call: analyze_protein_structure ***** Arguments: { "protein_structure": "3_folded.pdb" } *****</pre>
<pre>&gt;&gt;&gt;&gt;&gt;&gt;&gt; EXECUTING FUNCTION analyze_protein_structure... Analyzing the secondary strucute of this protein: code_protein/3_folded.pdb The protein analyzed has 120 residues. assistant (to chat_manager): ***** Response from calling function "analyze_protein_structure" ***** {   "H": 61.666666666666667,   "B": 0.0,   "E": 15.0,   "G": 0.0,   "I": 0.0,   "T": 8.333333333333332,   "S": 3.3333333333333335,   "P": 1.666666666666667,   "-": 10.0 } *****</pre>

```
assistant (to chat_manager):
**** Suggested function Call: calc_protein_ANM ****
Arguments:
{
"protein_structure": "1_folded.pdb",
"n_modes": 10
}
*****
```

```
>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
**** Response from calling function "calc_protein_ANM" ****
[
  1.8739,
  2.1563,
  2.7611,
  3.1086,
  3.8712,
  4.0481,
  4.3759,
  4.6717,
  4.8183,
  4.9126
]
*****
```

```
assistant (to chat_manager):
**** Suggested function Call: calc_protein_ANM ****
Arguments:
{
"protein_structure": "2_folded.pdb",
"n_modes": 10
}
*****
```

```
>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
**** Response from calling function "calc_protein_ANM" ****
[
  0.0444,
  0.1641,
  0.3379,
  0.5724,
  0.765,
  0.9568,
  1.4306,
  1.5344,
  1.6834,
  1.8099
]
*****
```

```
assistant (to chat_manager):
**** Suggested function Call: calc_protein_ANM ****
Arguments:
{ "protein_structure": "3_folded.pdb",
```

```
"n_modes": 10
} *****

>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):
**** Response from calling function "calc_protein_ANM" ****
[
  0.5256,
  1.0278,
  1.1566,
  1.2877,
  1.5521,
  1.9111,
  2.1887,
  2.4664,
  2.734,
  2.8731
]
*****

Planner (to chat_manager):
**** Suggested function Call: save_to_csv_file ****
Arguments:
{
  "input_JSON_dictionary": "{
  'Protein Number': ['1', '2', '3'],
  'Amino Acid Sequence': [
  'MIIINIKTENGLSITYNSDEKKLELKYTPVKSPEDFKFPEDAKATISEVEYKGGKVKIKIDAKLY
  VSPDLKAKLTIEVNADISQEEADKIIDEFIKLLLESLGNIKLKVTKDGKNKYTIEVE', 'GSPLPRP
  PLSPEEQEALRKKKAQEKYNEFVSKIKELLRRAADRVRGEPVELIEKTIKIGDYEYKIVATSP
  EEAKELDENLIKEMIDLGFKPSKEFSDKLVEAARLIREGRVDEALRLLED', 'APLDPDDLSA
  QLRAAIDELVRLGYEEEVSKPEFIEALRLYALDLGLKEVVLRRVTPAPASQPGVYTVEDVTV
  DLEALRKQELSPEEQARLEKIRAKYDEMLADPEFQALLDEVLARARAA' ], 'Secondary Struc-
  ture (Pre-Fold)': [ {'H': 13.333333333333334, 'B': 0.0, 'E': 46.666666666666664, 'G': 0.0, 'I': 0.0,
  'T': 14.166666666666666, 'S': 7.5, 'P': 0.0, '-': 18.333333333333332}, {'H': 61.66666666666667,
  'B': 0.0, 'E': 11.666666666666666, 'G': 0.0, 'I': 0.0, 'T': 7.5, 'S': 3.333333333333335, 'P':
  3.333333333333335, '-': 12.5}, {'H': 57.49999999999999, 'B': 0.0, 'E': 13.333333333333334,
  'G': 0.0, 'I': 4.166666666666666, 'T': 8.333333333333332, 'S': 3.333333333333335, 'P':
  6.666666666666667, '-': 6.666666666666667} ], 'Frequencies (Pre-Fold)': [ [2.0337, 2.8678, 3.3843,
  3.6263, 3.9904, 4.5381, 4.8373, 4.8956, 5.1492, 5.4416], [0.0207, 0.1058, 0.1782, 0.4189, 0.49, 0.9015,
  1.1832, 1.8257, 2.1212, 2.8726], [0.7546, 1.0836, 1.5026, 1.8874, 2.0844, 2.3192, 2.7975, 3.0199,
  3.0669, 3.1382] ], 'Secondary Structure (Post-Fold)': [ {'H': 15.833333333333332, 'B': 0.0, 'E':
  46.666666666666664, 'G': 2.5, 'I': 0.0, 'T': 14.166666666666666, 'S': 4.166666666666666, 'P': 0.0,
  '-': 16.666666666666664}, {'H': 62.5, 'B': 0.0, 'E': 11.666666666666666, 'G': 0.0, 'I': 0.0, 'T':
  6.666666666666667, 'S': 1.666666666666667, 'P': 4.166666666666666, '-': 13.333333333333334},
  {'H': 61.66666666666667, 'B': 0.0, 'E': 15.0, 'G': 0.0, 'I': 0.0, 'T': 8.333333333333332, 'S':
  3.333333333333335, 'P': 1.666666666666667, '-': 10.0} ], 'Frequencies (Post-Fold)': [ [1.8739,
  2.1563, 2.7611, 3.1086, 3.8712, 4.0481, 4.3759, 4.6717, 4.8183, 4.9126], [0.0444, 0.1641, 0.3379,
  0.5724, 0.765, 0.9568, 1.4306, 1.5344, 1.6834, 1.8099], [0.5256, 1.0278, 1.1566, 1.2877, 1.5521,
  1.9111, 2.1887, 2.4664, 2.734, 2.8731] ] }",
  "output_csv_name": "protein_data.csv"
}
*****

>>>>>>> EXECUTING FUNCTION save_to_csv_file...
```

```
assistant (to chat_manager):
**** Response from calling function "save_to_csv_file" ****
Error: Expecting property name enclosed in double quotes: line 2 column 5 (char 6)
*****
```

```
Critic (to chat_manager):
I apologize for the mistake. It seems there was an issue with the formatting of the input JSON dictionary
for saving to the CSV file. Let's try again with properly formatted JSON.
Plan to fix the error:
1. Correctly format JSON with double quotes instead of single quotations.
2. Call 'save_to_csv_file' function again with correctly formatted JSON dictionary.
I'll directly proceed with this plan as the error was due to syntax not plan.
Here is the corrected data:
**** Suggested function Call: save_to_csv_file ****
Arguments:
{ "input_JSON_dictionary": " { \"Protein Number \": [\"1\", \"2\", \"3\"], \"Amino Acid Se-
quence\": [\"MIIINIKTENGLSITYNSDEKKLELKYTPVKSPEDFKFPEDAKATISEVEYKGGK
KVIKIDAKLYVSPDLSKAKLTIEVNADISQEEADKIIDEFIKLLLESLGNIKLKVTKDGNYKT
IEVE\", \"GSPLPRPPLSPEEQEALRKAQEKYNEFVSKIKELLRRAADRVRERGEPVELIEKT
IKIGDYEYKIVATSPEEAKELLENLIKEMIDLGFKPSKEFSDKLVEAARLIREGRVDEALRLL
DEM\", \"APLDPDDLSAQLRAAIDELVRLGYEEEVSKPEFIEALRLYALDLGLKEVVLRRV
TPAPASQPGVYTVEDVTVDLEALRKQELSPEEQARLEKIRAKYDEMLADPEFQALLDEVL
ARARAA\"], \"Secondary Structure (Pre-Fold)\": [ { \"H\": 13.333333333333334, \"B\": 0.0,
\"E\": 46.666666666666664, \"G\": 0.0, \"I\": 0.0, \"T\": 14.166666666666666, \"S\":
7.5, \"P\": 0.0, \"-\": 18.333333333333332}, { \"H\": 61.666666666666667, \"B\": 0.0,
\"E\": 11.666666666666666, \"G\": 0.0, \"I\": 0.0, \"T\": 7.5, \"S\": 3.333333333333335,
\"P\": 3.333333333333335, \"-\": 12.5}, { \"H\": 57.499999999999999, \"B\": 0.0, \"E\":
13.333333333333334, \"G\": 0.0, \"I\": 4.166666666666666, \"T\": 8.333333333333332, \"S\":
3.333333333333335, \"P\": 6.666666666666667, \"-\": 6.666666666666667}], \"Frequencies
(Pre-Fold)\": [[2.0337, 2.8678, 3.3843, 3.6263, 3.9904, 4.5381, 4.8373, 4.8956, 5.1492,
5.4416],[0.0207, 0.1058, 0.1782, 0.4189, 0.49, 0.9015, 1.1832, 1.8257, 2.1212, 2.8726],[0.7546,
1.0836, 1.5026, 1.8874, 2.0844, 2.3192, 2.7975, 3.0199, 3.0669, 3.1382]], \"Secondary Structure
(Post-Fold)\": [ { \"H\": 15.833333333333332, \"B\": 0.0, \"E\": 46.666666666666664, \"G\":
2.5, \"I\": 0.0, \"T\": 14.166666666666666, \"S\": 4.166666666666666, \"P\": 0.0, \"-\":
16.666666666666664}, { \"H\": 62.5, \"B\": 0.0, \"E\": 11.666666666666666, \"G\": 0.0, \"I\":
0.0, \"T\": 6.666666666666667, \"S\": 1.666666666666667, \"P\": 4.166666666666666, \"-\":
13.333333333333334}, { \"H\": 61.666666666666667, \"B\": 0.0, \"E\": 15.0, \"G\": 0.0, \"I\":
0.0, \"T\": 8.333333333333332, \"S\": 3.333333333333335, \"P\": 1.666666666666667, \"-\":
10.0}], \"Frequencies (Post-Fold)\": [[1.8739, 2.1563, 2.7611, 3.1086, 3.8712, 4.0481, 4.3759,
4.6717, 4.8183, 4.9126],[0.0444, 0.1641, 0.3379, 0.5724, 0.765, 0.9568, 1.4306, 1.5344, 1.6834,
1.8099],[0.5256, 1.0278, 1.1566, 1.2877, 1.5521, 1.9111, 2.1887, 2.4664, 2.734, 2.8731]]\",
\"output_csv_name\": \"protein_data.csv\"
}
*****
```

```
>>>>>>> EXECUTING FUNCTION save_to_csv_file...
the results have been saved to csv file: protein_data.csv
assistant (to chat_manager):
**** Response from calling function "save_to_csv_file" ****
None
*****
```

Critic (to chat\_manager): 0 The plan was executed successfully and all data including Amino Acid Sequence, Secondary Structure (Pre-Fold and Post-Fold), and Frequencies (Pre-Fold and Post-Fold) for three proteins have been saved into a CSV file named 'protein\_data.csv'. Each row of the file corresponds to a protein and each column represents a protein property. Let me know if you need any further assistance.

Provide feedback to chat\_manager. Press enter to skip and use auto-reply, or type 'exit' to end the conversation: Good job, Thanks a lot!  
user\_proxy (to chat\_manager):  
Good job, Thanks a lot!

Planner (to chat\_manager):  
You're welcome! I'm glad I could assist. If you have any other tasks or need further assistance, feel free to ask. Have a great day!

assistant (to chat\_manager):  
TERMINATE

Table B5: The final results generated by the group chat in the form of a CSV file, without human intervention, for the second experiment, Section 4.2.

Protein Name #	AA Sequence	Secondary Structure	Unfolding Energy	Max Force	First 10 Frequencies
mainly_alpha_protein_1	SMKKIEDYIREKLLK ALGLSDEEIEERVK QLMEGIKNPKKFEK ELQKRNDRESLLIF KEAYALYEASKDK EKGKKLINKVQSER DKWETEQAEEAARA AAAA	{'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'I': 0.0, 'T': 4.0, 'S': 1.0, 'P': 0.0, '-': 6.0}	0.381	0.444	[0.2329, 0.4901, 0.9331, 1.3741, 1.7347, 2.1598, 2.3686, 2.6359, 2.8555, 3.0364]
mainly_alpha_protein_2	MSKKEIEELKKKLD EIVETLKEYARQGD DACKKAADLIEEVK KALEEGNPEKYSQL KKKLTDAINKAIEE YRKRFEAEGKPEEA QKVIDKLLKKILDEIT N	{'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'I': 0.0, 'T': 5.0, 'S': 0.0, 'P': 0.0, '-': 6.0}	0.376	0.536	[1.6126, 2.0783, 2.3073, 2.4565, 3.399, 3.475, 4.1377, 4.7104, 4.8864, 5.2187]
mainly_beta_protein_1	TTVTVTPPVADADG NEHSTVTAYGNKV TITITCPSNCTVTET VDGVAKTLGTVSG NQTITETRTIAPDEV VTRTYTCTPNASAT SSKTQTVTIKGSQP AP	{'H': 0.0, 'B': 0.0, 'E': 64.0, 'G': 0.0, 'I': 0.0, 'T': 10.0, 'S': 6.0, 'P': 0.0, '-': 20.0}	0.462	0.533	[1.2806, 1.5057, 1.9846, 2.1025, 2.4723, 2.702, 2.9931, 3.1498, 3.4432, 4.1685]
mainly_beta_protein_2	SLKAKNLEEMIKEA EKLGYRDEVEKIIN EIRDKFKKLGVKISE KTLAYIAYLRLLGV KIDWDKIKKVKKA TPADFRVSEEDLKK PEIQKILEKIKKEIN	{'H': 58.0, 'B': 0.0, 'E': 8.0, 'G': 6.0, 'I': 0.0, 'T': 8.0, 'S': 4.0, 'P': 3.0, '-': 13.0}	0.371	0.548	[2.8864, 4.3752, 4.5928, 4.8295, 5.0854, 5.5618, 5.8646, 6.007, 6.3847, 7.1246]
alpha_beta_protein_1	APT VKTFEDTINGQ KVTVTVTASPGGKI TIKTSPGYGDEVAK AFIEELKKQNVLES YKVESAPGKETTIS DVKVKSGATVTFY VINNGKKGKEYSVT VDA	{'H': 15.0, 'B': 0.0, 'E': 59.0, 'G': 3.0, 'I': 0.0, 'T': 12.0, 'S': 1.0, 'P': 0.0, '-': 10.0}	0.424	0.535	[2.4383, 2.5651, 3.3175, 3.8231, 3.9673, 4.2655, 4.6393, 5.1509, 5.6023, 5.9555]
alpha_beta_protein_2	MELKVTEKKGKGD YKVKVIELNTPDKR YIIIESDASRESLIKA AEALLQGKEVEPTP VNEKNVVLFEDED VKTSIERSKCLKFKSD NPEENIKKALEYLL K	{'H': 35.0, 'B': 0.0, 'E': 29.0, 'G': 0.0, 'I': 0.0, 'T': 3.0, 'S': 12.0, 'P': 3.0, '-': 18.0}	0.376	0.543	[2.8756, 3.8895, 4.0594, 4.2831, 4.5542, 5.171, 5.3661, 5.4312, 6.1964, 6.3066]

Table B6: The complete chat records between the agents for the second experiment, Section 4.2.

<p>user_proxy (to chat_manager):</p> <p>Design two proteins of length 100 for each family of mainly beta, mainly alpha, and alpha beta proteins. Then compute the maximum force and unfolding energy, analyze the secondary structure and compute the first 10 frequencies for these proteins.</p>
<p>Planner (to chat_manager):</p> <p>Plan 1:</p> <ol style="list-style-type: none"> <li>1. Use the "design_protein_from_CATH" function with inputs of length of 100, CATH_ANNOTATION of 1 (mainly alpha), and a protein name for the first mainly alpha protein. Repeat with a different name for the second protein.</li> <li>2. Use the same function again with inputs of length 100, CATH_ANNOTATION of 2 (mainly beta), and a protein name for the first mainly beta protein. Repeat with a different name for the second protein.</li> <li>3. Use the function once more with inputs of length 100, CATH_ANNOTATION of 3 (alpha beta), and a protein name for the first alpha beta protein. Repeat with a different name for the second protein.</li> <li>4. Use the "analyze_protein_structure" function on each of the protein structures obtained.</li> <li>5. Use the "calucalte_force_energy_from_seq" function to compute the maximum force and unfolding energy for each of the proteins.</li> <li>6. Use the "calc_protein_ANM" function to compute the first 10 frequencies for each protein.</li> </ol> <p>Does this plan sound good?</p>
<p>Critic (to chat_manager):</p> <p>Yes, this plan sounds good. It covers each step required for the task. The parameter inputs for each function match with the required parameters as stated in their function definitions. Thus, this plan should successfully accomplish the task.</p>
<p>assistant (to chat_manager):</p> <p>***** Suggested function Call: design_protein_from_CATH *****</p> <p>Arguments:</p> <pre>{ "name": "mainly_alpha_protein_1", "CATH_ANNOTATION": "1", "length": 100 }</pre> <p>*****</p>
<p>&gt;&gt;&gt;&gt;&gt;&gt;&gt; EXECUTING FUNCTION design_protein_from_CATH...</p> <p>We use this CATH to generate protein with length 100: 1</p> <p>assistant (to chat_manager):</p> <p>***** Response from calling function "design_protein_from_CATH" *****</p> <pre>( './code_protein/mainly_alpha_protein_1.pdb', 'SMKKIEDYIREKCLKALGLSDEEIEERVKQLMEGIK NPKKFEKELQKRNDRESLLIFKEAYALYEASKDKEKGKKLINKVQSERDKWETEQAEEAARAA AAA')</pre> <p>*****</p>



```
assistant (to chat_manager):

**** Suggested function Call: design_protein_from_CATH ****
Arguments:
{
"name": "mainly_alpha_protein_2",
"CATH_ANNOTATION": "1",
"length": 100
}
*****

>>>>>>> EXECUTING FUNCTION design_protein_from_CATH...
We use this CATH to generate protein with length 100: 1
assistant (to chat_manager):

**** Response from calling function "design_protein_from_CATH" ****
('./code_protein/mainly_alpha_protein_2.pdb', 'MSKKEIEELKKKLDEIVETLKEYARQGDDACKK
AADLIEEVKKALEEGNPEKYSQLKKKLTDAINKAIEEYRKRFEAEGKPPEEAQKVIDKLLKILD
EITN')
*****

assistant (to chat_manager):

**** Suggested function Call: design_protein_from_CATH ****
Arguments:
{
"name": "mainly_beta_protein_1",
"CATH_ANNOTATION": "2",
"length": 100
} *****

>>>>>>> EXECUTING FUNCTION design_protein_from_CATH...
We use this CATH to generate protein with length 100: 2
assistant (to chat_manager):

**** Response from calling function "design_protein_from_CATH" ****
('./code_protein/mainly_beta_protein_1.pdb', 'TTVTVTPPVADADGNEHSTVTA YGNKVTTITITCPS
NCTVTETVDGVAKTLGTVSGNQTTITETRTIAPDEVVTRTYTCTPNASATSSKTQVTIKGSQPA
P')
*****

assistant (to chat_manager):

**** Suggested function Call: design_protein_from_CATH ****
Arguments:
{
"name": "mainly_beta_protein_2",
"CATH_ANNOTATION": "2",
"length": 100
}
*****

>>>>>>> EXECUTING FUNCTION design_protein_from_CATH...
```

```
We use this CATH to generate protein with length 100: 2
assistant (to chat_manager):

**** Response from calling function "design_protein_from_CATH" ****
('./code_protein/mainly_beta_protein_2.pdb', 'SLKAKNLEEMIKEAEKLGYSRDEVEKIINEIRDK-
FKKLGVKISEKTLAYIAYLRLGKIDWDKIKKVKKATPADFRVSEEDLKKPEIQKILEKIKKEIN')
*****

assistant (to chat_manager):

**** Suggested function Call: design_protein_from_CATH ****
Arguments:

{
"name": "alpha_beta_protein_1",
"CATH_ANNOTATION": "3",
"length": 100
}
*****

>>>>>>> EXECUTING FUNCTION design_protein_from_CATH...
We use this CATH to generate protein with length 100: 3
assistant (to chat_manager):

**** Response from calling function "design_protein_from_CATH" ****
('./code_protein/alpha_beta_protein_1.pdb', 'APTVKTFEDTINGQKVTVTVTASPGGKITIKTSPGYG
DEVAKAFIEELKKQNVLESYKVESAPGKETTISDVKVKSGATVTFYVINNGKKKKEYSVTVD
A')
*****

assistant (to chat_manager):

**** Suggested function Call: design_protein_from_CATH ****
Arguments:

{
"name": "alpha_beta_protein_2",
"CATH_ANNOTATION": "3",
"length": 100
}

>>>>>>> EXECUTING FUNCTION design_protein_from_CATH...
We use this CATH to generate protein with length 100: 3
assistant (to chat_manager):

**** Response from calling function "design_protein_from_CATH" ****
('./code_protein/alpha_beta_protein_2.pdb', 'MELKVTEKKGKGDYKVKVIELNTPDKRYIIIESDAS
RESLIKAAEALLQGKEVEPTPVNEKNVVLFEDEDVKTSIERSKKLKSDNPEENIKKALEYLLK
') *****

>>>>>>> EXECUTING FUNCTION analyze_protein_structure...
Analyzing the secondary strucute of this protein:
./code_protein/mainly_alpha_protein_1.pdb
The protein analyzed has 100 residues.
assistant (to chat_manager):
```

```
***** Response from calling function "analyze_protein_structure" *****
{
  "H": 89.0,
  "B": 0.0,
  "E": 0.0,
  "G": 0.0,
  "I": 0.0,
  "T": 4.0,
  "S": 1.0,
  "P": 0.0,
  "-": 6.0
}
*****
```

```
assistant (to chat_manager):

***** Suggested function Call: analyze_protein_structure *****
Arguments:

{
"protein_structure": "./code_protein/mainly_alpha_protein_2.pdb"
} *****
```

```
>>>>>>> EXECUTING FUNCTION analyze_protein_structure...
Analyzing the secondary strucute of this protein:
./code_protein/mainly_alpha_protein_2.pdb
The protein analyzed has 100 residues.
assistant (to chat_manager):

***** Response from calling function "analyze_protein_structure" *****
{
  "H": 89.0,
  "B": 0.0,
  "E": 0.0,
  "G": 0.0,
  "I": 0.0,
  "T": 5.0,
  "S": 0.0,
  "P": 0.0,
  "-": 6.0
}
*****
```

```
assistant (to chat_manager):

***** Suggested function Call: analyze_protein_structure *****
Arguments:

{
"protein_structure": "./code_protein/mainly_beta_protein_1.pdb"
} *****
```

```
~~~~~ EXECUTING FUNCTION analyze_protein_structure...
Analyzing the secondary strucute of this protein:
./code_protein/mainly_beta_protein_1.pdb
The protein analyzed has 100 residues.
```

assistant (to chat\_manager):

\*\*\*\*\* Response from calling function "analyze\_protein\_structure" \*\*\*\*\*

```
{
  "H": 0.0,
  "B": 0.0,
  "E": 64.0,
  "G": 0.0,
  "I": 0.0,
  "T": 10.0,
  "S": 6.0,
  "P": 0.0,
  "-": 20.0
}
*****
```

assistant (to chat\_manager):

\*\*\*\*\* Suggested function Call: analyze\_protein\_structure \*\*\*\*\*

Arguments:

```
{
  "protein_structure": "./code_protein/mainly_beta_protein_2.pdb"
}
*****
```

>>>>>>> EXECUTING FUNCTION analyze\_protein\_structure...  
Analyzing the secondary structure of this protein: ./code\_protein/mainly\_beta\_protein\_2.pdb  
The protein analyzed has 100 residues.

assistant (to chat\_manager):

\*\*\*\*\* Response from calling function "analyze\_protein\_structure" \*\*\*\*\*

```
{
  "H": 57.99999999999999,
  "B": 0.0,
  "E": 8.0,
  "G": 6.0,
  "I": 0.0,
  "T": 8.0,
  "S": 4.0,
  "P": 3.0,
  "-": 13.0
}
*****
```

assistant (to chat\_manager):

\*\*\*\*\* Suggested function Call: analyze\_protein\_structure \*\*\*\*\*

Arguments:

```
{
  "protein_structure": "./code_protein/alpha_beta_protein_1.pdb"
}
*****
```

>>>>>>> EXECUTING FUNCTION analyze\_protein\_structure...

Analyzing the secondary strucute of this protein:

./code\_protein/alpha\_beta\_protein\_1.pdb

The protein analyzed has 100 residues.

assistant (to chat\_manager):

\*\*\*\*\* Response from calling function "analyze\_protein\_structure" \*\*\*\*\*

```
{
  "H": 15.0,
  "B": 0.0,
  "E": 59.0,
  "G": 3.0,
  "I": 0.0,
  "T": 12.0,
  "S": 1.0,
  "P": 0.0,
  "-": 10.0
}
```

\*\*\*\*\*

assistant (to chat\_manager):

\*\*\*\*\* Suggested function Call: analyze\_protein\_structure \*\*\*\*\*

Arguments:

```
{
  "protein_structure": "./code_protein/alpha_beta_protein_2.pdb"
}
```

\*\*\*\*\*

>>>>>>> EXECUTING FUNCTION analyze\_protein\_structure...

Analyzing the secondary strucute of this protein:

./code\_protein/alpha\_beta\_protein\_2.pdb

The protein analyzed has 100 residues.

assistant (to chat\_manager):

\*\*\*\*\* Response from calling function "analyze\_protein\_structure" \*\*\*\*\* {

```
  "H": 35.0,
  "B": 0.0,
  "E": 28.999999999999996,
  "G": 0.0,
  "I": 0.0,
  "T": 3.0,
  "S": 12.0,
  "P": 3.0,
  "-": 18.0
}
```

\*\*\*\*\*

Planner (to chat\_manager):

\*\*\*\*\* Suggested function Call: calucalte\_force\_energy\_from\_seq \*\*\*\*\*

Arguments:

```
{
```

```
"sequence" : "SMKKIEDYIREKLLKALGLSDEEIEERVVKQLMEGIKNPKKFEKELQKRNDRESLLI  
FKEAYALYEASKDKEKGKLLINKVQSERDKWETEQAEEAARAAAAA"  
}  
*****
```

```
>>>>>>> EXECUTING FUNCTION calucalte_force_energy_from_seq...  
CalculateForceEnergy;SMKKIEDYIREKLLKALGLSDEEIEERVVKQLMEGIKNPKKFEKELQKRND  
RESLLIFKEAYALYEASKDKEKGKLLINKVQSERDKWETEQAEEAARAAAAA;  
assistant (to chat_manager):  
  
***** Response from calling function "calucalte_force_energy_from_seq" *****  
"0.381,0.444"  
*****
```

```
assistant (to chat_manager):  
  
***** Suggested function Call: calucalte_force_energy_from_seq *****  
Arguments:  
  
{  
"sequence" : "MSKKEIEELKKKLDEIVETLKEYARQGDDACKKAADLIEEVKKALEEGNPEKY  
SQLKKKLTDAINKAIEEYRKRFEAEGKPEEAQKVIDKLLKILDEITN"  
}  
*****
```

```
>>>>>>> EXECUTING FUNCTION calucalte_force_energy_from_seq...  
CalculateForceEnergy;MSKKEIEELKKKLDEIVETLKEYARQGDDACKKAADLIEEVKKALEEG  
NPEKYSQLKKKLTDAINKAIEEYRKRFEAEGKPEEAQKVIDKLLKILDEITN;  
assistant (to chat_manager):  
  
***** Response from calling function "calucalte_force_energy_from_seq" *****  
"0.376,0.536"  
*****
```

```
assistant (to chat_manager):  
  
***** Suggested function Call: calucalte_force_energy_from_seq *****  
Arguments:  
  
{  
"sequence" : "TTVTVTPPVADADGNEHSTVTAYGNKVTITITCPSNCTVTETVDGVAKTLGTVS  
GNQTITETRTIAPDEVVTRTYTCTPNASATSSKTQVTIKGSQPAP"  
}  
***** >>>>>>>  
EXECUTING FUNCTION calucalte_force_energy_from_seq...  
CalculateForceEnergy<TTVTVTPPVADADGNEHSTVTAYGNKVTITITCPSNCTVTETVDGVAK  
TLGTVSGNQITITETRTIAPDEVVTRTYTCTPNASATSSKTQVTIKGSQPAP>  
assistant (to chat_manager):  
  
***** Response from calling function "calucalte_force_energy_from_seq" *****  
"0.462,0.533"  
*****
```

```
assistant (to chat_manager):  
  
***** Suggested function Call: calucalte_force_energy_from_seq *****
```

Arguments: <pre>{ "sequence": "SLKAKNLEEMIKEAEKLGYSRDEVEKIINEIRDKFKKLGVKISEKTLAYIAYLRL LGVKIDWDKIKKVKKATPADFRVSEEDLKKPEIQKILEKIKKEIN" } *****</pre>
>>>>>>> EXECUTING FUNCTION calucalte_force_energy_from_seq... CalculateForceEnergy<SLKAKNLEEMIKEAEKLGYSRDEVEKIINEIRDKFKKLGVKISEKTLAY IAYLRLLGVKIDWDKIKKVKKATPADFRVSEEDLKKPEIQKILEKIKKEIN> assistant (to chat_manager):  ***** Response from calling function "calucalte_force_energy_from_seq" ***** "0.371,0.548" *****
Planner (to chat_manager):  ***** Suggested function Call: calucalte_force_energy_from_seq ***** Arguments: <pre>{ "sequence": "APTVKTFEDTINGQKVTVTVTASPGGKITIKTSPGYGDEVAKAFIEELKKQNVLE SYKVESAPGKETTISDVKVKSGATVTFYVINNGKKGKEYSVTVDA" } *****</pre>
>>>>>>> EXECUTING FUNCTION calucalte_force_energy_from_seq... CalculateForceEnergy<APTVKTFEDTINGQKVTVTVTASPGGKITIKTSPGYGDEVAKAFIEELK KQNVLESYKVESAPGKETTISDVKVKSGATVTFYVINNGKKGKEYSVTVDA> assistant (to chat_manager):  ***** Response from calling function "calucalte_force_energy_from_seq" ***** "0.424,0.535" *****
assistant (to chat_manager):  ***** Suggested function Call: calucalte_force_energy_from_seq ***** Arguments: <pre>{ "sequence": "MELKVTEKKGKGDYKVKVIELNTPDKRYIIIESDASRESLIKAAEALLQGKEVEP TPVNEKNVLFEDVDKTSIERSKCLFKSDNPEENIKKALEYLLK" } *****</pre>
>>>>>>> EXECUTING FUNCTION calucalte_force_energy_from_seq... CalculateForceEnergy<MELKVTEKKGKGDYKVKVIELNTPDKRYIIIESDASRESLIKAAEALLQ GKEVEPTPVNEKNVLFEDVDKTSIERSKCLFKSDNPEENIKKALEYLLK> assistant (to chat_manager):  ***** Response from calling function "calucalte_force_energy_from_seq" ***** "0.376,0.543" *****
Planner (to chat_manager):

```
**** Suggested function Call: calc_protein_ANM ****
Arguments:
{
  "protein_structure": "./code_protein/mainly_alpha_protein_1.pdb",
  "n_modes": 10
}
*****

>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
assistant (to chat_manager):

**** Response from calling function "calc_protein_ANM" ****
[
  0.2329,
  0.4901,
  0.9331,
  1.3741,
  1.7347,
  2.1598,
  2.3686,
  2.6359,
  2.8555,
  3.0364
]
*****

assistant (to chat_manager):

**** Suggested function Call: calc_protein_ANM ****
Arguments:
{
  "protein_structure": "./code_protein/mainly_alpha_protein_2.pdb",
  "n_modes": 10
}
*****

>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
computing ANM for protein structure: ./code_protein/mainly_alpha_protein_2.p
assistant (to chat_manager):

**** Response from calling function "calc_protein_ANM" ****
[
  1.6126,
  2.0783,
  2.3073,
  2.4565,
  3.399,
  3.475,
  4.1377,
  4.7104,
  4.8864,
  5.2187
]
```



```
*****  
Planner (to chat_manager):  
**** Suggested function Call: calc_protein_ANM ****  
Arguments:  
{  
"protein_structure": "/code_protein/mainly_beta_protein_1.pdb",  
"n_modes": 10  
}  
*****  
>>>>>>> EXECUTING FUNCTION calc_protein_ANM...  
computing ANM for protein structure: ./code_protein/mainly_beta_protein_1.pdb  
assistant (to chat_manager):  
**** Response from calling function "calc_protein_ANM" ****  
[ 1.2806,  
  1.5057,  
  1.9846,  
  2.1025,  
  2.4723,  
  2.702,  
  2.9931,  
  3.1498,  
  3.4432,  
  4.1685  
]  
*****  
assistant (to chat_manager):  
**** Suggested function Call: calc_protein_ANM ****  
Arguments:  
{  
"protein_structure": "/code_protein/mainly_beta_protein_2.pdb",  
"n_modes": 10  
}  
*****  
>>>>>>> EXECUTING FUNCTION calc_protein_ANM...  
computing ANM for protein structure: ./code_protein/mainly_beta_protein_2.pdb  
assistant (to chat_manager):  
**** Response from calling function "calc_protein_ANM" ****  
[  
  2.8864,  
  4.3752,  
  4.5928,  
  4.8295,  
  5.0854,  
  5.5618,  
  5.8646,  
  6.007,  
]
```

```
6.3847,
7.1246
]
*****

Planner (to chat_manager):

**** Suggested function Call: calc_protein_ANM ****
Arguments:

{
"protein_structure": "/code_protein/alpha_beta_protein_1.pdb",
"n_modes": 10
}
*****

>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
computing ANM for protein structure: ./code_protein/alpha_beta_protein_1.pdb
assistant (to chat_manager):

**** Response from calling function "calc_protein_ANM" ****
[
2.4383,
2.5651,
3.3175,
3.8231,
3.9673,
4.2655,
4.6393,
5.1509,
5.6023,
5.9555
]
*****

assistant (to chat_manager):

**** Suggested function Call: calc_protein_ANM ****
Arguments:

{
"protein_structure": "/code_protein/alpha_beta_protein_2.pdb",
"n_modes": 10
}
*****

>>>>>>> EXECUTING FUNCTION calc_protein_ANM...
computing ANM for protein structure: ./code_protein/alpha_beta_protein_2.pdb
assistant (to chat_manager):

**** Response from calling function "calc_protein_ANM" ****
[
2.8756,
3.8895,
4.0594,
4.2831,
```

4.5542,  
5.171,  
5.3661,  
5.4312,  
6.1964,  
6.3066

] \*\*\*\*\*

Critic (to chat\_manager):

The protein designs and their analyses are as follows:

For the mainly alpha proteins:

1. Protein mainly\_alpha\_protein\_1: - Sequence: SMKKIEDYIREKLLKALGLSDEEIEERVKQLMEGI  
KNPKKFEKELQKRNDRESLLIFKEAYALYEASKDKEKGGKLINKVQSERDKWETEQAEEAARA  
AAAA

- Secondary structure: 'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'I': 0.0, 'T': 4.0, 'S': 1.0, 'P': 0.0, '-': 6.0  
- Unfolding energy and maximum force: 0.381,0.444  
- First 10 frequencies: [0.2329, 0.4901, 0.9331, 1.3741, 1.7347, 2.1598, 2.3686, 2.6359, 2.8555, 3.0364]

2. Protein mainly\_alpha\_protein\_2:

- Sequence: MSKKEIEELKKKLDEIVETLKEYARQGDDACKKAADLIEEVKKALEEGNPEKYSQ  
LKKKLTDAINKAIEEYRKRFEAEGKPEEAQKVIDKLLKILDEITN

- Secondary structure: 'H': 89.0, 'B': 0.0, 'E': 0.0, 'G': 0.0, 'I': 0.0, 'T': 5.0, 'S': 0.0, 'P': 0.0, '-': 6.0  
- Unfolding energy and maximum force: 0.376,0.536  
- First 10 frequencies: [1.6126, 2.0783, 2.3073, 2.4565, 3.399, 3.475, 4.1377, 4.7104, 4.8864, 5.2187]

For the mainly beta proteins:

1. Protein mainly\_beta\_protein\_1:

- Sequence:  
- Secondary structure: 'H': 0.0, 'B': 0.0, 'E': 64.0, 'G': 0.0, 'I': 0.0, 'T': 10.0, 'S': 6.0, 'P': 0.0, '-':  
20.0  
- Unfolding energy and maximum force: 0.462,0.533  
- First 10 frequencies: [1.2806, 1.5057, 1.9846, 2.1025, 2.4723, 2.702, 2.9931, 3.1498, 3.4432, 4.1685]

2. Protein mainly\_beta\_protein\_2: - Sequence: SLKAKNLEEMIKEAEKLGYSRDEVEKIINEIRDKF  
KKGKISEKTLAYIAYLRLLGVKIDWDKIKKVKKATPADFRVSEEDLKKPEIQKILEKIKKEI  
N

- Secondary structure: 'H': 57.99999999999999, 'B': 0.0, 'E': 8.0, 'G': 6.0, 'I': 0.0, 'T': 8.0, 'S': 4.0,  
'P': 3.0, '-': 13.0  
- Unfolding energy and maximum force: 0.371,0.548  
- First 10 frequencies: [2.8864, 4.3752, 4.5928, 4.8295, 5.0854, 5.5618, 5.8646, 6.007, 6.3847, 7.1246]

For the alpha beta proteins:

1. Protein alpha\_beta\_protein\_1:

- Sequence: APTVKTFEDTINGQKVTVTVTASPGGKITIKTSPGYGDEVAKAFIEELKKQNVLES  
YKVESAPGKETTISDVKVKSGATVTFYVINNGKKGKEYSVTVDA  
- Secondary structure: 'H': 15.0, 'B': 0.0, 'E': 59.0, 'G': 3.0, 'I': 0.0, 'T': 12.0, 'S': 1.0, 'P': 0.0, '-':  
10.0  
- Unfolding energy and maximum force: 0.424,0.535

```

- First 10 frequencies: [2.4383, 2.5651, 3.3175, 3.8231, 3.9673, 4.2655, 4.6393, 5.1509, 5.6023, 5.9555]

2. Protein alpha_beta_protein_2:
- Sequence: MELKVTEKKGKGDYKVKVIELNTPDKRYIIIESDASRESLIKAAEALLQGKEVEPT
PVNEKNVVLFEDEDVKTSIERSKKLFKSDNPEENIKKALEYLLK
- Secondary structure: 'H': 35.0, 'B': 0.0, 'E': 28.999999999999996, 'G': 0.0, 'I': 0.0, 'T': 3.0, 'S':
12.0, 'P': 3.0, '-': 18.0
- Unfolding energy and maximum force: 0.376,0.543
- First 10 frequencies: [2.8756, 3.8895, 4.0594, 4.2831, 4.5542, 5.171, 5.3661, 5.4312, 6.1964, 6.3066]

```

user\_proxy (to chat\_manager):

Could you save the results in a CSV file named "protein\_analysis.csv," where each row corresponds to a protein, and each column represents a specific property? Include the AA sequence in the results and use suitable identifiers for the columns.

Planner (to chat\_manager):

```

***** Suggested function Call: save_to_csv_file *****
Arguments:
{
  "input_JSON_dictionary" :      "{ \"Protein Name\":      [ \"mainly_alpha_protein_1\",
  \"mainly_alpha_protein_2\",      \"mainly_beta_protein_1\",      \"mainly_beta_protein_2\",
  \"alpha_beta_protein_1\", \"alpha_beta_protein_2\" ], \"AA Sequence\": [ \"SMKKIEDYIREKL
KALGLSDEIEERVKQLMEGIKNPKKFEKELQKRNDRESLLIFKEAYALYEASKDKEKGKKLI
NKVQSERDKWETEQAEEAARAAAAA\", \"MSKKEIEELKKKLDEIVETLKEYARQGDDACKK
AADLIEEVKKALEEGNPEKYSQLKKKLTDAINKAIEEYRKRFEAEGKPEEAQKVIDKLLKILD
EITN\", \"TTVTVTPPVADADGNEHSTVTAYGNKVTITITCPSNCTVTETVDGVAKTTLGTVSG
NQTITERTIAPDEVVTRTYTCTPNASATSSKTQVTVIKGSQPAP\", \"LKAKNLEEMIKEAEK
LGYSRDEVEKIINEIRDKFKKLGVKISEKTLAYIAYLRLLGVKIDWDKIKVKKATPADFRVS
EEDLKKPEIQKILEKIKKEIN\", \"APT VKTFEDTINGQKVTVTVTASPGGKITIKTSPGYGDEV
AKAFIEELKKQNVLESYKVESAPGKETTISDVKVKS GATVTFYVINNGKKGKEYSVTVDA\",
\"MELKVTEKKGKGDYKVKVIELNTPDKRYIIIESDASRESLIKAAEALLQGKEVEPTPVNEKN
VVLFEDEDVKTSIERSKKLFKSDNPEENIKKALEYLLK\" ], \"Secondary Structure\": [{ \"H\":
89.0, \"B\": 0.0, \"E\": 0.0, \"G\": 0.0, \"I\": 0.0, \"T\": 4.0, \"S\": 1.0, \"P\": 0.0, \"-\": 6.0},
{ \"H\": 89.0, \"B\": 0.0, \"E\": 0.0, \"G\": 0.0, \"I\": 0.0, \"T\": 5.0, \"S\": 0.0, \"P\": 0.0,
\"-\": 6.0}, { \"H\": 0.0, \"B\": 0.0, \"E\": 64.0, \"G\": 0.0, \"I\": 0.0, \"T\": 10.0, \"S\": 6.0,
\"P\": 0.0, \"-\": 20.0}, { \"H\": 58.0, \"B\": 0.0, \"E\": 8.0, \"G\": 6.0, \"I\": 0.0, \"T\": 8.0,
\"S\": 4.0, \"P\": 3.0, \"-\": 13.0}, { \"H\": 15.0, \"B\": 0.0, \"E\": 59.0, \"G\": 3.0, \"I\": 0.0,
\"T\": 12.0, \"S\": 1.0, \"P\": 0.0, \"-\": 10.0}, { \"H\": 35.0, \"B\": 0.0, \"E\": 29.0, \"G\":
0.0, \"I\": 0.0, \"T\": 3.0, \"S\": 12.0, \"P\": 3.0, \"-\": 18.0}], \"Unfolding Energy\": [0.381,
0.376, 0.462, 0.371, 0.424, 0.376], \"Max Force\": [0.444, 0.536, 0.533, 0.548, 0.535, 0.543], \"First
10 Frequencies\": [[0.2329, 0.4901, 0.9331, 1.3741, 1.7347, 2.1598, 2.3686, 2.6359, 2.8555, 3.0364],
[1.6126, 2.0783, 2.3073, 2.4565, 3.399, 3.475, 4.1377, 4.7104, 4.8864, 5.2187], [1.2806, 1.5057,
1.9846, 2.1025, 2.4723, 2.702, 2.9931, 3.1498, 3.4432, 4.1685], [2.8864, 4.3752, 4.5928, 4.8295,
5.0854, 5.5618, 5.8646, 6.007, 6.3847, 7.1246], [2.4383, 2.5651, 3.3175, 3.8231, 3.9673, 4.2655,
4.6393, 5.1509, 5.6023, 5.9555], [2.8756, 3.8895, 4.0594, 4.2831, 4.5542, 5.171, 5.3661, 5.4312,
6.1964, 6.3066]]}",
  "output_csv_name" : "protein_analysis.csv"
}
*****

```

```
>>>>>>> EXECUTING FUNCTION save_to_csv_file...
```

<p>the results have been saved to csv file: protein_analysis.csv assistant (to chat_manager):</p> <p>**** Response from calling function "save_to_csv_file" **** None *****</p>
<p>user_proxy (to chat_manager):</p> <p>Based on the results of this example, can you check if the protein generator (Chroma) has been successful in creating proteins with desired structure?</p>
<p>Critic (to chat_manager):</p> <p>Based on our results it seems that the Chroma protein generator was successful in creating proteins with desired secondary structure content that generally correspond to their CATH annotations: - Both of the mainly alpha proteins showed a very high percentage (around 89%) of alpha-helix secondary structure ('H'), which is in line with their designation. - The mainly beta proteins showed higher percentages of extended strand/beta-sheet secondary structure ('E'). Though, the percentages varied quite a bit (64% for mainly_beta_protein_1 and only 8% for mainly_beta_protein_2), which could be due to the complex nature of beta-structures. - As for the alpha beta proteins, both showed both helix ('H') and sheet ('E') characters, thus also meeting the requirement of having mixed content.</p> <p>However, some proteins, most notably mainly_beta_protein_2 and mainly_alpha_protein_1, also showed significant portions of residues with no classified secondary structures ('-'). This can be potentially due to incomplete folding or limitations of the protein generation or structure prediction process.</p> <p>To be certain about the tertiary structure and its classifications (Alpha, Beta, Alpha/Beta), we would have to look at the actual 3D protein models - the secondary structure does not provide full insight into the 3D structure, which is what the CATH classification is based on. For this, more extensive protein structure analysis would be needed. But overall, the generator seems to show promising results.</p>
<p>Provide feedback to chat_manager. Press enter to skip and use auto-reply, or type 'exit' to end the conversation: Thanks. user_proxy (to chat_manager):</p> <p>Thanks.</p>
<p>Planner (to chat_manager):</p> <p>You're welcome! If you have any other questions or need further assistance, feel free to ask.</p>
<p>assistant (to chat_manager):</p> <p>TERMINATE</p>

Table B7: The constants used to normalize the force and energy values in the mechanical unfolding data set to train and validate ForceGPT.

	<b>Force (Kcal/(mol<math>\times</math>Å))</b>	<b>Energy (Kcal/mol)</b>
<i>Normalized constants</i>	898.4	97656.6

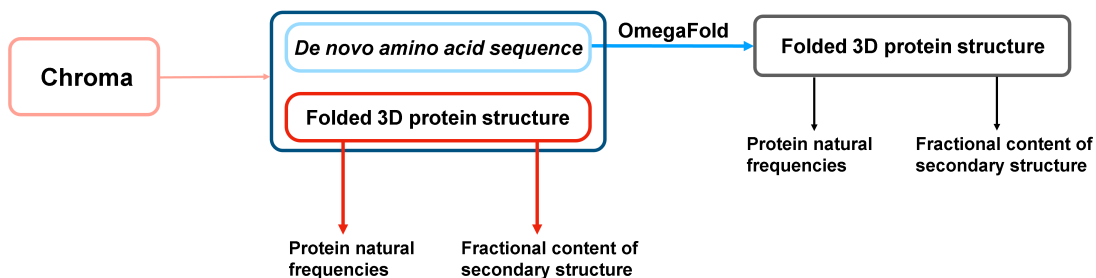


Figure B1: **Overview of the multi-agent work to solve the complex task posed in the first experiment, Section 4.1.** First, the multi-agent uses Chroma to generate *de novo* protein sequences and then computes natural frequencies and secondary structures content for the generated structures. Next, from *de novo* AA sequences, the model finds the 3D folded structures using OmegaFold and finally computes the frequencies and secondary structure content for the protein structures. The results obtained from the Chroma and OmegaFold 3D protein structures are compared in Figure 3.

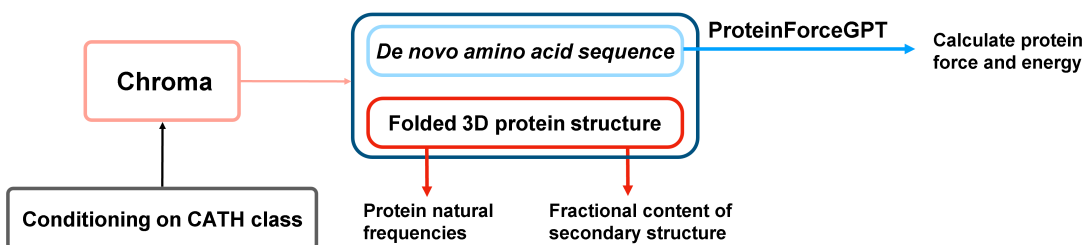


Figure B2: **Overview of the multi-agent work to solve the complex task posed in the second experiment, Section 4.2.** First the multi-agent uses Chroma to generate *de novo* protein sequences and structures conditioned on the input CATH class. Then using the generated protein structures, the natural frequencies and secondary structures content are computed. Next, the force (maximum force along the unfolding force-extension curve) and energy (the area under the force-extension curve) are computed from *de novo* AA sequences using ProteinForceGPT.

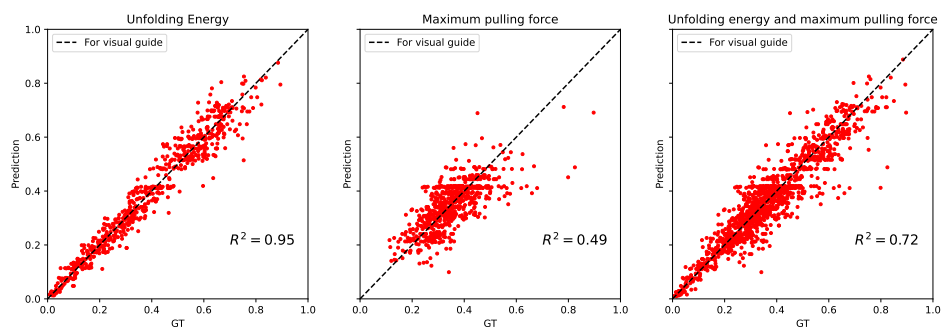


Figure B3: **Assessment of the ForceGPT model on the test set.** We test the model with mechanical unfolding responses from proteins in the test set, for normalized unfolding energy (task: CalculateEnergy<seq>), maximum pulling force (task: CalculateForce<seq>), and Unfolding energy and maximum pulling force (task: CalculateEnergyForce<seq>).