A Framework for Scientific Discovery through Video Games

ACM Books

Editor in Chief
M. Tamer Özsu, *University of Waterloo*

ACM Books is a new series of high-quality books for the computer science community, published by ACM in collaboration with Morgan & Claypool Publishers. ACM Books publications are widely distributed in both print and digital formats through booksellers and to libraries (and library consortia) and individual ACM members via the ACM Digital Library platform.

This book by Seth Cooper is a revised version of the dissertation that won the 2011 ACM Doctoral Dissertation Award. Other books in the series include those listed below.

A Framework for Scientific Discovery through Video Games
Seth Cooper, *University of Washington*
2014

Trust Extension as a Mechanism for Secure Code Execution on Commodity Computers
Bryan Jeffrey Parno, *Microsoft Research*
2014

Embracing Interference in Wireless Systems
Shyamnath Gollakota, *University of Washington*
2014

A Framework for Scientific Discovery through Video Games

Seth Cooper
University of Washington

ACM Books #3

A Framework for Scientific Discovery through Video Games

Seth Cooper

books.acm.org
www.morganclaypool.com

ISBN: 978-1-62705-504-8 print
ISBN: 978-1-62705-505-5 ebook
ISBN: 978-1-62705-506-2 ePub
Series ISSN: (to come)

DOIs: 10.1145/2625848 Book
10.1145/2625848.2625849 Preface
10.1145/2625848.2625850 Chapter 1
10.1145/2625848.2625851 Chapter 2
10.1145/2625848.2625852 Chapter 3
10.1145/2625848.2625853 Chapter 4
10.1145/2625848.2625854 Chapter 5
10.1145/2625848.2625855 Chapter 6
10.1145/2625848.2625856 Chapter 7
10.1145/2625848.2625857 Bibliography

A publication in the ACM Books series, #3
Editor in Chief: M. Tamer Özsu, *University of Waterloo*

First Edition

10 9 8 7 6 5 4 3 2 1

To my parents, Harris and Beth Cooper

Contents

Preface **xiii**

Chapter 1 **Introduction 1**

1.1 Motivation **1**
1.2 Problem Statement **2**
1.2.1 Game Design Problem **2**
1.2.2 Biochemistry Discovery Problem **3**
1.3 Outline **4**

Chapter 2 **Related Literature 5**

2.1 Volunteer Computing and Human Computation **5**
2.2 Serious Games and Gamification **7**
2.3 Computational Biochemistry **8**

Chapter 3 **Framework 9**

3.1 Introduction **9**
3.2 Biochemistry Background **11**
3.3 Framework Description **12**
3.3.1 Architecture **12**
3.3.2 Coevolution Strategy **13**
3.3.3 Categorization as a game **15**
3.4 Game Design Challenges **16**
3.4.1 Visualizations **16**
3.4.2 Interactions **19**
3.4.3 Scoring **23**
3.4.4 Introductory Levels **23**
3.5 Rewards and Social Interaction **26**
3.5.1 Rewards and ranking types **30**
3.6 Conclusion **31**

Chapter 4 Protein Structure Prediction 33
4.1 Introduction 33
4.2 Quest to the Natives 37
4.3 CASP8 Experiments 40
4.4 Evaluation 42
4.5 Rebuild and Refine Comparison 44
4.5.1 First strand swap example 56
4.5.2 Player contribution and expertise 57
4.6 Alignment Tool and CASP9 58
4.7 Solution of Crystal Structure 62
4.8 Conclusion 63

Chapter 5 Protein Design 67
5.1 Introduction 67
5.2 Framework Extension 68
5.2.1 Foldit 68
5.2.2 Iteration Strategy 69
5.3 Science Transfer 70
5.3.1 Visualizations 70
5.3.2 Tools 72
5.3.3 Conditions 73
5.4 Introductory Levels 73
5.5 Examples 75
5.5.1 Fibronectin 75
5.5.2 Diels-Alder 78
5.6 Conclusion 80

Chapter 6 Protein Structure Refinement Algorithms 83
6.1 Introduction 83
6.2 Related Work 87
6.3 Overview 88
6.3.1 Cookbook 88
6.4 CASP9 Analysis 90
6.4.1 Recipe Sharing 90
6.4.2 Inheritance Relationships 93
6.4.3 Ratings 95
6.5 Script Recipe Adoption Analysis 97
6.6 Algorithm Categories 98

6.7 Context Dependence 99
6.8 Recipe Evolution 101
6.9 Performance Comparison 104
6.10 Conclusion 106

Chapter 7 Conclusion 109

7.1 Contributions 109
7.2 Future Work 109

Bibliography 111

Author's Biography 119

Preface

When we first set out to create Foldit over six years ago, it wasn't clear that a game-based approach to scientific discovery would work. So we planned from the start for the game to be continually adapting and changing, in order to keep improving based on the lessons we'd learn. It took several years of design, development, and continued iteration from a team of computer scientists and biochemists until the game was at a point where we made our first exciting discovery. The nature of the challenging problems we were facing required this time and refinement to solve.

We are now seeing a number of other games that allow players to contribute to scientific research. Much of this growth has been in fields related to biology and biochemistry: EteRNA for designing RNA shapes, EyeWire for mapping neurons, and Phylo for aligning genetic sequences. Each of these games has had exciting scientific results produced by gameplay. Games are being applied in other areas as well, such as in the Algoraph suite of games for solving graph theory problems. I have been involved in the development of two more science games: Nanocrafter, which aims to push the frontiers of DNA-based synthetic biology, and Flow Jam, which allows players to help formally verify software.

The Foldit community has continued to grow and adapt to the new scientific challenges that we have posed. We have continued to look further into the design of synthetic proteins and their applications for health and understanding proteins. The Foldit team is continually grateful to the player community for their creativity, problem solving, and enthusiasm. Their brainpower and fresh insights have been critical to the successes of the project; without the players, there would be no game.

Scientific discovery games are an exciting part of the growing effort towards engaging the public in science. I look forward to new ways for anyone with an interest and a passion to contribute to scientific discovery.

Acknowledgments

There are many people to thank for helping and supporting me during the creation of this book.

- my advisor, Zoran Popović, for his guidance, support, and the opportunity to work on such a great video game as part of my research;
- David Baker for his expertise and time for helping make this a successful project;
- my committee members, David Salesin, John Zahorjan, and Ethan Merrit, for their valuable feedback throughout the project;
- the Foldit development team—Adrien Treuille and Janos Barbero were instrumental at the very beginning of the project, starting off on the considerable task of creating Foldit;
- the members of the Baker lab for lending their considerable biochemical expertise to the project. Firas Khatib contributed excellent biochemical insight and interaction with the Foldit community, and Andrew Leaver-Fay was always available and helpful in understanding the Rosetta codebase;
- the community of Foldit players for their enormous contribution to this work, for always surprising me with their ingenuity, and for their patience with the development of the game; and
- my parents, Harris and Beth Cooper, my sister, Emily Cooper, and my wife, Fayette Shaw, for all their help and support throughout school.

My work was supported by the National Science Foundation, DARPA, the Howard Hughes Medical Institute, NVIDIA, Electronic Arts, Sony, Microsoft, and Adobe.

Seth Cooper
July 2014

1 Introduction

1.1 Motivation

Despite the massive amounts of computational power available, many difficult scientific problems still remain computationally intractable. Fortunately, people possess great skill in problem solving and creativity, and individual problem solving skills can be augmented by working together in groups. However, only a small population of people are involved in scientific inquiry and advancing science.

The following questions arise: (1) How much more scientific advancement would be possible if more people were involved? (2) Can we integrate what people and computers, respectively, do well? We would like to maximize the effectiveness of this human-computer symbiosis, to find places were computational power is most useful and where human ability can best be applied. People and computers are often good at solving different types of problems; for example, a person would likely translate a passage of text more naturally, and a computer would likely be able to numerically optimize a function faster. We would like to be able to combine the best abilities of each in order to solve challenging problems that neither could alone.

The goal of this book is to determine if it is possible to design the coevolution of human-computer symbiosis to solve currently open problems in science. Two particular areas where humans can excel are spatial reasoning and creativity. People are able to reason spatially by forming mental models of objects, their environment, and the spatial relationships between them [Byrne and Johnson-Laird 1989, Tversky 1993].

People enjoy expressing their creativity, and many successful video games require players to think about objects in space and their spatial relationships to each other. Tetris[1] is a popular example of this. There are many physical puzzles that rely on spatial reasoning as well, such as Rubik's Cube[2] and many sliding block puzzles. Recently, there has been a rise in popularity of video games whose explicit purpose is to help

1. http://www.tetris.com/; last retrieved May 2014.

2. http://www.rubiks.com/; last retrieved May 2014.

people train their cognitive skills, such as spatial reasoning. Brain Age[3] and Big Brain Academy[4] are examples of this.

A scientific field that naturally requires spatial reasoning and creativity for problem solving is biochemistry. Many problems in biochemistry are fundamentally spatial structural problems, particularly when dealing with protein structures. Proteins are important to biochemistry and our understanding of life itself, because they are indispensable to living systems and perform many important tasks in the cell, including structural, transport, defensive, and catalytic roles. The way proteins achieve their function is due to their shapes and how they interact with other molecules. They involve the relationships between physical objects in three-dimensional space; a protein's structure determines its function [Zhang and Kim 2003].

To explore the potential of this human-computer framework for solving scientific problems, we have developed Foldit,[5] an online video game that casts protein structure manipulation as a puzzle solving competition. The game tries to predict naturally occurring protein structures and to design novel proteins not previously seen in nature. In order to achieve this goal, the game gives players the ability to manipulate and optimize protein structures while competing and collaborating with other players to discover the best structures. Foldit's YouTube channel can be found at http://www.youtube.com/user/uwfoldit; http://www.youtube.com/watch?v=lGYJyur4FUA gives a good introduction to the game.

1.2 Problem Statement

1.2.1 Game Design Problem

Designing a game for scientific discovery presents many distinct challenges. One of the primary purposes of using a game is to maximize the engagement and retention of the players. However, it is not enough to simply make the game as fun as possible; this goal must also be balanced with the need for relevant scientific outcomes. For most games, the designer is free to make decisions based only on what will make the game fun. In a scientific discovery game, the tension between the freedom to design for engagement and the realism of the scientific constraints is a key challenge. Thus, an important question is, how can we design a game that is both engaging and produces useful results? We would also like to know what kinds of problems would lend themselves

3. http://brainage.com/; last retrieved May 2014.

4. http://bigbrainacademy.com/; last retrieved May 2014.

5. http://fold.it/

to such contributions by non-scientists, and how can we identify these problems and map them onto a game.

We presume that the game players begin without any knowledge of the scientific field the game is based in. Given this, we would like for the players to gain the domain knowledge necessary to make a contribution to a challenging scientific problem quickly. This is not necessarily general expertise in the subject area or formal scientific expertise. Players may develop their own specialized form of expertise unique to the problem presentation within the game. How can the game best support the training of players to the point where they can make a contribution, and integrate players into the scientific process? We would like to use structures from games to teach players, and keep players interested and involved long-term. We would also like to spread the expertise gained by experienced players and to help new players learn from it.

The game's architecture should support the coevolution of both the players and the game itself. In this way, as the players adapt to the game by gaining experience in how the game works and solving the problems presented, the game can also adapt to how the players best use it to become a better tool. How can we allow for this coevolution of the game and the player base?

1.2.2 Biochemistry Discovery Problem

Predicting protein structures computationally is a central goal for computational biochemists because so much can be understood about a protein's function once its structure is known, and because it is so challenging to observe a protein's structure directly. Proteins—chains of smaller molecules called amino acids—are central to biochemistry because they are the primary chemical for almost all cellular processes. Understanding a protein's structure is necessary to understand its functions, because a protein's shape determines how it will interact with other molecules. Thus, an important problem in biochemistry is the *protein structure prediction* problem: given the sequence of amimo acids that make up a protein, what is its structure? It is possible to experimentally determine a protein's structure through methods such as X-Ray Crystallography and Nuclear Magnetic Resonance spectroscopy. Experimental methods, however, can be costly, time consuming, and difficult. This makes computational methods that can accurately predict a protein's structure an attractive solution. However, computational methods are often intractable; the vast number of possible shapes a protein can take make it difficult to find the correct structure. The spatial nature of this problem makes it a good candidate for the application of human spatial reasoning.

A related problem is that of protein *design*: given a desired function for a protein, what is the amino acid sequence that, when folded, will carry it out? In this case, computational methods are even more attractive. Synthesizing proteins to test every

design would be prohibitively expensive, while computational methods can allow us to filter out designs that are not likely to work. Protein design has implications for drug design, in inhibitors and vaccines, for biofuel design, in enzymes, and for other areas. Human creativity can be applied to help create novel proteins that did not exist before.

1.3 Outline

In this book we will show the effectiveness of the game-based framework as an approach to scientific discovery. Chapter 2 discusses the literature related to this book. Chapter 3 gives an overview of the game-based framework used in this research, describing the dual goals of engagement and scientific relevance, and the coevolution approach we take. A discussion of using this framework for problem solving as applied to protein structure prediction is given in Chapter 4, and we show that players can predict the unknown structures of naturally occurring proteins, even where all previous methods have failed. Further discussion of applying this framework to leverage player creativity for protein design is given in Chapter 5, and we show that players can become an integral part of the design of novel and effective proteins. Chapter 6 describes an approach to allowing players to codify and automate their strategies, and we show that players can socially develop highly effective algorithms. Finally, Chapter 7 provides a summary and discusses possible future directions for research.

2 Related Literature

This book is related to several bodies of existing literature, including volunteer computing, human computation, serious games, computational biochemistry, and visualization and interaction.

2.1 Volunteer Computing and Human Computation

Volunteer computing is a method by which volunteers are able to donate their computer's spare time and space to various projects. The volunteer computing model has risen in popularity recently, and has allowed scientists access to unprecedented amounts of computational power. One of the oldest and largest volunteer computing projects is SETI@home[1] [Sullivan III et al. 1997]. This project uses a screensaver to analyze radio telescope data. There is an open source platform for developing volunteer computing projects, the Berkeley Open Infrastructure for Network Computing (BOINC),[2] which allows users to manage and share their computer's resources between the many projects using the platform [Anderson 2004]. BOINC projects have a variety of goals, from climate prediction [Stainforth et al. 2005] to searching for pulsars [Knispel et al. 2010].

By using the volunteer computing model, projects not only gain access to massive computation, but also allows the public to make contributions to science. However, with this model, their contributions are mostly passive—they don't even have to be at their computer. This work aims to use not only the power of networks of computers, but also that of networks of humans, and allow people to make active contributions to science.

There has been work recently on leveraging a human workforce for computational tasks that computers are not yet able to perform satisfactorily. A more general field

1. http://setiathome.ssl.berkeley.edu/; last retrieved May 2014.

2. http://boinc.berkeley.edu/; last retrieved May 2014.

of "human computation" or "distributed thinking" is emerging. On a smaller scale, augmenting automated heuristics with interactive human input can help to solve basic spatial problems [Anderson et al. 2000, Lesh et al. 2005]. On a larger scale, general tasks desired for humans to perform are posted online, and users can determine which tasks they would like to perform. Amazon's Mechanical Turk[3] is one example of such a system, where users are actually paid to perform tasks. Example tasks include translation of text, rating search results, and determining the tone of an article. Bossa is an open source system for managing similar user tasks.[4]

In this context, there has been much interest in using games as a means of motivating people to perform tasks that are currently difficult for computers. One particularly active area is in computer vision and image recognition. Humans are particularly adept at reading words in images and determining the objects in a scene, when compared with current computational methods. The difference in ability is strong enough that vision-based tests are often used as a proof of humanity with CAPTCHAs [Ahn et al. 2003].

Games such as the ESP game [von Ahn and Dabbish 2004], Peekaboom [von Ahn et al. 2006], and Google Image Labeler[5] use human image-recognition ability to produce labeled images from gameplay. Image recognition has also been used for finding particular features of interest in scientific data, such as looking for signs of interstellar dust [Westphal et al. 2010], measuring and aligning features on a planet's surface,[6] and classifying galaxy shapes.[7] These projects have been successful in motivating players to sift through large image sets, which would otherwise be a mundane task.

Some games have approached other types of problems. Pebble It[8] is a game which studies human solutions to the graph pebbling problem, with the goal of developing better algorithms to solve it [Cusack et al. 2006]. Outside of games, some work has examined how to fit human problem solving into various optimization problems [Anderson et al. 2000, Lesh et al. 2005]. This work is different because it leverages a deeper human problem solving ability to create interesting scientific results.

3. http://www.mturk.com/; last retrieved May 2014.

4. http://boinc.berkeley.edu/trac/wiki/BossaIntro; last retrieved May 2014.

5. http://images.google.com/imagelabeler/; last retrieved May 2014.

6. NASA Be A Martian, http://beamartian.jpl.nasa.gov/; last retrieved May 2014.

7. Galaxy Zoo, http://www.galaxyzoo.org/; last retrieved May 2014.

8. http://pebbleit.hope.edu/; last retrieved May 2014.

2.2 Serious Games and Gamification

Recently, a field known as "serious games" has been identified. The most general definition is any game that has a purpose beyond simply entertaining the player; however, it often connotes games whose purpose is training or education. The line between game and simulation or application is also not always well defined. A game-based approach is appealing because games are meant to be engaging and motivating. Furthermore, other fields are taking advantage of the fact that gaming has pushed the limits of interactive simulation and authoring technology, such as 3D engines [Susi et al. 2007]. Some of the wide-ranging applications of serious games are firefighter training [Backlund et al. 2007], raising awareness of social issues,[9] and military recruitment.[10] In this book, the main goal is to generate useful scientific discoveries; however, other aspects of game design, such as the requirement that the game be fun, contribute to achieving this goal, as the results rely on players playing the game.

One particular subgenre of serious games is games for health. Playing games has, in some settings, been shown to be beneficial to the player's health. Games have been shown to be useful for rehabilitation, development, and therapy, and even for distracting patients from pain or bad habits [Adriaenssens et al. 1988, Griffiths 2005]. Nintendo's Wii Fit package[11] is intended to help players improve their personal fitness.

Many games emphasize social interactions as well. Massively multiplayer online games (MMOs), like World of Warcraft[12] and Second Life,[13] often host persistent virtual worlds where players can customize avatars, socialize, and work together with other players. Diverse niche MMOs exist, targeting teens or people interested in racing, allowing people with similar interests to interact [Zenke 2008].

Similarly, Alternate Reality Games (ARGs) engage large groups of people to participating in narratives in the real world [Martin et al. 2006]. Often, multiple forms of technology will be used to coordinate the players, who will be working together towards a common goal. I Love Bees, a popular ARG, had players work together to find payphones and answer prerecorded questions [Terdiman 2004].

9. Darfur is Dying, http://www.darfurisdying.com; last retrieved May 2014.

10. America's Army, http://www.americasarmy.com/; last retrieved May 2014.

11. http://www.nintendo.com/wiifit/; last retrieved May 2014.

12. http://us.battle.net/wow/en/; last retrieved May 2014.

13. http://secondlife.com/; last retrieved May 2014.

2.3 Computational Biochemistry

In the field of biochemistry, many computational methods have been used to study protein folding, predict protein structures, and design new proteins. The most closely related to our work is Rosetta. Rosetta combines structural energy minimization with a Monte Carlo search algorithm to predict native protein structures [Rohl et al. 2004]. Foldit uses the Rosetta software for its energy function, as well as many of the algorithms and functionality for energy minimization and protein manipulation. In order to access massive amounts of computation for searching a protein's large structural space, the volunteer computing project Rosetta@home[14] runs Rosetta's algorithms on volunteer's computers.

Another volunteer computing project, Folding@home,[15] aims to simulate the process of protein folding. The Folding@home project has also been ported to the PS3, and thus has access to powerful hardware and gives gamers the opportunity to help science. Folding@home's approach is based on simulating the molecular dynamics of protein folding [Pande et al. 2003]. One difference to note is that Rosetta and Foldit only attempt to determine the final structures, while Folding@home simulates the folding process. This makes Rosetta and Foldit more amenable to problems like protein design, where one is interested in the final folded structure, while Folding@home's approach is more useful for studying topics like protein misfolding.

Many visualizations for biological molecules have been developed to aid biochemists. Popular visualizations include Corey-Pauling-Koltun (CPK) [Corey and Pauling 1953], which displays the chemical properties of individual atoms, and cartoon or ribbon, which shows a more abstract, stylized version of the backbone's secondary structures [Natarajan et al. 2008]. PyMOL is a popular viewer that gives access to many of these visualizations.[16] These visualizations are tuned for scientists, and may not be appropriate for novices.

One possible approach to manipulating three-dimensional objects is to use widgets. Widgets associate behavior with geometry in the scene, so users can interact directly with the environment [Conner et al. 1992]. The Sculpt system allows users to interact with and guide a protein as it folds in the presence of minimization of a physically plausible energy [Surles et al. 1994]. While Sculpt has some features in common with Foldit, it is a single-user application, while Foldit is a multiplayer game, with many new features.

14. http://boinc.bakerlab.org/rosetta/; last retrieved May 2014.

15. http://folding.stanford.edu/; last retrieved May 2014.

16. The PyMOL Molecular Graphics System, http://www.pymol.org/; last retrieved May 2014

3 Framework

3.1 Introduction

This chapter introduces a general framework for *scientific discovery games*. We present guidelines for mapping a scientific problem into a game, and address the often conflicting goals of engagement and scientific relevance. The driving example is *Foldit*, a game for scientific discovery in biochemistry. We describe the architecture of the game. The architecture is flexible and able to coevolve, along with the game's players, to improve as a tool. We discuss the teaching and reward structures in the game, intended to appeal to a wide variety of players, regardless of biochemistry background.

A scientific discovery game translates a class of computationally difficult scientific problems into puzzles, and provides a game-like mechanism for non-scientist players to help solve these problems. Many traditional aspects of game design apply to scientific discovery games, including the design of introductory levels to draw newcomers and explain game mechanics, the use of a client-server architecture for competition and collaboration, and the requirement that the game be fun. However, unlike games whose goal is entertainment or education, scientific discovery games introduce a unique challenge: *enabling non-scientist natural problem solvers to advance a specific scientific domain*. This challenge influences all aspects of the game design. First, visualization and graphics need to promote human ability to see complex solutions and convey accurate scientific information while remaining accessible to beginners. Second, interaction design must optimize for natural interactions suitable for the human exploration process, while still respecting scientific constraints. Finally, the scoring mechanism needs to be informative enough to promote multiple human strategies, while remaining true to the latest models of the underlying scientific phenomenon. Perhaps the most distinguishing feature and the greatest difficulty of design for this type of game is that the solution to the scientific problem, and thus the solution to the corresponding puzzles, is unknown. Since we do not know the solution *a priori*, we cannot design the game with specific solutions in mind.

To explore this space, we focused on human ability to reason about 3D structures and on the biochemistry domain, where many problems tend to be structural. We developed *Foldit*, a biochemical discovery game. In this chapter, we discuss the framework for Foldit's design, with emphasis on the game's initial focus on protein structure

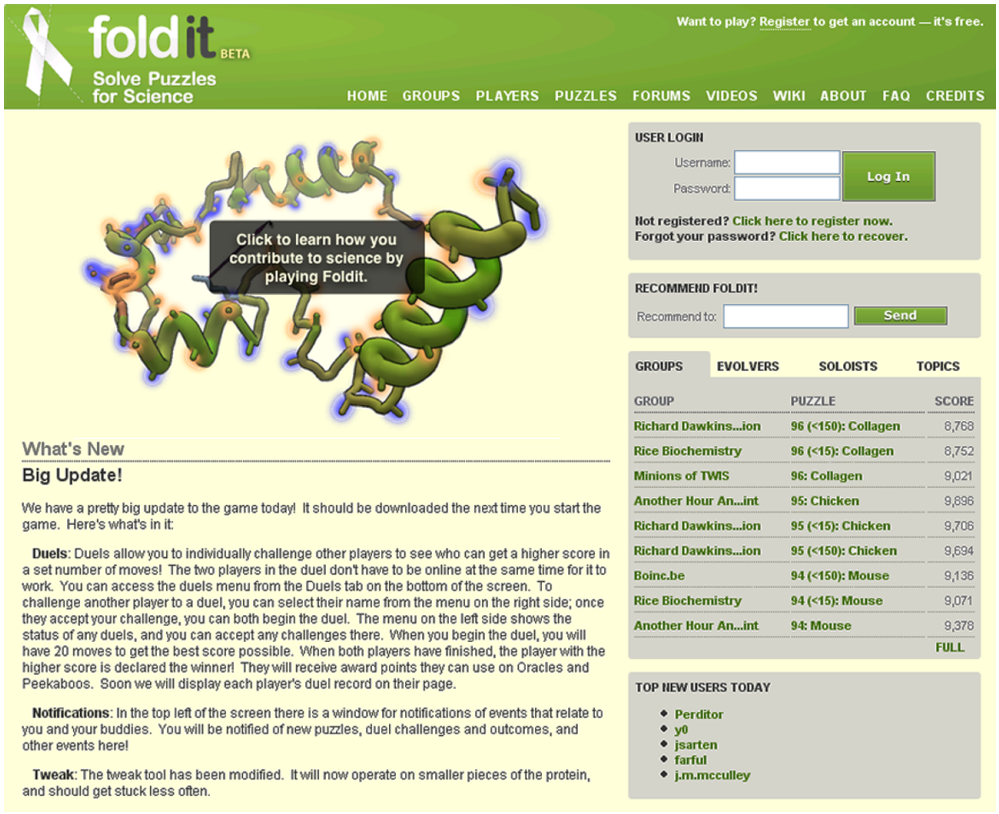


Figure 3.1 **Foldit webpage.** The front page shows recent news about the game, the top players and groups for the current puzzles, and allows the player to log in.

prediction—determining a protein's shape given its sequence of constituent amino acids. Protein structure prediction involves finding favorable interactions that form when the protein's chemical groups come into contact—essentially a 3D jigsaw puzzle. We believe that humans' innate spatial reasoning ability makes it possible for non-scientists to make useful contributions to this problem. We leverage scientists' knowledge to shape the rules of the game, thus enabling a much larger pool of non-scientists to make discoveries within this framework.

The webpage for Foldit is located at http://fold.it. The front page is shown in Figure 3.1. Foldit was publicly released in May 2008. During the first two years following release, we ran roughly 600 structure prediction puzzles and had over 57,000 players from a wide variety of backgrounds participate.

The rest of this chapter describes our experience designing Foldit, with a special emphasis on the unique challenges posed by making biochemistry problems accessible to anyone. The creation of Foldit was a challenging and multidisciplinary project, drawing together computer science, art, game design and biochemistry. Moreover, we did not know ahead of time which parts of the problem players would be best at solving, or which in-game manipulation tools they would use most effectively. The only way to find out was to have people play Foldit. In order to deal with these and other uncertainties, we took an iterative approach both before and after releasing the game to the public. We have continually evolved the gameplay in response to massive gameplay traces, player feedback and scientists' analysis, and continue even now with this iterative process as we add features and expand the set of biochemical problems to which the Foldit community can contribute.

Games are often designed with an iterative approach, which involves designing, testing, and evaluating repeatedly until the player's experience meets some criteria [Fullerton 2008]. For most games, the main criterion for the player's experience is simply to have fun. Player feedback and playtesting are an integral part of the process, and there are a number of methods of gathering and incorporating this information from players [Ambinder 2009]. We have also continued the design process after the game's release, to incorporate data gathered from the players in a continual process of evolutionary redesigning [Kennerly 2003]. Our work differs from the standard iterative approach in that the game design space is constrained to conform with existing physical models, we include the input of scientists in the evaluation of the game, and we include the long-term coevolution of the players and game in the design.

3.2 Biochemistry Background

Here we provide some background on biochemistry and proteins that will be used throughout the rest of this work.

DNA, a cellular chemical perhaps more widely recognized than proteins, derives its entire purpose in encoding protein sequences. Proteins are coded for by DNA, and are created in the cell as a long chain of *amino acids*. A protein's amino acid sequence is known as its *primary structure*. There are twenty different types of amino acids. Regardless of type, some of atoms making up the amino acid will be the same; these are connected together and form the protein's *backbone*. However, the remaining atoms are different for each type; these extend outward from the backbone and are called *sidechains*. The atoms that make up the sidechains divide the amino acids into two main groups: *hydrophobic*, which prefer to be buried on the interior away from water;

and *hydrophilic*, which prefer to be exposed on the exterior near water. These preferences impact how the protein folds. As the amino acids are connected together, the protein begins to fold up; after the amino acids join together, they are often called *residues*. Local characteristics of the fold are referred to as *secondary structure*. These include: *helices*, which are tightly coiled; *sheets*, which are extended straight; and *loops*, which are everything else. The positions of the atoms making up a folded protein is its *tertiary structure*; the tertiary structure taken in nature is a *native structure*. The native structure is one that is lowest in free energy—it has the most favorable set of chemical interactions. It is well known that sequence determines structure [Anfinsen 1973]. In this book, the term sequence will refer to a protein's primary structure, and structure will refer to its tertiary structure, unless otherwise specified.

3.3 Framework Description

3.3.1 Architecture

Here we give an overview of the architecture of Foldit, which can be seen at a high level in Figure 3.2. Foldit uses a client-server architecture. Players must create an account and download the game in order to play. The game then communicates with a central server to send information about the local player and get information about other players.

Scientists post problems to the server; in the case of Foldit, these are protein structures for which the players are meant to find the native structures. An initial protein structure is associated with metadata such as a title and description, and parameterization such as which energy function terms to use. We call these *puzzles*, and they are posted on the server for a fixed amount of time (usually a week). While a puzzle is active, players can download it and interactively reshape the protein to try to achieve the best score. This often requires significant changes to the puzzle structures, which are given in various partially-folded states, and in some cases need to be completely refolded from a straight line. Players' structures, or *solutions*, are reported back to the server, and players are ranked against other players who are playing the same puzzle. Players can form groups with which to share their solutions through the server, allowing them to work together to find even better solutions than they could working alone. When one player *shares* a solution by uploading it to the server, other players in the same group are able to see it and download it. The social aspect of the game is supported by in-game chat, a website with forums, and a player-created wiki. At the close of a puzzle, the solution data is aggregated, and presented to the scientists for analysis.

The game is designed to be flexible, and the client allows automatic updating so that we can continually evolve the gameplay. The puzzle posting cycle and automatic

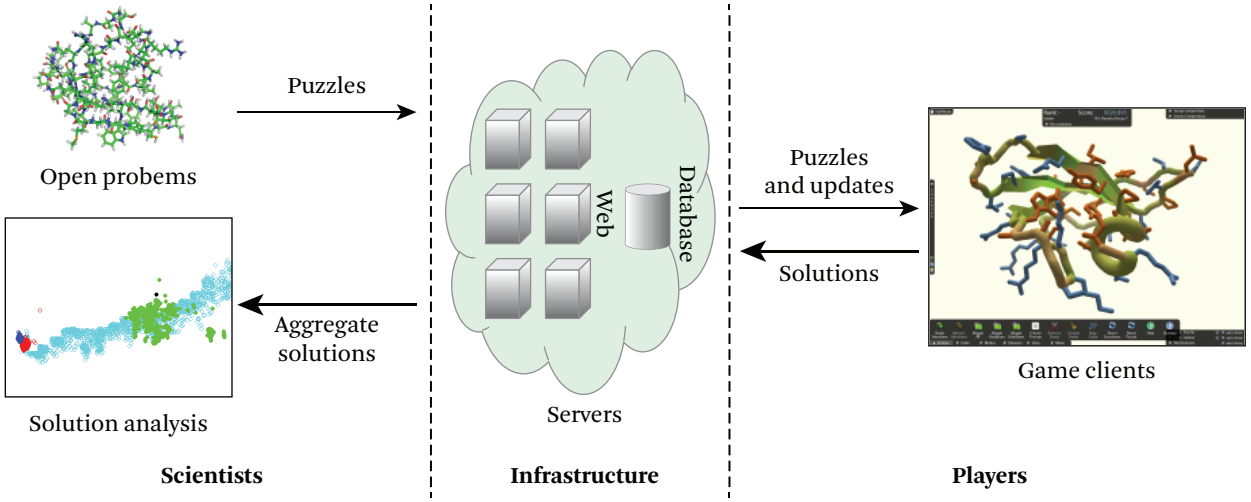


Figure 3.2 **Overview of architecture for scientific discovery games.** The biochemistry team provides structure prediction and design problems for the server. These problems become puzzles and are sent to each player's client. Players collaborate and compete to solve these problems and upload their solutions to the server, where they are aggregated and sent back to the biochemistry team for analysis. This analysis can then be used to improve the design of the game and puzzles. (Figure from Cooper et al. [2010b])

updates allow us to respond to not only player feedback, but also to scientists' analysis, as we introduce and refine gameplay elements.

Foldit is built on top of the Rosetta molecular modeling suite which has proven useful at a wide variety of protein modeling tasks [Rohl et al. 2004, Bradley et al. 2005, Qian et al. 2007, Kuhlman et al. 2003]. The suite contains an energy function which captures the interaction energies between protein elements, as well as a set of structural optimization subroutines. For protein structure prediction, structures closer to the native structure will have a lower energy than structures further away from it. Foldit uses this state-of-the-art energy function to compute player's scores, and also takes advantage of the optimization routines Rosetta makes available.

3.3.2 Coevolution Strategy

In order to arrive at the current state of Foldit, we took an coevolution approach to the game's design. Given the complexity of this undertaking, we realized that it was unlikely that all our initial decisions would be the best. There are three major groups relevant to our approach: (1) the scientists whose problems the game is meant to help solve; (2) the

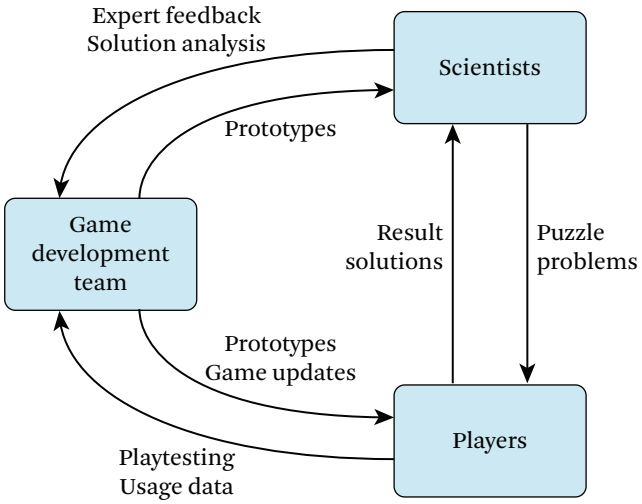


Figure 3.3 **Overview of the interactions between the three iterative design groups.** (Figure from Cooper et al. [2010b])

players; and (3) the game development team. The development team must incorporate feedback from the players to make sure the game is understandable and fun, and from the scientists to make sure that the results produced will be useful to them. An overview of the interactions between these three groups is given in Figure 3.3.

During the game's initial development, the development team and scientists must work together closely to determine an initial direction. This involves defining what problems to approach, what the fundamental gameplay mechanics needed are, and what the desired results are. Once possible games have been prototyped, player feedback can begin to be incorporated. Early playtesting helps to uncover what elements of the problem are fun and which can be most confusing and difficult to understand. This can help to both focus the gameplay and narrow the scope of the game to where players will most likely be able to contribute.

After making the game available to the public, a large amount of data and feedback can become available to help improve the game. As in a traditional game, data on gameplay can be gathered from players for an objective analysis of what players are doing, and feedback from the player community is extremely useful in determining new features. However, in a scientific discovery game, as scientists post puzzles and player solutions are analyzed, this analysis must then be incorporated in the design of the game, progressing towards ever better results.

Following this pattern, Foldit has evolved significantly since its initial release. A timeline of significant events in the evolution of the game are given in Figure 3.4.

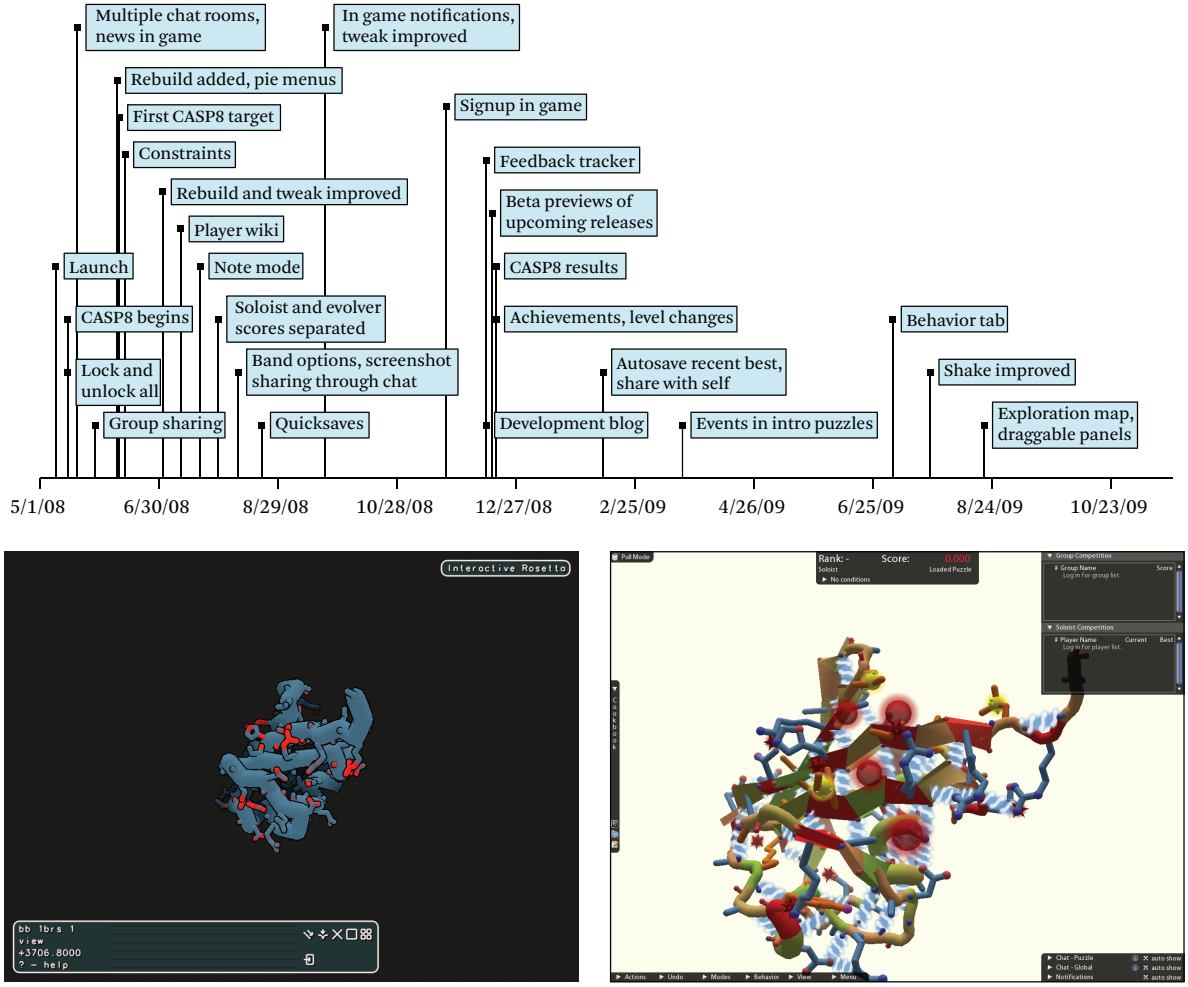


Figure 3.4 **Selected events from the game's evolution over time.** The timeline is shown on the top. Screenshots are included from before release (bottom left) and the current version (bottom right). (Figure from Cooper et al. [2010b])

3.3.3 Categorization as a Game

Although it relies heavily on simulation and visualization, Foldit can be classified as a game, as it possesses the qualities of a game set forth by Schell [Morgan Kaufmann]. Here we list the qualities and how Foldit embodies each.

1. Games are entered willfully: We do not require players to play Foldit.
2. Games have goals: Foldit's goal is to find the best scoring structure.

3. Games have conflict: Foldit has conflict with both the protein itself, trying to find a better score, and with other players, trying to outrank them.
4. Games have rules: The rules of Foldit are given by the scoring function, available moves, global point structure, and so forth.
5. Games can be won and lost: Each puzzle has a ranking, which could be broken down into "winners" and "losers".
6. Games are interactive: Foldit allows players to interactively reshape a protein and gives them immediate feedback.
7. Games have challenge: Similar to conflict, Foldit's challenge arises from achieving higher scores and competing with other players.
8. Games create their own internal value: Foldit's global points have value for ranking within the game.
9. Games engage players: Foldit keeps players engaged in manipulating protein structures.
10. Games are closed, formal systems: Foldit's rules define the pieces of the system and how they work together.

3.4 Game Design Challenges

3.4.1 Visualizations

While a user is playing Foldit, several visualizations are available. These help the player determine when they are or aren't doing well, and show which areas of the protein they could improve and what is wrong with them, so the player can think about how to fix any problems. Figure 3.5 shows a screenshot of the game's main screen. We intend for the game to look like a game and not necessarily a scientific illustration. While scientific illustration techniques are useful for scientists, they may not be for our purposes, and may in fact be intimidating for non-scientists. Many of the visualizations have options, or can be turned off and on by the player. They include the following.

The protein. The protein itself is rendered in a cartoon-like style. This style is abstract and does not show the exact positions of all the atoms in the protein. The helices, sheets, and loops appear differently along the backbone, and sidechains are rendered very simply. The protein is colored by the score of each residue.

Clashes. These are red flashing spiky balls. They appear where two atoms are too close together, which will severely reduce the score.

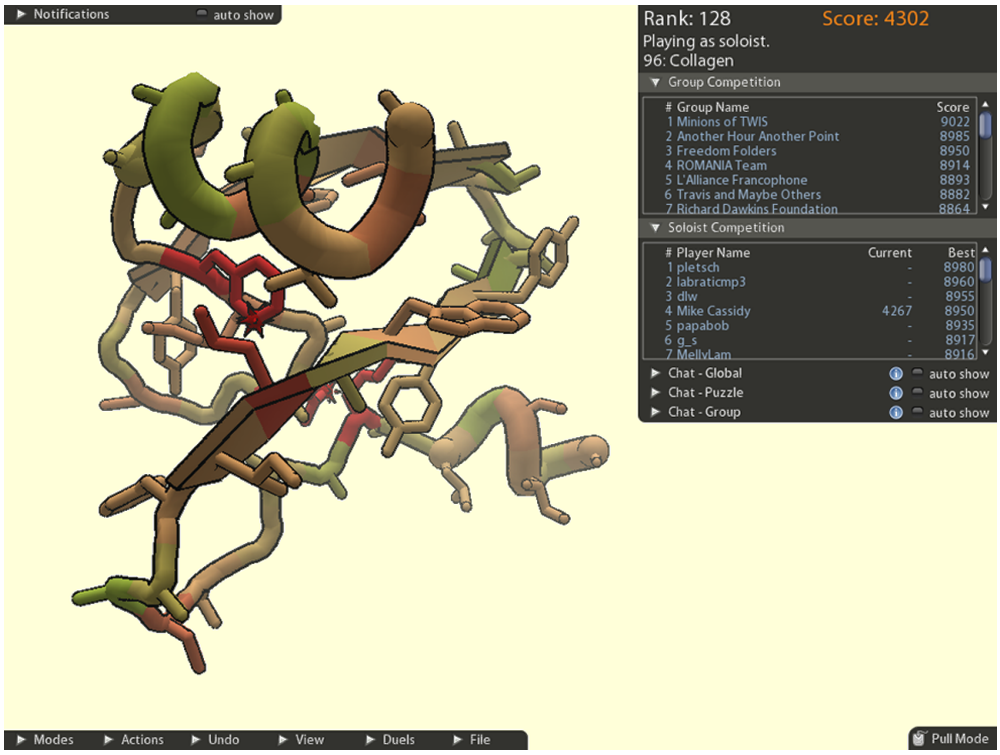


Figure 3.5 **Foldit's main game screen.** The puzzle *Collagen* is shown. The protein is in the center; some clashes are visible. The panel in the top right shows the player's rank and score, leaderboards for groups and individuals in the current puzzle, and chat. Menus and information are in the other corners of the screen.

Hydrogen bonds. These will appear as blue and white ladders where hydrogen bonds have been formed. These bonds improve the score and help hold the protein together.

Hydrophobic sidechains. Hydrophobic and hydrophilic sidechains are shown in different colors. Burying hydrophobic sidechains in the core of the protein can improve the score.

Voids. These yellow spheres will appear where there is empty space in the protein. Filling in the space can improve the score.

The visualizations in a scientific discovery game must achieve several purposes in order to allow players to apply their problem-solving skills. They must *reflect and*

illuminate the natural rules of the system, in a way that makes state of the system evident to the player and directs them to where their contribution will be most useful. At the same time, the visualizations need to *manage and hide the complexity of the system*, so that players are not immediately overwhelmed by information. They must be *approachable by players* who have no knowledge of the scientific problem at hand. Thus, they should look inviting and fun, and not bring back memories of high school textbooks. Ideally, they should be customizable, because as with other aspects of the game, it is not clear from the outset what the best visualization will be, and different players may have different preferences.

In order to make the visualization of Foldit *reflect and illuminate* the fundamental properties of proteins, we worked with scientists to distill simple rules upon which to base them. The first rule is to avoid clashes. Clashes occur when atoms are unrealistically close to each other, causing a large repulsive force. These can be prevented by keeping the atoms from overlapping, and are represented by spiky, rotating spheres that float between the overlapping atoms. The second rule is to fill voids, or empty spaces in the protein. Packing the protein tightly will remove voids. Voids are represented as bubble-like objects that pop when they come in contact with the protein. Clashes and voids appear red, as natural proteins should not generally have any. The third rule is to bury exposed hydrophobics. Hydrophobics are sidechains whose chemical properties are such that it is favorable for them to be on the interior of the protein. Exposed hydrophobics are represented as small, pulsing spheres that move along their sidechain. These are drawn in yellow, rather than red, because natural proteins may have some exposed hydrophobics. The fourth rule is to maintain and create hydrogen bonds, which form between particular pairs of atoms and hold the protein together. Hydrogen bonds appear as undulating bars between the bonded atoms, and are drawn in blue, because they are good.

Due to the spatial nature of the problem, the visualization of the protein closely matches the actual geometry of the protein. To make the overall structure stand out, sheets, helices, and loops are stylized, similar to many scientific visualization tools.[1] Sheets appear with a zig-zag pattern that will form hydrogen bonds when properly fit together. Color also plays a large role in the visualization of the protein. The backbone color reflects the score of the protein in a particular region—going from red in poor scoring regions to green in good scoring regions—so players can see where they can gain the most points. The sidechains are colored by hydrophobicity, so players can quickly see if they are extending them in the preferred direction. By coloring backbone and

1. The PyMOL Molecular Graphics System, http://www.pymol.org/; last retrieved May 2014.

sidechain independently we can display more information while not introducing too much visual clutter.

Foldit takes a number of approaches to *manage and hide* the complexity of huge networks of interconnected atoms that make up a protein. Many unimportant details are hidden. Hydrogen atoms, which are plentiful on the protein but do not add a lot of structural information, are hidden. However, hidden information will reappear if it becomes important to the player: sidechains can disappear entirely to make the overall structure of the protein's backbone clearer, but will reappear if they are causing a problem, such as if they are involved in a clash. Many actual clashes themselves are also hidden: only the worst clash is shown on a per-amino acid basis. This prevents the player from being overwhelmed by the number of clashes if the protein is compressed too tightly.

To make the game *approachable*, we gave the protein itself a bright, cartoonish look. Many pieces of the visualizations move playfully around the protein. There are a wide variety of visualization options available in the game as well, such as alternative colorings and geometries for the protein. These can be accessed through a special menu option that is turned off by default. This approach allows more advanced players the ability to customize their view in the view options menu, but keeps things simple for newcomers.

Visualizations such as voids and exposed hydrophobics can be computationally expensive to compute. To keep the game interactive, we compute such visualizations in a separate thread, which will update the visualization after a delay.

3.4.2 Interactions

Foldit also provides several different methods of acting on the protein. Figure 3.6 is a screenshot of user interaction. Players need actions that allow them to manipulate the proteins in a way which will bring them closer to their goal. They should be intuitive and useful; we have tried to structure our input around the idea of *touchability*, or direct manipulation. Whenever possible, we have tried to make operations act directly on the protein itself. Some of the possible actions the user can perform are:

Pulling. This is intended to be the primary method of interaction. When the user clicks and drags on part of the protein, a purple arrow extends from the protein to the location of the user's mouse cursor. The protein will then try to move to stay under the mouse, while still satisfying some energetic preferences.

Bands. Purple bands can be placed by the user to attach one residue to another, or a residue to a point in space. When the user performs another operation on the protein, bands will pull on the attached residues. This can allow the user to keep

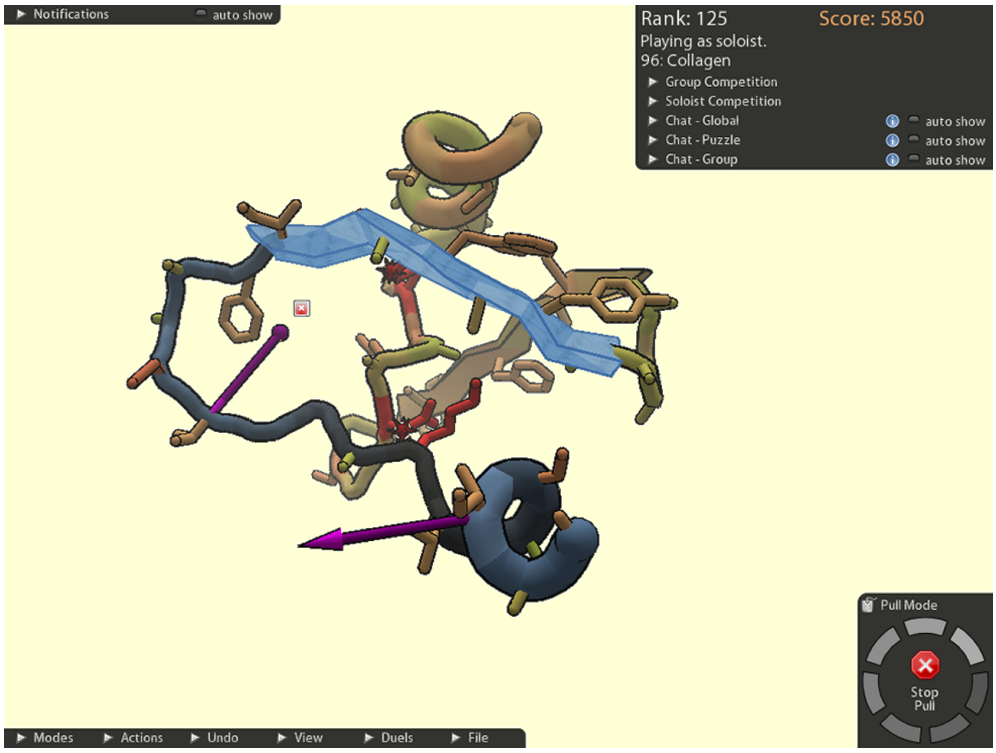


Figure 3.6 **Foldit's main game screen during interaction.** The puzzle *Collagen* is shown. The player is acting on the protein. The icy blue sheet is locked and the purple cylinder with the round end is a band. The purple cylinder with the pointed end shows where the user is pulling on the backbone. The dark blue part of the backbone is affected by the pull.

parts of the protein in place, pull to specific points in space, or pull in multiple places at once.

Locks. Locks prevent the protein from being affected by operations. The user can lock individual residues or whole secondary structures, giving them an ice-like appearance. Locks allow a kind of implicit selection; by locking two residues, the user can then easily operate on the residues between them.

Wiggle and shake. Wiggle and shake are two automatic actions the user can launch. Wiggle performs an optimization over the backbone, and shake performs an optimization over the sidechains. They can be performed globally as well as locally by using locks.

Rebuild. Rebuild allows the user to specify a section of the protein to be modified primarily by Rosetta fragment insertion, a process of copying backbone angles from similar native structures. This operation has a large element of randomness and can result in drastic changes to the structure.

The interactions in a scientific discovery game must also meet several criteria. They must *respect the constraints of the system* required. However, they must also be *sufficient to explore* the space of solutions enough to be able to solve the problem. They should also be as *intuitive and fun* as possible.

To ensure that the player interactions *respect the constraints* of protein folding, we developed a number of tools for players to use based on the powerful set of optimizations offered by Rosetta. By using Rosetta as a model for our interactions, we could ensure that they would result in plausible for protein structures. However, these optimizations only formed the basis for the moves, and they all needed to be adapted for interactive and intuitive use by players. The primary method of interaction in Foldit is directly manipulating the protein through pulling, by clicking and dragging the mouse. Depending on the location of the pull, this performs various Rosetta-based optimizations with the player's pull as a soft constraint. There are also buttons to launch automatic algorithms for continuous energy minimization (*wiggle*); discrete sidechain energy optimization (*shake*); fragment insertion (*rebuild*); and the ability to rigidly rotate, translate, and shift sections of the protein (*tweak*). Players are able to achieve fine-grained control over these optimization through two methods. First, *freezing*, which prevents parts of the protein from moving, and second, *bands*, which can connect amino acids and pull on them independently of the player. When making large restructuring operations, the repulsion force between atoms can overpower what the player is trying to do and make it more difficult to interact with the protein. To get around this, we have added a *behavior* menu, with a slider that allows player to adjust the strength of the repulsion during interactions.

There are additional modes for interaction that define what the mouse buttons do when the user clicks on the protein. The primary mode is the *pull mode*, described above. The *structure mode* allows the player to redefine the secondary structure labeling of the protein. This is done by directly assigning from a menu, or dragging existing labels across the protein. The *note mode* allows players to add their own notes and remarks to sections of the protein. These can be used by an individual or to communicate between players sharing solutions.

In order to confirm that the interactions available in Foldit were *sufficient to explore* enough protein structures to allow the players to make a discovery, we ran several puzzles in which the native structure was visible as a guide. With this native guide,

players were able to use the tools in Foldit to get close to the native structure. The fact that players were able to do this suggested that the interactions would be sufficient to reach the native structure on unknown proteins as well.

Further, to encourage players to use the available interactions to explore the space in new ways, we added an *exploration map*. The exploration map plots all solutions found by Foldit players for a puzzle based on their score and how different they are from the puzzle's starting structure. The map gives players a rough idea of the solution landscape: the different areas other players are exploring, and the scores they found there. Players might be exploring a new region on the map that initially gives a worse score, but by working hard in this new unexplored region, they might find a better shape and get the highest score.

In order to make the interactions more *intuitive and fun*, we followed the concept of *touchability*—being able to directly interact with the protein as though you could actually touch it. Before embracing this concept, our designs only manipulated the protein though indirect sliders, buttons, and plots. However, we soon changed the design to cause actions to occur by clicking on the protein itself. While the major optimizations are still launched by buttons, actions like pulling, attaching bands, freezing, tweaking, and others are performed directly on the protein. This also led us to the mode-based interface, which allowed us to use the mouse buttons for more operations by changing what they did in different modes.

The goals of interactivity and accuracy can sometimes conflict. Over time, the game evolved to address these dual goals. The automatic shake tool uses a discrete optimization over possible sidechain positions called rotamers. However, as the number of rotamers becomes large, running the optimization over all of them at once becomes too slow to be usable. Therefore, we rewrote the high level algorithm to run on small, spatially coherent sets of rotamers, rather than all at once. By running many shorter optimizations in sequence rather than one long one, players can see partial results and cancel the optimization early, while also getting the advantage of finer sampling from a larger number of rotamers.

Another place we traded off between interactivity and accuracy was in the ideality of inter-residue degrees of freedom. In order to allow local modifications to the backbone at interactive rates, we found it necessary to allow small non-idealities to occur in the protein structure at the points where residues connected—allowing degrees of freedom to vary that typically would remain fixed. A small amount of non-ideality was acceptable, so we allowed it, but included a penalty in the score. We found that, initially, the level of non-ideality in the solutions was unacceptably high. After increasing the penalties, the non-idealities fell into a range that was acceptable to the scientists, while still maintaining the desired player experience.

3.4.3 Scoring

By definition, the final outcome in a scientific discovery game is unknown. Therefore, we cannot base the goal of the game on reaching a particular known state. The goal must be one that will *direct players toward the solution* to the problem and encourages players to explore the space.

In Foldit, we want to motivate the players to find the best possible protein structures, yet we do not know what those structures are. To do this, we organize the game in the form of a competition, where a player's goal is to do better than other players. In a sense, this allows players who find good structures to set the goal for the other players. To make sure that better scores will *direct players toward the solution*, we base scoring on the Rosetta energy function, which reports a lower energy for structures nearer the native. However, we negate the energy so players are competing for a higher score. The energy function is broken up into several terms—such as clashing or hydrogen bonding—based on where the contribution to the score is coming from, and this information is made available to the players.

In each puzzle, players are ranked by the score of the best solution they have found, and at the close of a puzzle, *global points* are assigned based on ranking. These global points are accumulated over time, allowing players to have an overall ranking against all other players. Groups of players are also ranked and scored in a similar fashion. Initially, all players were ranked together in a single leaderboard. However, feedback from the players indicated that the solution sharing architecture was unfair to individuals working alone who had to compete against players in groups who could simply take another group member's solution and move to the top of the rankings. To prevent this unfairness, we separated rankings into soloists, for players who worked without trading solutions, and evolvers, for players who worked by improving other players' solutions. This dealt with the issue of unfairness while allowing us to reward specialization. However, it did cause players to have to divide their efforts if they were interested in ranking highly in both.

3.4.4 Introductory Levels

We do not expect players to have a background in the scientific problem, or even be familiar with it. Thus, if players are to be successful, it is necessary to *teach players the system* and how the gameplay, visualizations, interactions, and scoring work.

In Foldit, we *teach* gameplay concepts through a series of introductory levels—offline puzzles that have an associated goal score, which, when reached, will complete the level, unlocking the next one. Each level introduces the player to new problem related concepts as though they are the rules of a game. These concepts are introduced through

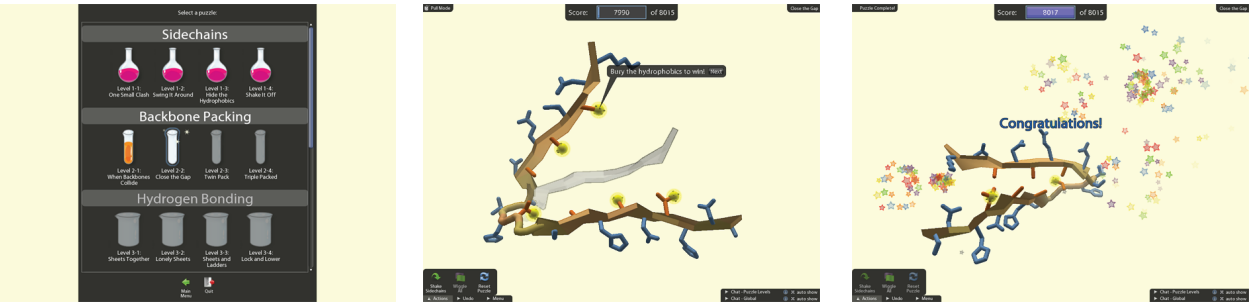


Figure 3.7 **Flow through introductory levels.** First, the player selects an available level from the menu. While playing the level, text bubbles pop up to guide the player. When the goal score is reached, a short reward animation is played. (Figure from Cooper et al. [2010b])

an event-driven system of "text-bubble" hints, where short text hints appear to guide the player based on what they are doing. The levels are designed such that using the newly introduced concepts is the easiest way to reach the goal score. The levels are organized into sets, each dealing with a particular high-level concept, such as hydrogen bonding. An example of the flow through of an introductory level is shown in Figure 3.7.

We refined these levels using both qualitative feedback from playtesting and quantitative data gathered from gameplay. First, before releasing levels, we performed think-alouds, where we would invite players to play through the levels and say out loud what they were thinking [Ramey et al. 2006]; this would help us to determine where players were confused and what worked well, as well as what adjustments could be made to the gameplay to be more fun. In each round, we interviewed 3–5 people and collected the top complaints and points of confusion. Speaking with additional people each round would have led to diminishing returns where players would repeat the issues found by other players. After each round, we quickly made improvements to the game and ran another set of think-alouds, until the major issues were resolved.

Second, once the levels were released, we gathered data on how far players progressed through them. This gave us a good high-level view of where the most troublesome places were for most people. We could then focus our efforts on levels that caused the largest drop-off in players, make adjustments, and observe the results. An example of this process is given in Figure 3.8. We are able to aggregate data over a period of time, instead of looking at just a single day.

New players are not required to finish any introductory levels before trying out the online puzzles. To help bridge the gap between the levels and the online puzzles and keep players from being completely overwhelmed by a potentially complicated or dif-

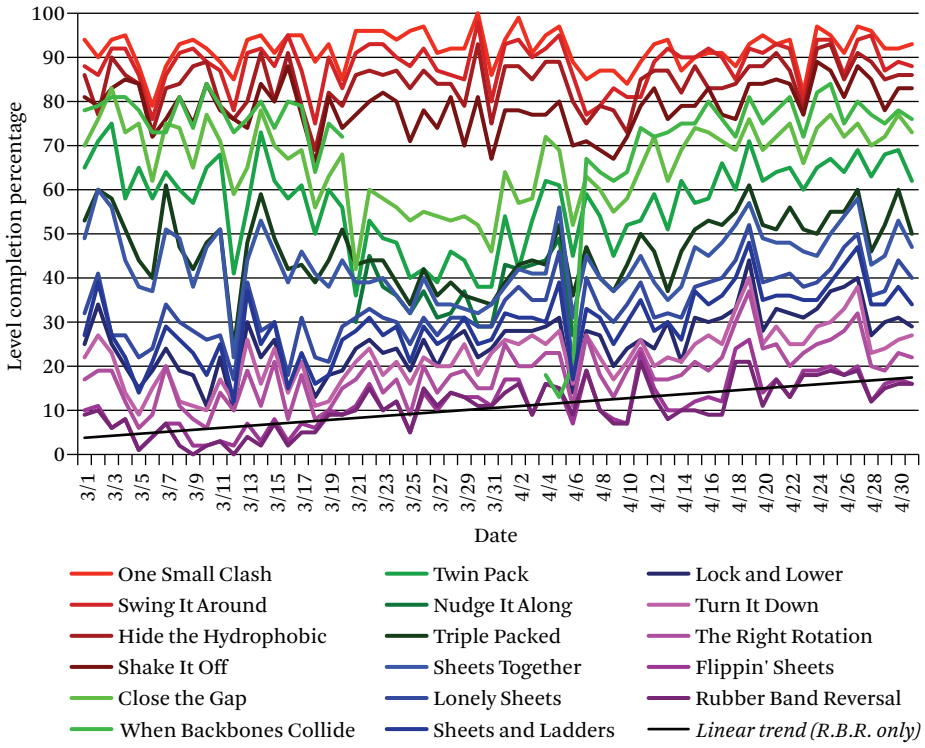


Figure 3.8 **Completion percentage for each introductory level per day over a two month period.** New players must start with the first level, "One Small Clash," and have to successfully complete it before attempting the next level, "Swing it Around," and so on, until the last level: "Rubber Band Reversal". On 3/21 all the intro levels were changed to use text bubble hints; the "Close the Gap" level was removed and "Nudge it Along" was added as a level. This closed the space between the blue levels, but created a large gap in the green levels. The removal of the level between the red and green lines seemed to be the main culprit, therefore on 4/06 "Close the Gap" was restored and "Nudge it Along" was removed. This fixed the large gap in the green levels while maintaining the overall shift upward; the linear trend of new players completing all the intro levels increases over this two month period (gray line at the bottom). (Figure from Cooper et al. [2010b])

ficult puzzle that has recently been posted, we include beginner online puzzles. These are similar to the other online puzzles, but available only to new Foldit players, who have fewer than 150 global points. Beginner puzzles are posted for one month and involve known solved proteins whose native fold has been randomly modified and is also shown as a transparent guide superimposed with the current solution. This native

guide serves to give newer players an attainable goal, as well as an idea of what native protein folds look like.

3.5 Rewards and Social Interaction

Foldit has several methods of rewarding for playing. We have found that social interaction can be a powerful force for motivating players and keeping them interested.

Foldit presents the player with challenging puzzles. Overcoming such a challenge can be a reward. Furthermore, competing with other players and attempting to outdo them is also motivating. The in-game scoreboard informs players of their relative standing, and top groups and players for each puzzle show up on the webpage. Each player's rank also appears near their name on the webpage, and each player has a personal page that displays their scores for the puzzles they have played.

Foldit is designed to provide players with short-, medium-, and long-term rewards. For example, in the short term, interacting with the protein causes colors to change and sounds to play. When players interact with the protein, they receive feedback. Good moves and success will cause positive sounds, like fanfare, to play and reinforcing messages to appear. In the medium term, players can increase their rank in an individual puzzle. Ranking up causes a large "Rank Up!" message to appear. In the longer term, players accumulate points for getting good rankings in puzzles. These then affect their overall ranking in the game. Players can also learn about the connection between the game and scientific outcomes. All of these systems are deisgned with the intention of rewarding players over different time scales.

The game provides support for interaction with other players through in-game chat. The chat allows sending screenshots through to facilitate discussion of solutions. The website also supports a bulletin board and messaging system.

The primary social organization in the game is that we allow players to form groups. In-game sharing of solutions is facilitated between group members, so that multiple people can work together to improve a solution. A player can upload their solutions and their group members can wee what solutions their group members have uploaded. Group members can then download those solutions and continue working on them, resharing them if they wish. In this way players can work together serially. Solutions can be marked up with comments to communicate with group members. Communication within groups is facilitated by group-based in-game chat and bulletin boards on the webpage.

For each puzzle, players are ranked by two separate scores, depending on if they have worked alone or started from another player's solution. Players have a soloist score, for solutions they have worked on alone, and an evolver score, for solutions they have

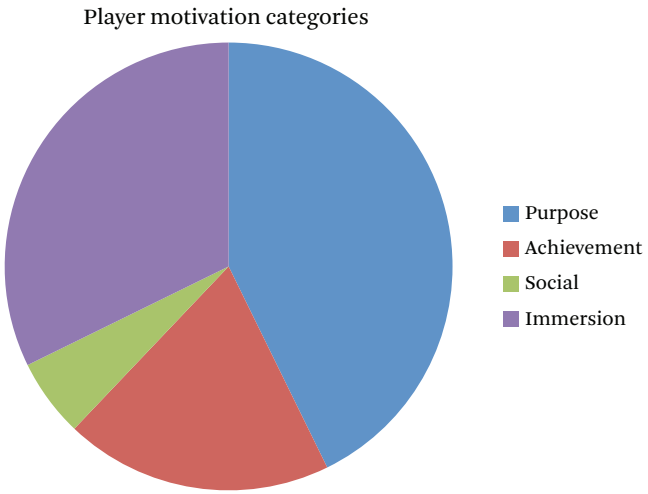


Figure 3.9 **Player motivation survey.** Results of a short informal survey posted on the Foldit website asking players their motivation for playing. 48 players responded with up to 3 reasons each. Responses were categorized based on Yee's main motivation components [Yee 2006], with an additional category for motivation related to Foldit's scientific *purpose*. Note that over half of the responses fall outside of the *purpose* category. Example responses for *purpose* include "To crack the protein folding code for science" and "To understand the folding process better"; for *achievement* "To get a higher score than the next player" and "It's fiendishly addictive in a Pavlovian manner (get points, feel good)"; for *social* "The people (community) are great" and "Great camaraderie"; and for *immersion* "It's fun and relaxing" and "I like the visualization of molecules". (Figure from Cooper et al. [2010a])

collaborated with others on. Each group also has a score, which is the maximum of the scores of all the members. The purpose of separate soloist and evolver scores is to not harshly penalize players who do not wish to play in groups, and reward players for the styles of play they at which are best.

A survey of Foldit players (Figure 3.9) revealed that while the purpose of contributing to science is a motivating factor for many players, Foldit also attracts players interested in achievement through competition and point accumulation, social interaction through chat and web-based communication, and immersion through engaging gameplay and exploration of protein shapes [Yee 2006]. We expect generally future scientific discovery games will also benefit from varied motivation sets.

We have found the game to be approachable by a wide variety of people, not only those with a scientific background (Figures 3.10 and 3.11); in fact, few top players are professionally involved in biochemistry (Figure 3.12).

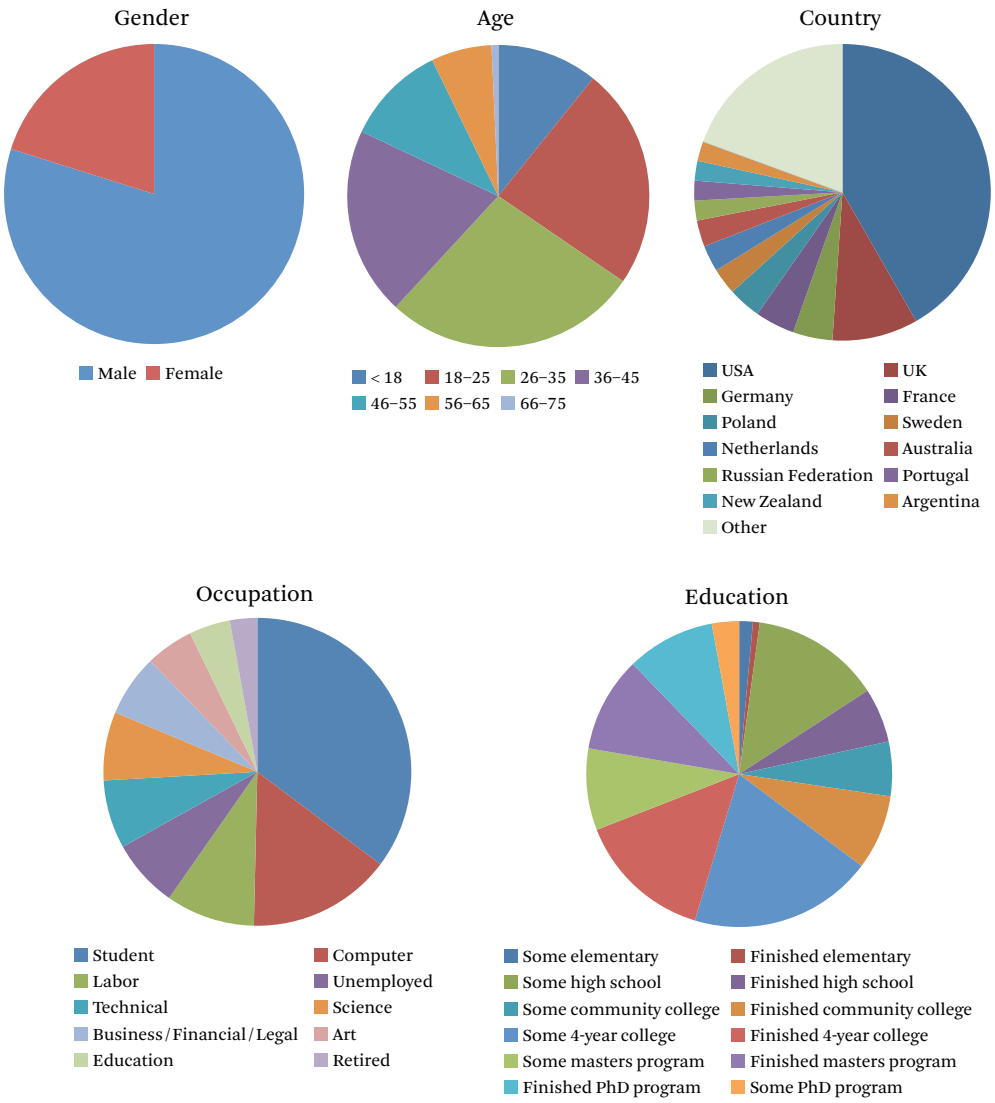


Figure 3.10 **Demographic survey for all players.** Results of a demographic survey posted on the Foldit website. There were 149 responses, showing the variety of backgrounds Foldit players have. Results for all responses are shown. (Figure from Cooper et al. [2010a])

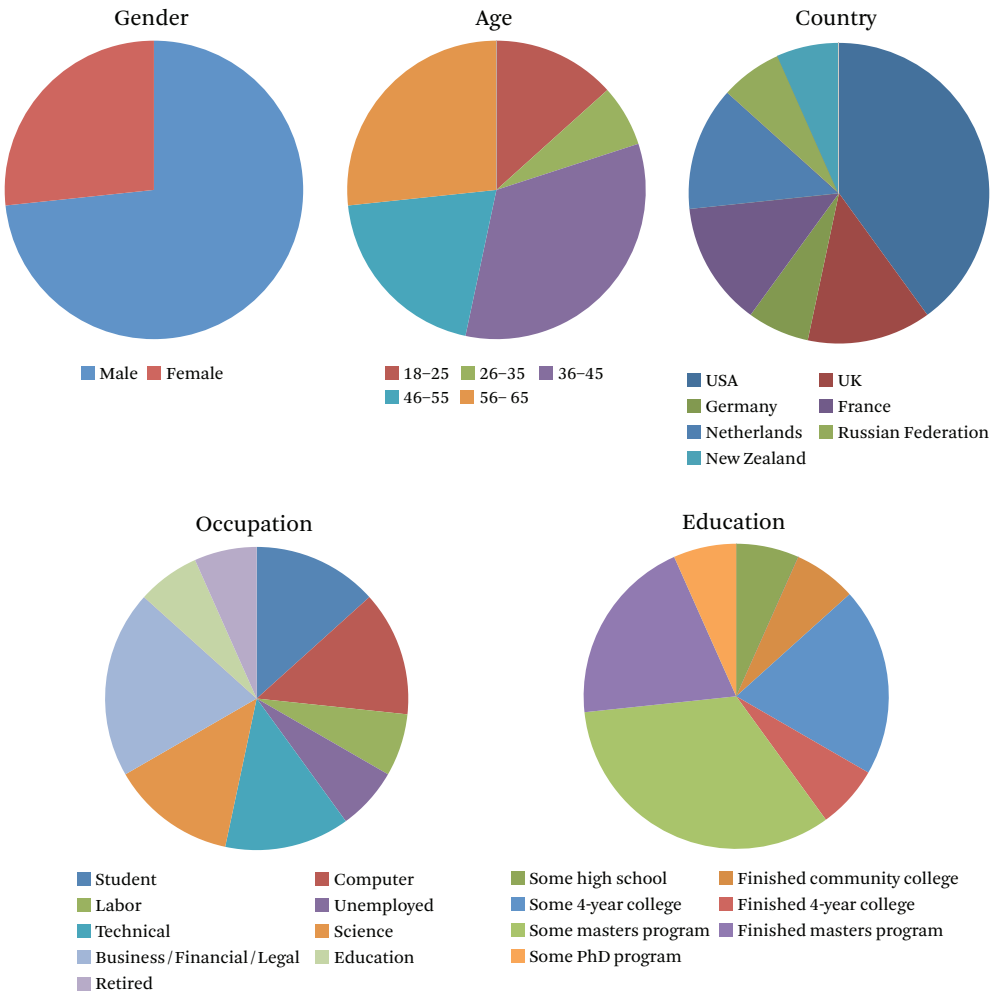


Figure 3.11 **Demographic survey for top players.** Results of the survey from Figure 3.10 for responding players who were in the top 50 soloists or evolvers. (Figure from Cooper et al. [2010a])

Players were able to find out about the game through a variety of means. When Foldit initially launched, announcements were made on the Rosetta@home forums. Foldit also received some publicity from the press at that time, and on occasion since then, which continues to allow new users to discover the game. Talks and word of mouth also help to attract new players.

In the first month after Foldit's launch (5/8/2008–6/8/2008), 59% of traffic to the Foldit website was from referring sites, 29% was direct traffic, and 12% was from search

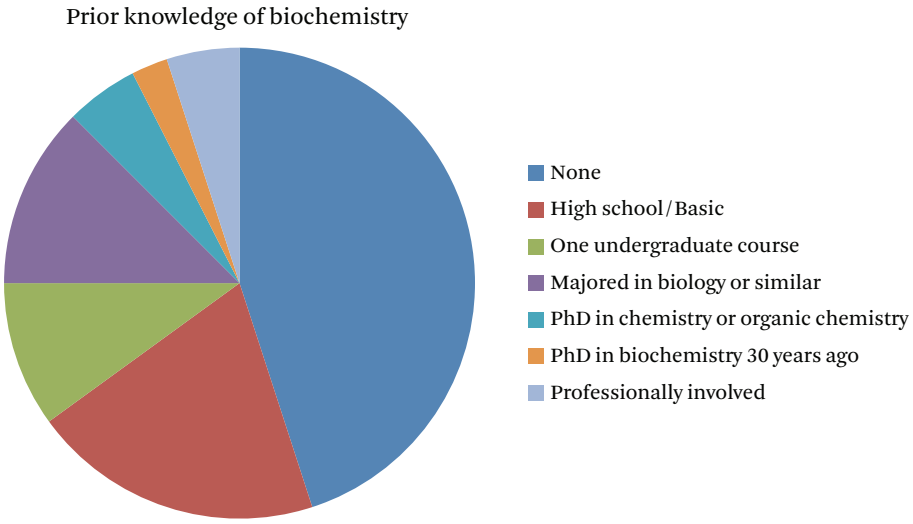


Figure 3.12 **Player biochemistry experience survey.** Results of an informal email survey asking players about their prior biochemistry knowledge. We emailed the top 20 Foldit players in all categories (soloist, evolver, all hands) asking them if they had any prior biochemistry experience. Most players replied along the lines of "none" or "I took basic chemistry in high school". Other players replied that they took "one quarter of chemistry as an undergrad" while some players had a "bachelor's degree in biology". A minority hold advanced degrees in some form of chemistry or are biochemists in their professions. (Figure from Cooper et al. [2010a])

engines. In the rest of 2008 (after 6/9/2008), traffic was 41% direct traffic, 34% referring sites, and 25% search engines. The year 2009 was similar, with 42% direct traffic, 31% referring sites, and 27% search engines.

3.5.1 Rewards and Ranking Types

The reward system in Foldit is set up to reward the absolute best score achieved by a player. However, we are interested in recognizing the efforts of players who work together. Thus, we have set up a system to reward players who work alone and together separately.

Within Foldit, players are globally ranked in different categories. Typical puzzles are divided into soloist and evolver rankings, and a player can only open solutions from other players in their same group. Soloist rankings are based on solutions that only one player has edited. Note that they may be able to look at other solutions and talk to other players about them, but they have done all the editing to create a particular

solution themselves. Evolver rankings are based on solutions that more than one player has edited. When a player opens a solution that has created by another player, they must improve its score by a small amount before they receive credit for it in the evolver rankings.

We have run a small set of puzzles that are set up such that the top solutions at any time are made available for all players to download and open, thus essentially allowing any player to work on improving the best solution. The resulting all hands rankings are based on solutions for these specific puzzles, regardless of how many edited a solution. In our preliminary exploration of this approach, we have found that this all hands system typically causes the players' solutions to converge more quickly to regions of the conformation space, rather than explore the space fully.

Foldit also has an achievement system—common in games—to reward players for performing specific discrete actions. These achievements reward a range of actions, including completing introductory levels, sharing solutions with their group, and getting high ranks on multiple puzzles. A player is informed in the game when they meet the requirements for an achievement, and a player's achievements are displayed on their webpage for other players to see.

Other reward schemes would be possible, such as rewarding the players by their relative contributions to solutions, rewarding more for collaboration, or for finding novel regions of conformation space. Further, collaboration and reward structures inspired by non-game contexts, such as open-source projects and the wiki model, could also be very effective. Experimenting with different reward systems and their effects remains an interesting avenue for future research.

3.6 Conclusion

This chapter introduced a general framework for mapping a scientific problem into a game. We have developed Foldit, while addressing the dual goals of engagement and scientific relevance. We have showed how the game's flexible architecture is able to coevolve along with the game's players. We have showed the teaching and reward structures in the game appeal to a wide variety of players, many of whom have little or no background in biochemistry.

Unlike most video games, where entertainment is the main goal and the design is entirely up to the creators, the design of Foldit was guided primarily by enabling anyone with a PC an opportunity to take part in scientific problem solving. One of the most difficult aspects of development was that we were designing a game in which the final outcome was not known. Even in Foldit puzzles where the best structure is unknown, the game has to guide the player toward that structure.

We described how we overcame these unique challenges and the coevolution approach we took to designing Foldit. We applied this approach to the visualizations and interactions in the game, as well as introducing the necessary concepts to players [Cooper et al. 2010b].

In designing Foldit, we learned the importance of including iterative adjustments to the game in the process of design, as the initial decisions can always be improved upon. We also learned not to expect the way that scientists view the problem to be the best way for players. Exploring different avenues for looking at scientific problems can lead to new and useful opportunities for problem solving. Thus, we also found that it is possible to make a fun experience by focusing on the exciting aspects of scientific problems, as opposed to the textbook details. We can take lessons from traditional game design to do this: rewarding players and keeping them interested are necessary for any game.

4 Protein Structure Prediction

4.1 Introduction

This chapter discusses applying the framework to the solution of a difficult biochemistry problem domain, the prediction of natural protein structures. Through the framework's combination of human problem solving and computational power, in certain cases game players predict structures more accurately than computational methods and perform well against other methods in worldwide competition. Players are also able to solve a long-standing structural biochemistry problem that was previously unsolvable through computational or experimental methods.

People exert significant amounts of problem solving effort playing computer games. Simple image- and text-recognition tasks have been successfully crowd-sourced through games [von Ahn and Dabbish 2004, von Ahn et al. 2006, Westphal et al. 2010], but it is not clear if more complex scientific problems can be similarly solved with human-directed computing. Protein structure prediction is one such problem: locating the biologically relevant native conformation of a protein is a formidable computational challenge given the very large size of the search space. Here we describe Foldit, a multiplayer online game that engages non-scientists in solving hard prediction problems. Foldit players interact with protein structures using direct manipulation tools and user-friendly versions of algorithms from the Rosetta structure prediction methodology [Rohl et al. 2004], while they compete and collaborate to optimize the computed energy. We show that top Foldit players excel at solving challenging structure refinement problems in which substantial backbone rearrangements are necessary to achieve burial of hydrophobic residues. Players working collaboratively develop a rich assortment of new strategies and algorithms; unlike computational approaches, they explore not only conformational space but also the space of possible search strategies. The integration of human visual problem-solving and strategy development capabilities with traditional computational algorithms through interactive multiplayer games is a powerful new approach to solving computationally-limited scientific problems.

While it has been known for over 40 years that the three-dimensional structures of proteins are determined by their amino acid sequences [Anfinsen 1973], protein structure prediction remains a largely unsolved problem for all but the smallest protein domains. The state-of-the-art Rosetta structure prediction methodology, for example, is limited primarily by conformational sampling; the native structure almost always has lower energy than any non-native conformation, but the free energy landscape that must be searched is extremely large—even small proteins have on the order of 1000 degrees of freedom—and rugged due to unfavorable atom-atom repulsion which can dominate the energy even quite close to the native state. To search this landscape, Rosetta uses a combination of stochastic and deterministic algorithms: rebuilding all or a portion of the chain from fragments, random perturbation to a subset of the backbone torsion angles, combinatorial optimization of protein sidechain conformations, gradient based energy minimization, and energy-dependent acceptance or rejection of structure changes [Das and Baker 2008, Qian et al. 2007, Bradley et al. 2005].

We hypothesized that human spatial reasoning could improve both the sampling of conformational space and the determination of when to pursue suboptimal conformations if the stochastic elements of the search were replaced with human decision making while retaining the deterministic Rosetta algorithms as user tools. We developed a multiplayer online game, Foldit, with the goal of producing accurate protein structure models through gameplay (Figure 4.1). Improperly folded protein conformations are posted online as puzzles for a fixed amount of time, during which players interactively reshape them in the direction they believe will lead to the highest score (the negative of the Rosetta energy). The player's current status is shown, along with a leaderboard of other players, and groups of players working together, competing in the same puzzle (Figure 4.1, arrows 8–9). To make the game approachable by players with no scientific training, many technical terms are replaced by terms in more common usage. We remove protein elements that hinder structural problem solving, and highlight energetically frustrated areas of the protein where the player can likely improve the structure (Figure 4.1, arrows 1–5). Sidechains are colored by hydrophobicity and the backbone is colored by energy. There are specific visual cues depicting hydrophobicity ("exposed hydrophobics"), interatomic repulsion ("clashes"), and cavities ("voids"). The players are given intuitive direct manipulation tools. The most immediate method of interaction is directly pulling on the protein. It is also possible to rotate helices and rewire beta sheet connectivity ("tweak"). Players are able to guide moves by introducing soft constraints ("rubber bands") and fixing degrees of freedom ("freezing") (Figure 4.1, arrows 6–7). They are also able to change the strength of the repulsion term to allow more freedom of movement. Available automatic moves—combinatorial sidechain rotamer packing ("shake"), gradient-based minimization ("wiggle"), frag-

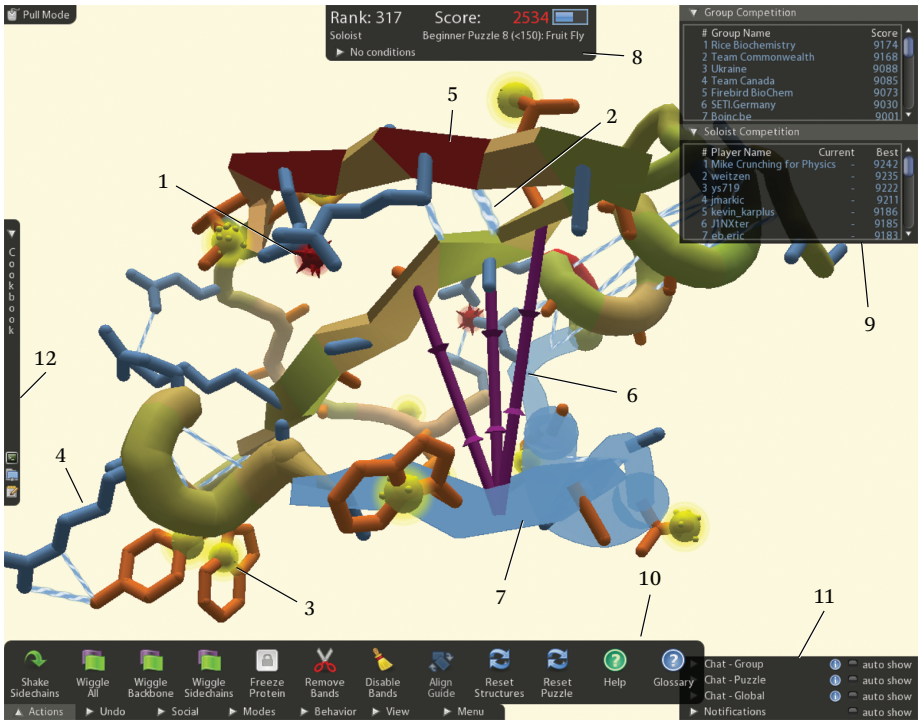
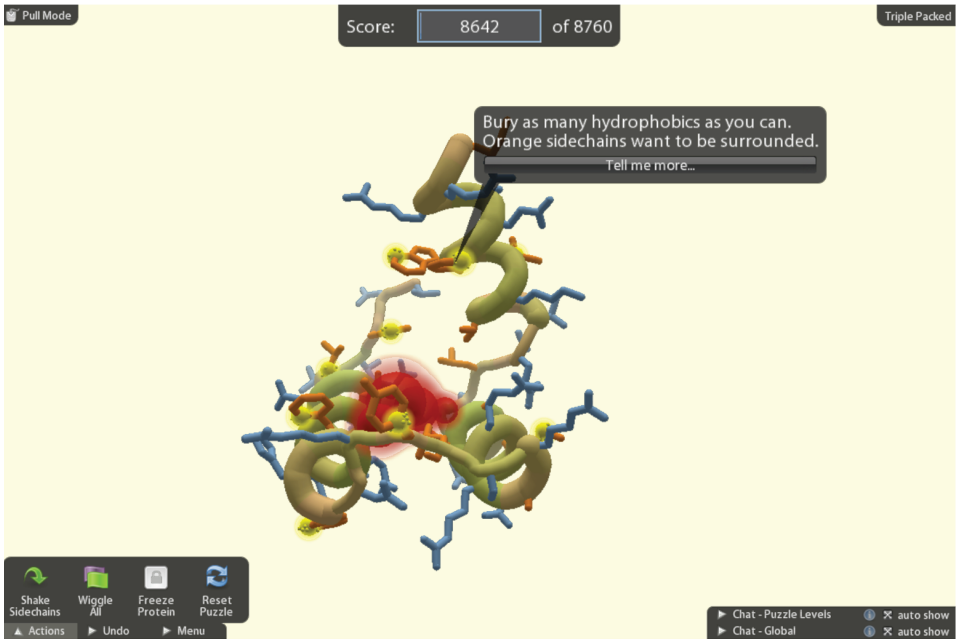


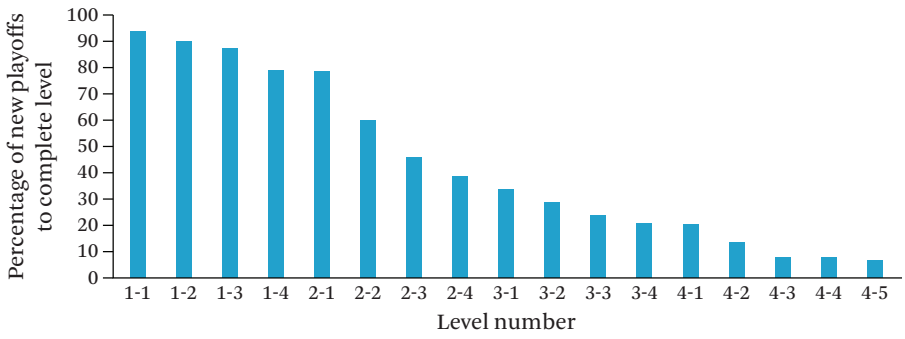
Figure 4.1 **Foldit screenshot illustrating tools and visualizations.** The visualizations include a clash representing atoms that are too close (arrow 1); a hydrogen bond (arrow 2); a hydrophobic sidechain with a yellow blob because it is exposed (arrow 3); a hydrophilic sidechain (arrow 4); and a segment of the backbone that is red due to high residue energy (arrow 5). The players can make modifications including bands (arrow 6), which add constraints to guide automated tools and freezing (arrow 7), which prevents degrees of freedom from changing. The GUI includes information about the player's current status, including score (arrow 8); a leaderboard (arrow 9), which shows the scores of other players and groups; toolbars for accessing tools and options (arrow 10); chat for interacting with other players (arrow 11); and a cookbook for making new automated tools or "recipes" (arrow 12). (Figure from Cooper et al. [2010a])

ment insertion ("rebuild")—are Rosetta optimizations modified to suit direct protein interaction and simplified to run at interactive speeds.

To engage players with no previous exposure to molecular biology, it was essential to introduce structure prediction concepts through a series of introductory levels (Figure 4.2 and Table 4.1): puzzles that are always available, and can be completed by



(a)



(b)

Figure 4.2 **Introductory levels.** (a) A screenshot of a Foldit introductory level. The levels are designed to teach players the basic concepts of protein folding. Each level is completed once a goal score is reached and the player then moves to the next level. In this level, the player is meant to bury the exposed hydrophobics in the voids that have appeared in the protein's core. Text bubbles appear to guide the player, give hints, and draw their attention to areas of interest. (b) Completion statistics for each introductory level, accumulated over one week. Each bar shows the percentage of new players who played any level, who also completed the given level. Players do not complete levels due to their difficulty, or because they decide to go directly to the scientific challenge puzzles. (Figure from Cooper et al. [2010a])

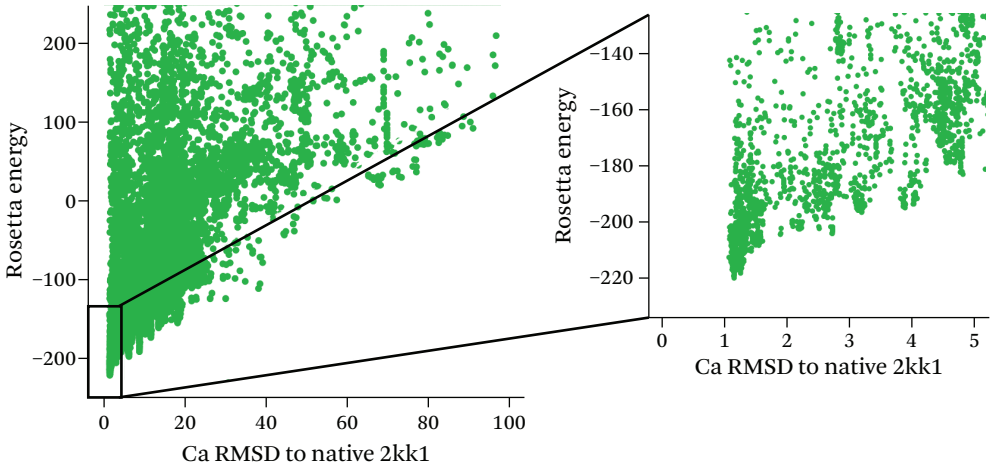
Table 4.1 **Structure prediction introductory levels.** Given are their order, name, and the main concepts introduced. (Table from Cooper et al. [2010a])

Number	Title	New Concepts
1-1	One Small Clash	Clashes; sidechain pull
1-2	Swing It Around	Camera controls
1-3	Hide the Hydrophobic	Exposed hydrophobics
1-4	Shake It Off	Shake
2-1	When Backbones Collide	Backbone pull
2-2	Close the Gap	Guide; wiggle all
2-3	Twin Pack	Voids
2-4	Triple Packed	Freeze
3-1	Sheets Together	Hydrogen bonds
3-2	Lonely Sheets	Rubber bands
3-3	Sheets and Ladders	
3-4	Lock and Lower	
4-1	Turn It down	Tweak rotate
4-2	The Right Rotation	
4-3	Flippin' Sheets	Tweak register shift
4-4	Rubber Band Reversal	Rebuild
4-5	Movin' Along	Rigid body moves

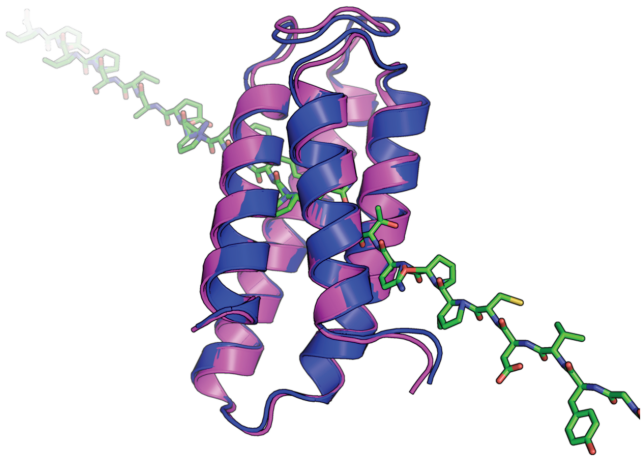
reaching a goal score. These levels teach the game's tools and visualizations, and certain strategies.

4.2 Quest to the Natives

To verify that the visualizations and tools are sufficient to achieve native conformations, we ran a series of "Quest to the Native" puzzles in which the native conformation is provided as a guide within the puzzle (Figures 4.3 and 4.4). We found that with this information players can use the available tools to consistently reach the native conformation. These puzzles also serve to familiarize players with the structural characteristics of native conformations.



(a)



(b)

Figure 4.3 **Quest to the Native extended chain.** Quest to the Native results starting from an extended chain conformation, showing that the tools in Foldit are sufficient to reach the native state. This "freestyle" Quest to the Native puzzle started as an extended chain, with the native shown as a guide. (a) RMSD plot of the Foldit player solutions for puzzle 986837, with the zoomed in box showing that the top scoring Foldit solution got within 1.183 Å of the native. (b) Superposition of the top scoring Foldit solution (in magenta) with model 1 of the native NMR structure 2kk1 (in blue). The starting extended chain conformation is also shown. (Figure from Cooper et al. [2010a])

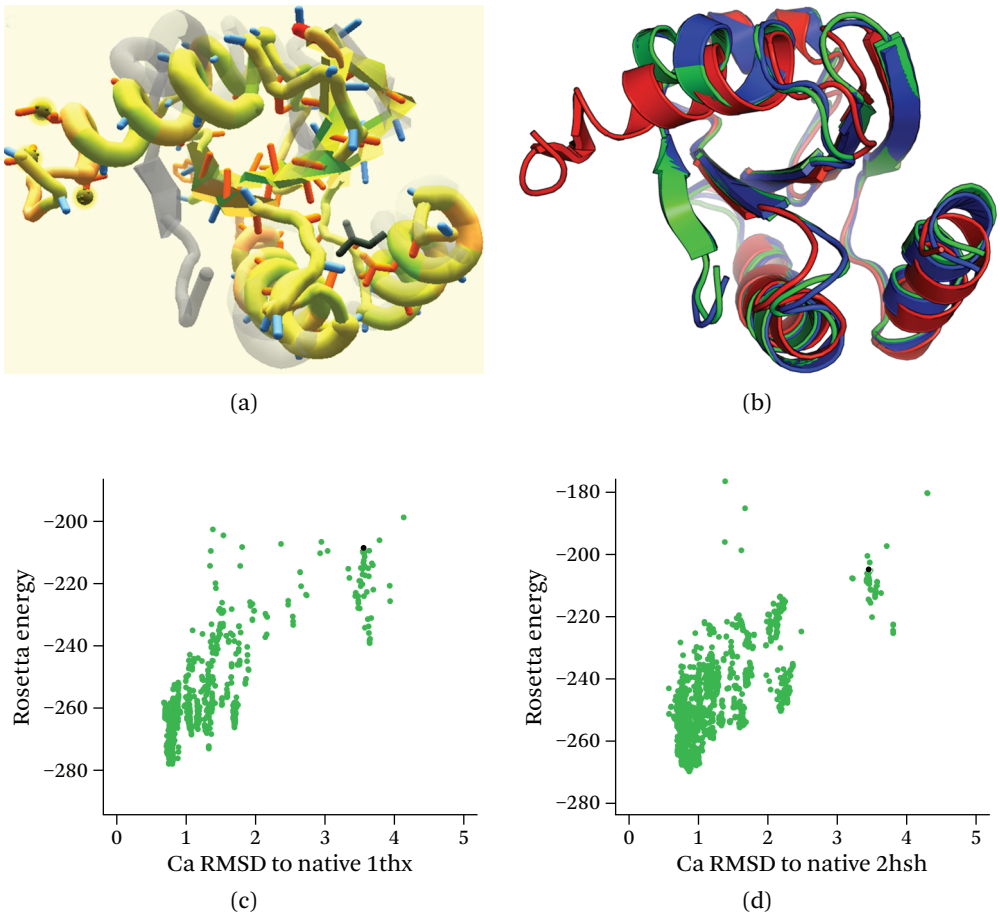


Figure 4.4 **Quest to the Native puzzles.** (a) A screenshot of a competition puzzle from the Quest to the Native series. These puzzles come with a transparent guide showing the native structure, which allows players to practice using the Foldit tools to match natives. (b) Superposition of the top scoring Foldit solution (in red) with the native structure 1thx (in blue) and the starting Foldit puzzle (in red). (c) RMSD plot of the Foldit player solutions (in green) for puzzle 986269, with the starting Foldit puzzle shown as the black dot. (d) RMSD plot of the Foldit player solutions (in green) for Quest to the Native puzzle 986597 that was based on the first strand swap Foldit puzzle (Figure 4.11). Only 9% of all Foldit players correctly swapped the strands in puzzle 986452, so we re-released the starting Rosetta model as a Quest to the Native puzzle with the solved native structure as a guide. This Quest to the Native puzzle allowed us to teach Foldit players of all levels how to swap strands as we did not have an intro puzzle explaining this type of move; we had previously never come across such a case in Foldit. The starting Foldit puzzle is shown as the black dot. This puzzle was used to teach Foldit players the tools necessary to swap strands. (Figure from Cooper et al. [2010a])

4.3 CASP8 Experiments

The Critical Assessment of Techniques for Protein Structure Prediction (CASP) is a biennial experiment aimed at assessing the state of the art in protein structure prediction methods. It is based on a set of proteins whose structures have been experimentally determined but not yet publicly announced. During CASP, the organizers of the experiment release the sequences as *targets* for prediction. Teams then submit predictions to CASP for evaluation. More information about CASP can be found on its website.[1] CASP8 took place over the summer of 2008, and several structures generated by Foldit players were submitted to CASP8 as part of the Baker Lab team.

To prepare for CASP8, we ran several experiments on targets from CASP7, because the target structures were available, and thus we could evaluate players' performance. This also highlights how we are able to use Foldit to determine how humans can be useful, and modify the game to take advantage of that.

We considered the types of targets released for CASP8 that would be appropriate for Foldit. We decided to focus on *homology modeling*, in particular *refinement*. Homology modeling can be used when a sequence with unknown structure is similar to a sequence with known structure. The known structure can be used as a *template* onto which the unknown structure can be threaded. The unknown structure can then be refined from that point to improve it. Refinement is known to be a challenging problem, as it is often difficult to move the template closer to the native structure [Tress et al. 2005].

The Baker Lab presented us with structures for our refinement experiments. We posted these structures as puzzles for Foldit, and compared the Rosetta energy and GDT-HA of the solution structures with the native structure. GDT-HA is a high quality metric for comparing protein structures that has been used by CASP assessors [Read and Chavali 2007]. GDT-HA compares the corresponding carbon alphas of two structures; a residue's carbon alpha is the atom on the backbone that connects to the sidechain. GDT-HA is the average of the percentage of carbon alphas that are closer than 0.5, 1.0, 2.0, 4.0, and 8.0 angstroms. Thus, if all carbon alphas are further apart than 8.0 angstroms, it would be 0, and if all are closer than 0.5 angstroms, it would be 100. A higher GDT-HA indicates closer structures, and since we are comparing to the native, better quality structures.

The first were puzzles were from CASP7 target T0308. The results of high scoring solutions can be seen in Figures 4.5(a), 4.5(b), and 4.5(c). The PDB codes of their templates were 1r4a, 1e0s, and 1zd9, respectively. We observed that although players were

1. http://predictioncenter.org/; last retrieved May 2014.

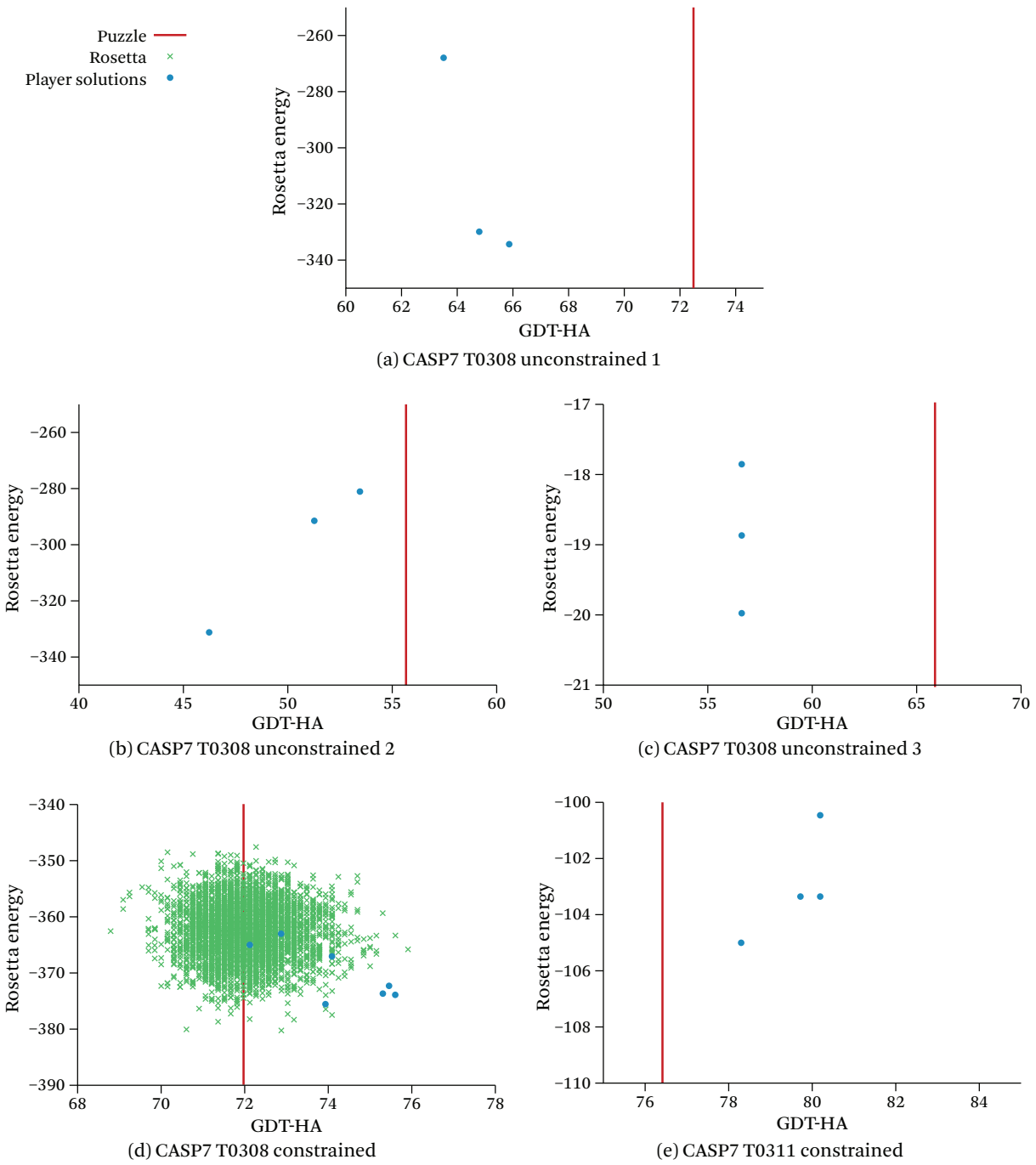


Figure 4.5 **Score and GDT-HA comparison for CASP7 puzzles.** A higher GDT-HA indicates a higher quality structure. (a), (b), and (c) are for T0308 with no constraints. (d) is for T0308 with constraints; structures generated by the Rosetta relax protocol are also shown. (e) is for T0311 with constraints. The puzzles with constraints produced higher quality solutions.

improving the energy of the structures, they were actually making them worse in terms of quality.

In response to this, we added *constraints* to puzzles. The constraints we used apply a pairwise distance penalty term to carbon alphas. This prevented the players' structures from straying too far from the puzzle structure. We then posted refinement structures with constraints provided by the Baker Lab for CASP7 T0308 and T0311. We observed that with constraints, the solutions not only improved the energy of the structures but also the quality when compared to the solutions without constraints. We also compared the players' solutions to T0308 to those generated by an automatic Rosetta relax protocol. We found that when compared to the relax protocol, the best player solutions were better in terms of both energy and quality. The results of the constrained experiments can be seen in Figures 4.5(d) and 4.5(e). The PDB codes of their templates were 1r4a and 1y7y, respectively. Note that although some of the structures generated by relax have good quality, they have poor energy, and thus would be difficult to find. Further, these protocols tend to converge after a certain point and running them longer leads to diminishing returns. These results show how we are able to experiment and modify the game to determine how players can be most useful.

Due to the positive results from our CASP7 experiments, we chose to make predictions for CASP8 targets with similar properties. For CASP8 targets, we ran several constrained puzzles and gave the Baker Lab access to player's solutions. For comparison, structures were retrieved from ftp://iole.swmed.edu/pub/casp8/targetpdb/ after the close of CASP8. We compared the GDT-HA of these submissions with the automated server submissions from other teams. For target T0423, three of the Baker Lab's five CASP8 submissions were solutions from Foldit. A histogram of the comparison can be seen in Figure 4.6(a). For target T0389, three of the five submission were also solutions from Foldit, compared in Figure 4.6(b).

Thus, by performing experiments, observing player's solutions, and modifying Foldit appropriately, we were able to make contributions to the Baker lab's CASP8 submissions. It is also interesting to note that the player's solutions do substantially better than many of the automated servers, and in several cases were of higher quality than the structures produced by Rosetta alone.

4.4 Evaluation

To evaluate if players could use the current design of Foldit to find the solutions to unknown scientific problems, we made several entries into the CASP8 competition in 2008 [Moult et al. 2009]. CASP is a semi-annual competition of protein structure prediction methods. Sequences (called targets) whose structures are unpublished, but

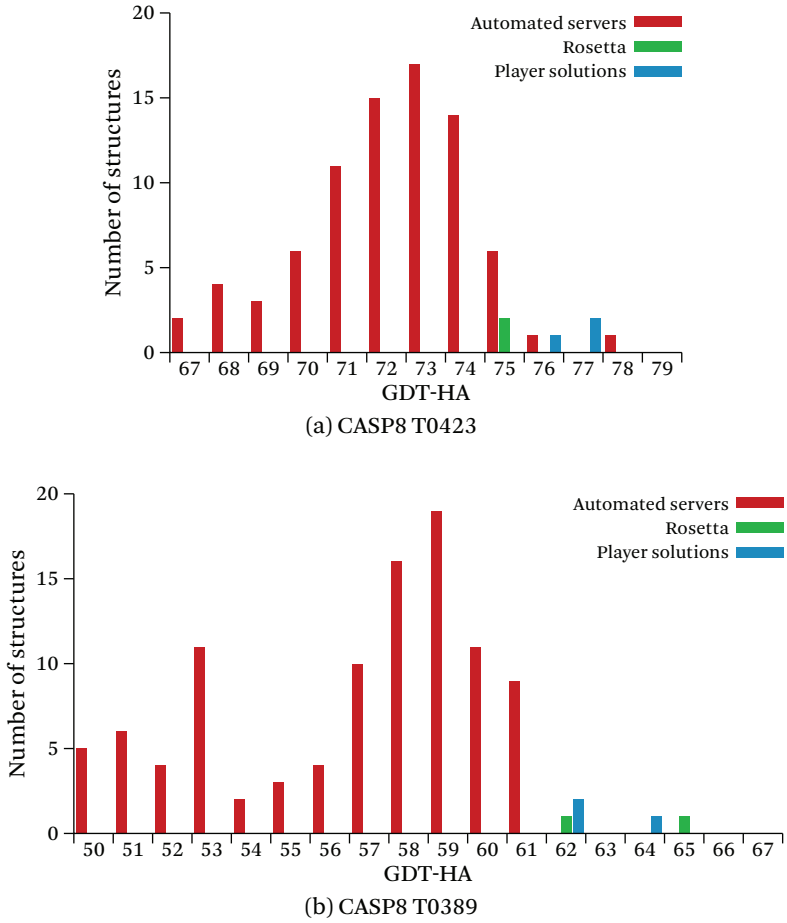


Figure 4.6 **GDT-HA histograms for CASP8 puzzles** The histograms compare predictions from automated servers, Rosetta, and Foldit players. The whole range of the servers' GDT-HA is not shown. (a) is for T0423; the GDT-HA for the servers ranged from 5.2 to 78.4. (b) is for T0389; the GDT-HA for the servers ranged from 6.9 to 61.6. The player solutions compare favorably with the other predictions.

have or will soon be experimentally determined, are posted, and teams make their predictions of the native structure. This ensures that predictions are blind—no one entering the competition knows the solution to the problem. In CASP8, we participated primarily in *homology modeling* targets. The sequences of these targets (with unknown structure) are similar to sequences with known structures, and homology information from related structures can be useful for making predictions.

To prepare for CASP8, we ran several puzzles using homology targets from CASP7 in order to refine our methods. Because these targets were from the previous CASP, we had access to the native structures for evaluation. In the first CASP7 puzzles we ran, we found that the players would often find solutions further from the native than the puzzle's starting structure. Moving away from the native is typical problem in homology modeling when refining structures. Consulting with biochemists, we decided to add scientist-defined penalty constraints to the scoring of the puzzles. These constraints integrated homology information from structures believed to be close to the native. The constraints affect the score function to penalize structures that move in less plausible ways. After running puzzles with these constraints, we found that players could improve on the quality of the starting structures.

Foldit player solutions were submitted for nine different CASP8 targets as part of the DBAKER team. For each target, several puzzles were run with various parameters and starting structures derived from homology information, in order to give the players a variety of places to start from in their search. The solutions from these puzzles, often numbering in the thousands, were aggregated and presented to the scientists. From this pool, the scientists selected submissions by a process of clustering the lowest energy structures, then selecting from those based on cluster size, energy, and visual inspection. This is similar to the process used to select automated prediction submissions from a large pool [Raman et al. 2009].

The resulting rankings are given in Table 4.2, for both the submissions that came from Foldit player solutions and the remaining DBAKER submissions that did not involve Foldit.

It is worth pointing out that this ranking (by the GDT_TS metric [Zemla 2003]) is just one of many possible for CASP assessment; for all but one of these targets the majority of predictions from the DBAKER team involved Foldit, and the puzzle constraints placed significant limits on what the players were able to do. However, we believe these results support the conclusion that the game has been designed in such a way that players can use it to solve scientific problems.

4.5 Rebuild and Refine Comparison

To evaluate players' abilities to solve structure prediction problems, we posted a series of prediction puzzles. Puzzles in this series were blind, in the sense that neither the target protein nor homologous proteins had structures contained within publicly available databases for the duration of the puzzles. Detailed information for these 10 blind structures, including comparisons between the best scoring Foldit predictions and the best scoring Rosetta predictions using the rebuild and refine protocol [Qian et al. 2007],

Table 4.2 **Rankings of predictions in CASP8 from the DBAKER team for targets that included Foldit solutions.** Rankings for the remaining DBAKER prediction, which did not use Foldit, are also given. Rankings were based on the GDT_TS metric [Zemla 2003]. (Table from Cooper et al. [2010b])

Target	Foldit Rankings	Other DBAKER Team Rankings	Total Entry Count (all teams)
TR389	3, 7, 21	4, 18	71
TR432	14, 37, 48	1, 8	83
TR453	23, 46, 69	28, 50	85
TR461	2, 12, 19	1, 26	83
TR469	2, 3, 45, 50	4	74
TR488	1, 3, 26	2, 18	77
TS423	12, 42, 43	2, 31	496
TS492	2, 96	1, 3, 448	527
TS499	59, 107	33, 455, 504	519

is given in Table 4.3. We found that Foldit players were particularly adept at solving puzzles requiring substantial backbone remodeling to bury exposed hydrophobic residues into the protein core (Figure 4.7). When a hydrophobic residue points outward into solvent, and no corresponding hole within the core is evident, stochastic Monte Carlo trajectories are unlikely to sample the coordinated backbone and sidechain shifts needed to properly bury the residue in the core. By adjusting the backbone to allow the exposed hydrophobic residue to pack properly in the core, players were able to solve these problems in a variety of blind scenarios including a register shift and a remodeled loop (Figure 4.7(a-b), a rotated helix (Figure 4.7(c), two remodeled loops (Figure 4.7(d), and a helix rotation and remodeled loop (Figure 4.7(e).

Players were also able to restructure beta sheets in order to improve hydrophobic burial and hydrogen bond quality. Automated methods have difficulty performing major protein restructuring operations to change beta sheet hydrogen-bond patterns, especially once the solution has settled in a local low-energy basin. Players were able to carry out these restructuring operations in such scenarios as strand swapping (Figure 4.8) and register shifting (Figure 4.7(a). In one strand swap puzzle, Foldit players were able to get within 1.06 Å of the native, with the top scoring Foldit prediction being 1.36 Å away. A superposition between the starting Foldit puzzle, the top scoring Foldit solution, and model 1 of the native NMR structure 2kpo are shown in Figure 4.8(b).

Table 4.3 **A listing of all the Foldit puzzles run in the blind data set.** A CA-RMSD comparison to the native is given between the best scoring model produced by Foldit players and the best scoring model produced by the Rosetta rebuild and refine protocol, given the same starting model(s). Solutions considerably better with one method than the other are indicated in bold. The solved structures (which were released after each puzzle ended) are represented by their PDB codes. Results from these Foldit puzzles can be accessed on the Foldit website by using the corresponding Foldit puzzle ID at http://fold.it/portal/node/ID. 2kky, 2kpt, 2kpm, 2kk1, and 2knr were taken from the CASD-NMR experiment [Rosato et al. 2009]. 2kpo was provided by Nobuyashu and Rie Koga. 2ki0 and 3epu were found by searching for unreleased structures on the PDB website (http://www.rcsb.org/pdb/search/searchStatus.do). 3lur and 3nuf were provided by the JCSG. Figures containing results for each puzzle are provided in the last column. (Table from Cooper et al. [2010a])

Puzzle ID	Foldit CA-RMSD	Rebuild and Refine CA-RMSD	Native	Method	Length	Figure(s)
986875	1.4	4.5	2kpo	NMR	99	4.8, 4.9
986698	1.8	3.7	2kky	NMR	102	4.10
986836	5.7	6.6	3epu	X-ray	136	4.7c, 4.11d
987088	3.5	4.3	2kpt	NMR	116	4.7a, b, 4.11a, b
987162	4.5	5.2	3lur	X-ray	158	4.11c
987076	3.3	3.5	2kpm	NMR	81	4.7e, 4.9c
986629	3.5	3.3	2kk1	NMR	135	4.9b
987145	2.6	2.3	3nuf	X-ray	105	4.7d, 4.9a
986844	6.9	5.8	2ki0	NMR	36	4.15a
986961	10.6	5.7	2knr	NMR	118	4.15b

Rosetta's rebuild and refine protocol, however, was unable to get within 2 Å of the native structure (Figure 4.8(a), yellow points). This example highlights a key difference between humans and computers. As shown in Figure 4.8(c), solving the strand swap problem required substantially unraveling the structure (Figure 4.8(c), right), with a corresponding unfavorable increase in energy (Figure 4.8(c), left). Players persisted with this reconfiguration despite the energy increase because they correctly recognized the swap could ultimately lead to lower energies. In contrast, while the Rosetta rebuild and refine protocol did sample some partially swapped conformations (Figure 4.8(a), leftmost yellow point), these were not retained in subsequent generations due to their

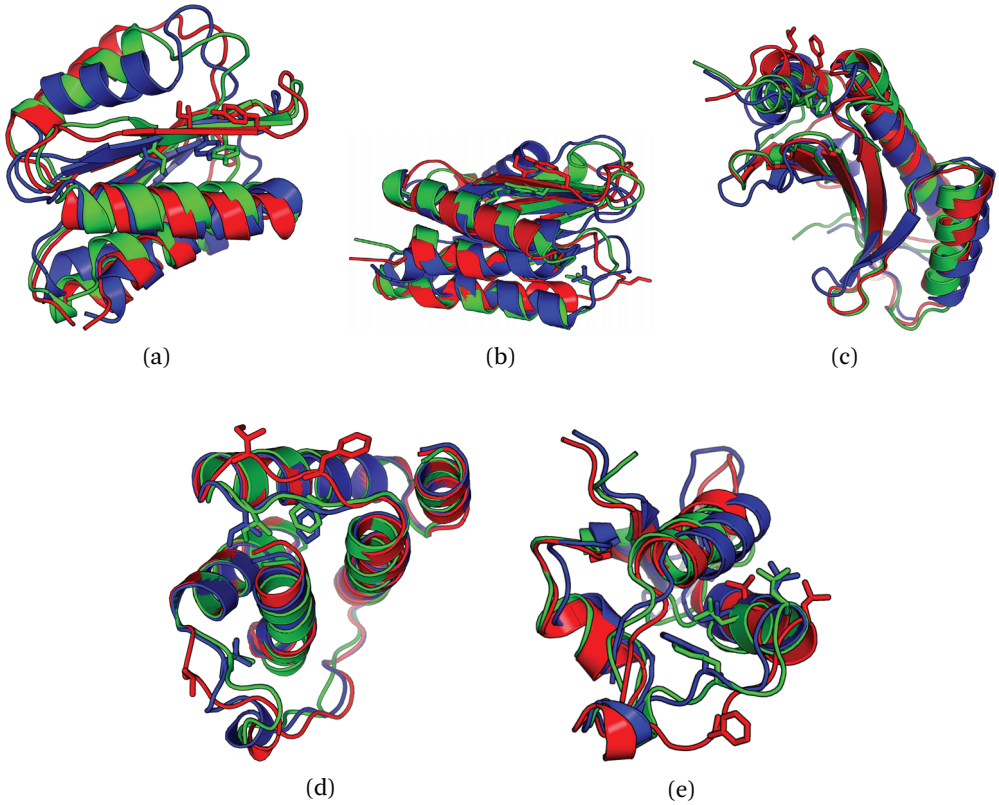


Figure 4.7 **Structure prediction problems solved by Foldit players.** Examples of blind structure prediction problems in which players were successfully able to improve structures. Native structures are shown in blue, starting puzzles in red, and top scoring Foldit predictions in green. (a) The red starting puzzle had a register shift and the top scoring green Foldit prediction correctly flips and slides the beta strand. (b) On the same structure as above, Foldit players correctly buried an exposed isoleucine in the loop on the bottom right by remodeling the loop backbone. (c) The top scoring Foldit prediction correctly rotated an entire helix that was misplaced in the starting puzzle. (d) The starting puzzle had an exposed isoleucine and phenylalanine on the top, as well as an exposed valine on the bottom left. The top scoring Foldit prediction was able to correctly bury these exposed hydrophobic residues. (e) Another successful Foldit helix rotation that correctly buries an exposed Phenylalanine. Images were produced using PyMOL software http://www.pymol.org/; last retrieved May 2014.. (Figure from Cooper et al. [2010a])

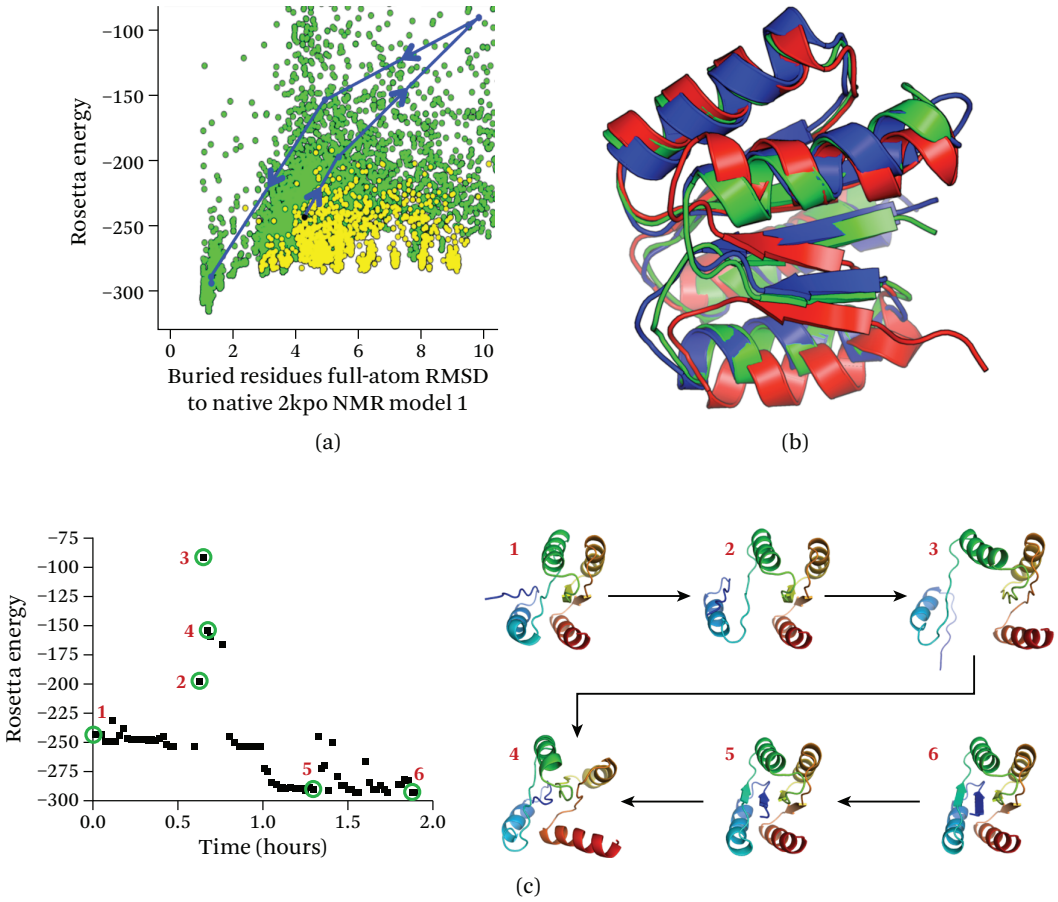


Figure 4.8 **Human predictors outperform the Rosetta rebuild and refine protocol by strand swap.** The puzzle shown is 986875. (a) Comparison of Foldit player solutions (green) to the low energy structures sampled in Rosetta rebuild and refine trajectories (yellow) for blind Foldit puzzle 986875 based on the recently determined structure and sequence of 2kpo. The x-axis is the all-atom RMSD to 2kpo, and the y-axis is the Rosetta energy. The starting Foldit puzzle was 4.28 Å away from the native structure (shown by the black dot on the plot); Foldit players sampled many different conformations, with the top scoring submission (the lowest scoring Rosetta energy) 1.4 Å away from the native, while the automated Rosetta protocol did not sample below 2 Å. The blue dots and lines correspond to the trajectory of a single Foldit player in c. (b) Superposition of the top-scoring Foldit prediction in green with the experimentally determined NMR model 1 in blue. The starting puzzle is in red, where the terminal strand is incorrectly swapped with its neighbor, 8% of all Foldit players were able to correctly swap these strands (Table 4.4). (c) A score trajectory with selected structures for the top scoring player in puzzle 986875 over a two hour window, showing how the player explores through high energy conformations to reach the native state. The y-axis is the Rosetta energy and the x-axis is the elapsed time in hours. The starting structure had a Rosetta energy of −243. Each point in the plot represents a solution produced by this player. The first structure (c1) is near the starting puzzle structure, shown as the black dot in a. The following structures (c2-6) are shown as blue dots in plot a. In structures c2-4 the player must explore higher energies to move the strand into place, shown by the blue lines. In structures c5-6 the player refines the strand pairing. (Figure from Cooper et al. [2010a])

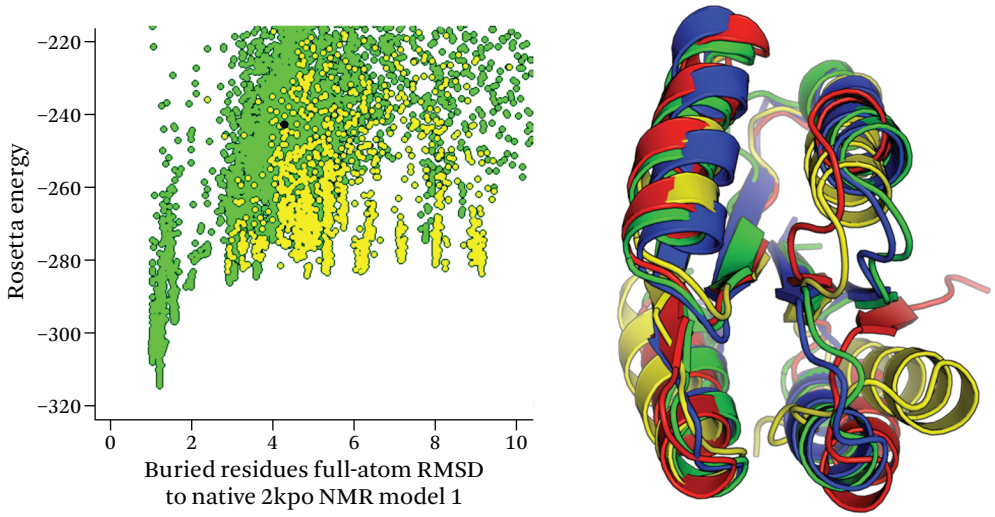


Figure 4.9 **Strand swap case where Foldit players outperform Rosetta (*blind trial*).** Comparing Foldit players to Rosetta rebuild and refine. (a) This is a zoomed in version of the full-atom RMSD plot shown in Figure 4.8(a). The starting Foldit puzzle was 4.28 Å away from the native structure (shown by the black dot on the plot). This is a comparison of Foldit player solutions (green) to the low energy structures sampled in Rosetta rebuild and refine trajectories (yellow) for blind Foldit puzzle 986875 based on the recently determined structure and sequence of 2kpo. The best scoring Foldit prediction was 1.4 Å away from the native, while the best scoring Rosetta rebuild and refine prediction was 4.5 Å away. (b) The native structure is shown in blue with the starting Foldit puzzle in red and the top scoring Foldit prediction shown in green. The best scoring Rosetta rebuild and refine prediction given the same starting model (shown in yellow) was unable to fix the strand swap. (Figure from Cooper et al. [2010a])

relatively high energies, resulting in the top Rosetta prediction being further from the native than the starting structure (Figure 4.9).

Human players are also able to distinguish which starting point will be most useful to them. Figure 4.10 shows a case where players were given ten different Rosetta predictions to choose from. Players were able to identify the model closest to the native structure, and to improve it further. Given the same 10 starting models, the Rosetta rebuild and refine protocol was unable to get as close to the native as the top scoring Foldit predictions.

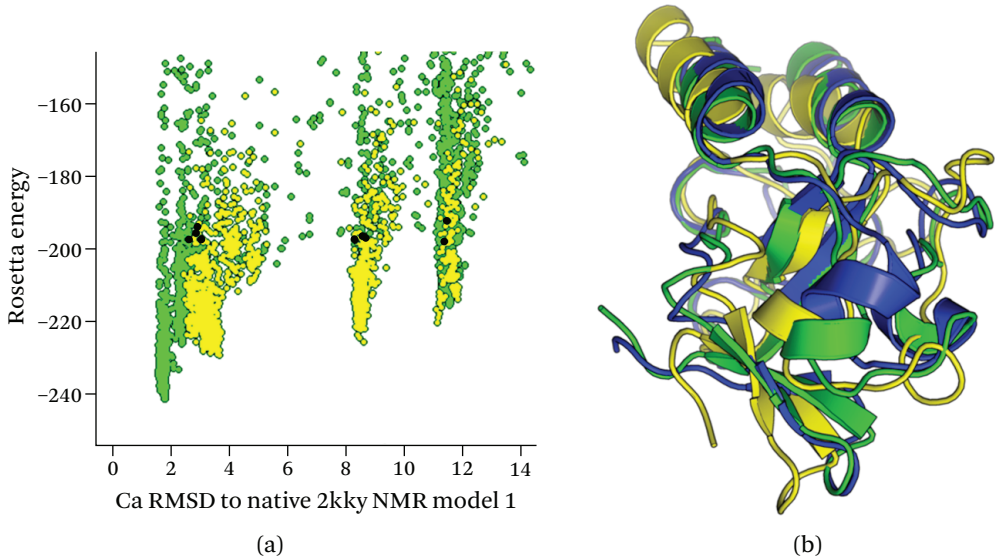


Figure 4.10 **Human predictors outperform the Rosetta rebuild and refine protocol by model selection** The puzzle shown is 986698. (a) Comparison of Foldit player solutions (green) to the low energy structures sampled in Rosetta rebuild and refine trajectories (yellow) for blind Foldit puzzle 986698 based on the recently determined structure and sequence of 2kky. Foldit players were able to get the best Foldit score by correctly picking from multiple alternative starting Rosetta models (black) the model that was closest to the native structure. (b) The native structure is shown in blue with the top scoring Foldit prediction shown in green. The top Rosetta rebuild and refine prediction given the same 10 starting models (shown in yellow) was unable to sample as close to the native as the Foldit players. (Figure from Cooper et al. [2010a])

Foldit players performed similarly to the Rosetta rebuild and refine protocol for 3 of the 10 blind puzzles (Figure 4.11). They outperformed Rosetta on five of the puzzles (Figures 4.8, 4.10, 4.9, and 4.12), including the two above cases where players performed significantly better. A larger set of successful solutions for similar, although non-blind, puzzles are described in Figures 4.13, 4.14, and 4.15. For 2 of the 10 blind puzzles, the top Rosetta rebuild and refine prediction was numerically better than the Foldit solution (Table 4.3) but still basically incorrect (RMSD to native structure > 5.7 Å) (Figure 4.16).

Despite the promising results described above, there still exists room for improvement. For one particularly difficult class of problems, players are only given an extended protein chain to start from. Although the Foldit tools are sufficient to reach

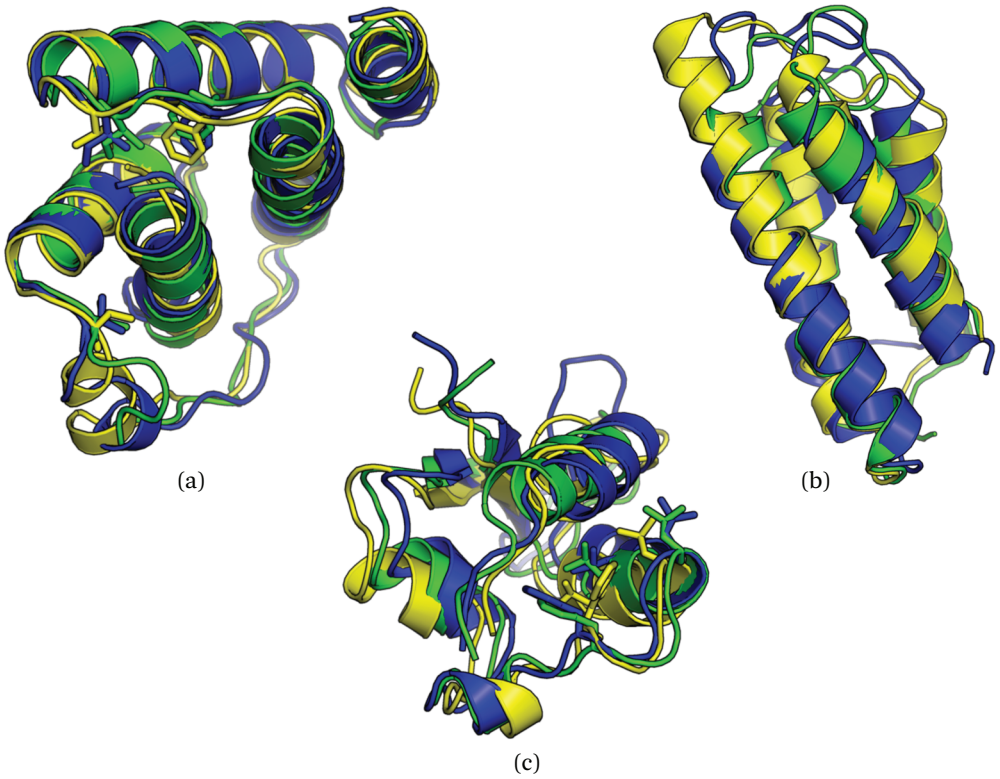


Figure 4.11 **Foldit players performing similarly to Rosetta (*blind trials*).** Comparing Foldit players to Rosetta rebuild and refine. Native structures are shown in blue. The top scoring Foldit predictions are shown in green. The best scoring Rosetta rebuild and refine predictions given the same starting model are shown in yellow. (a) This is the same puzzle as in Figure 4.7(d). The best scoring Foldit and Rosetta predictions were able to correctly bury the hydrophobic residues. (b) Both the best scoring Foldit and Rosetta predictions had trouble with the loops and ends of the helices in puzzle 986629. (c) This is the same puzzle as in Figure 4.7(e). The best scoring Foldit and Rosetta predictions were able to correctly rotate the helix and bury the exposed Phenylalanine. (Figure from Cooper et al. [2010a])

the native conformation from this unfolded start (Figure 4.3), players can have trouble reaching it from so far away (Figure 4.16(a). This indicates the need to find the right balance between humans and computational methods; players guided by visual cues perform better in resolving incorrect features in partially correct models than "blank slate" de novo folding of an extended featureless protein chain.

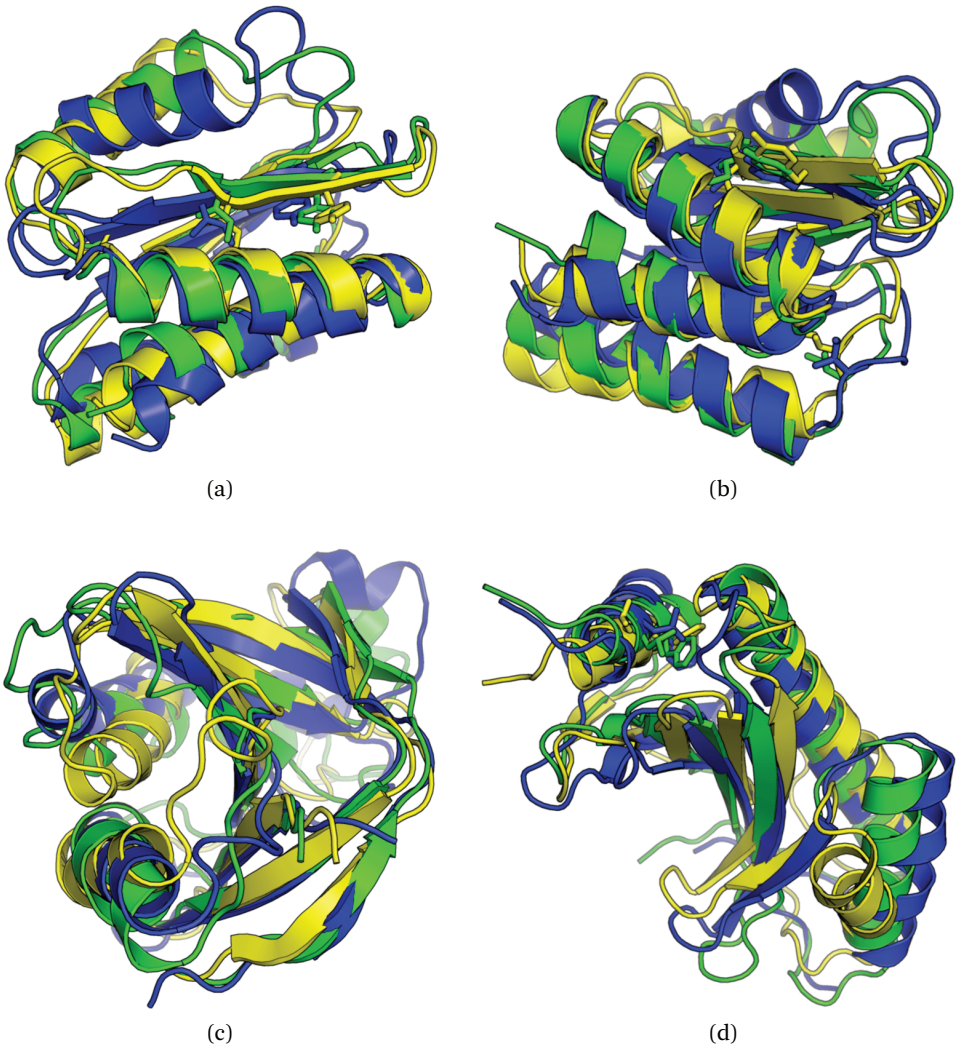


Figure 4.12 **Foldit players outperforming Rosetta (*blind trials*).** Comparing Foldit players to Rosetta rebuild and refine. Native structures are shown in blue. The top scoring Foldit predictions are shown in green. The best scoring Rosetta rebuild and refine predictions given the same starting model are shown in yellow. (a) This is the same puzzle as in Figure 4.7(a). The best scoring Foldit and Rosetta predictions both correctly fixed the register shift, but the best scoring Foldit prediction got it closer to the native. (b) This is the same puzzle as in Figure 4.7(b). Both predictions correctly buried the exposed isoleucine in the bottom right, but the best scoring Foldit prediction remodeled the loop more accurately. (c) The best scoring Foldit prediction for puzzle 987162 was able to place the helices and loops on the left side of the protein better than the best scoring Rosetta rebuild and refine prediction. (d) This is the same puzzle as in Figure 4.7(c). Both predictions correctly rotated the helix on the top left, but the best scoring Rosetta rebuild and refine prediction was unable to keep the bottom right helix in its proper place. (Figure from Cooper et al. [2010a])

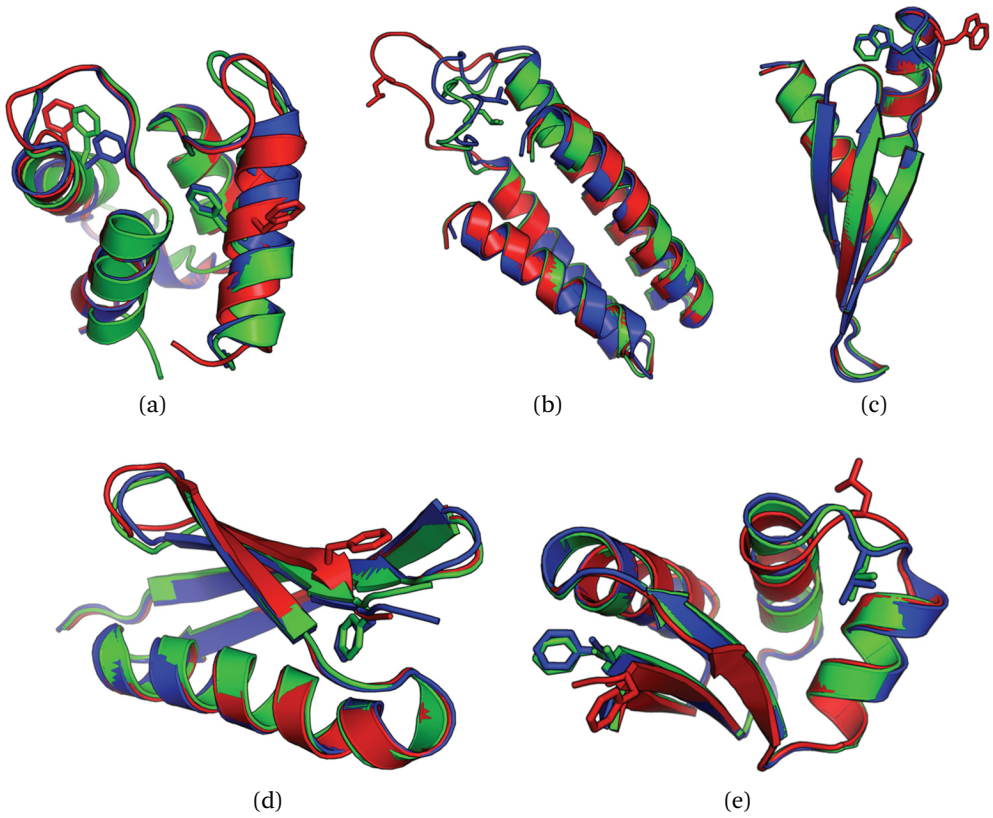


Figure 4.13 **Additional hydrophobic burial puzzles (*non-blind trials*).** Different hydrophobic examples. Native structures are shown in blue. Starting Foldit puzzles are shown in red. Foldit predictions are shown in green. (a) The right helix shows a successful helix rotation, but the helix on the left was not rotated far enough. (b) The entire red loop was correctly rebuilt to bury the leucine. (c) The exposed tryptophan was successfully buried. (d) The starting puzzle had a register shift and the Foldit prediction was able to flip and slide the beta strand. (e) This puzzle contained an exposed leucine between two helices as well as a register shift. The Foldit prediction was able to retrieve both native conformations. (Figure from Cooper et al. [2010a])

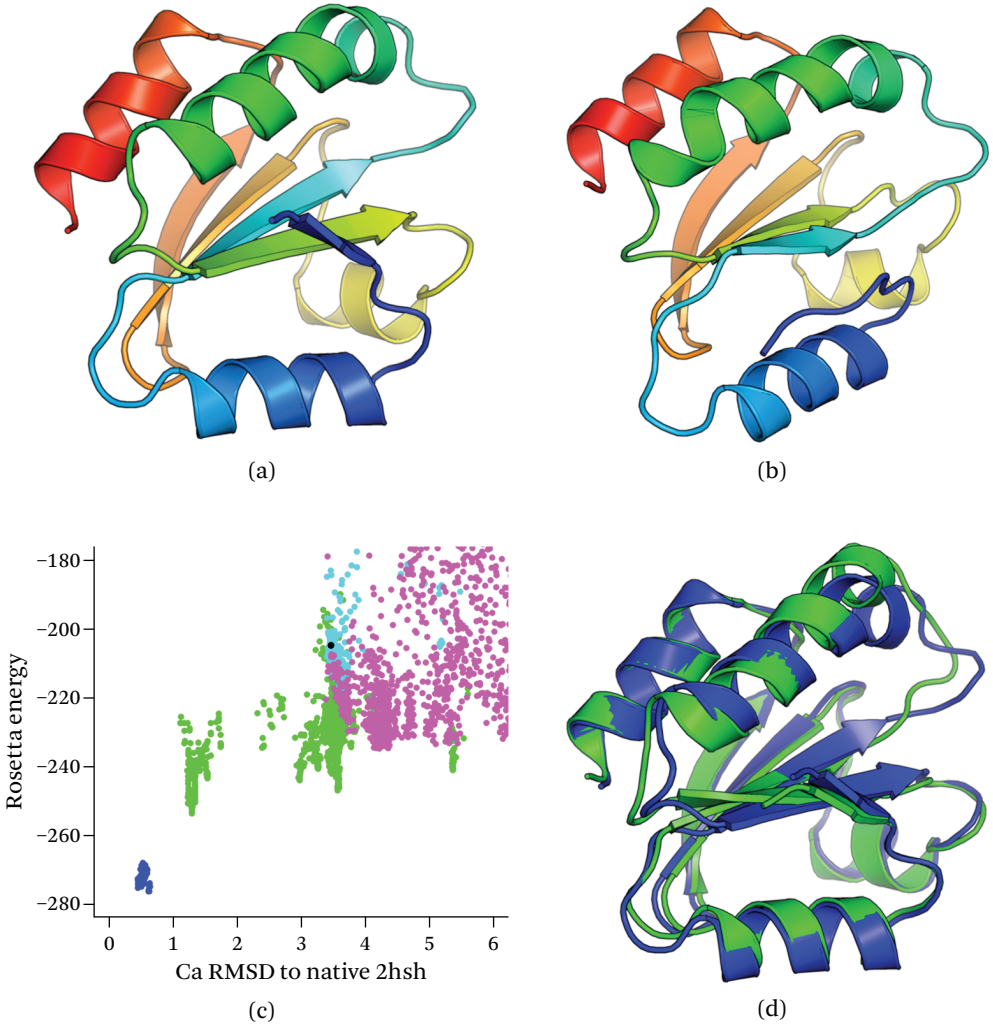


Figure 4.14 **First strand swap puzzle (*non-blind trial*).** The first strand swap Foldit puzzle, 986452, where Foldit players were able to outperform Rosetta's best automated method, rebuild and refine. (a) The solved native structure, 2hsh. (b) A Rosetta prediction which incorrectly swapped the cyan and yellow strands, this model was used as the starting model for Foldit puzzle 986452. (c) RMSD plot showing the starting Foldit puzzle (the black dot), Rosetta rebuild and refine predictions given the same starting model (in magenta), Foldit beginner predictions (in cyan), all Foldit predictions (in green), and relaxed natives in blue. (d) Superposition between the native in blue and the best scoring Foldit prediction in green. Unlike Rosetta, the top scoring Foldit players were able to correctly swap the strands. (Figure from Cooper et al. [2010a])

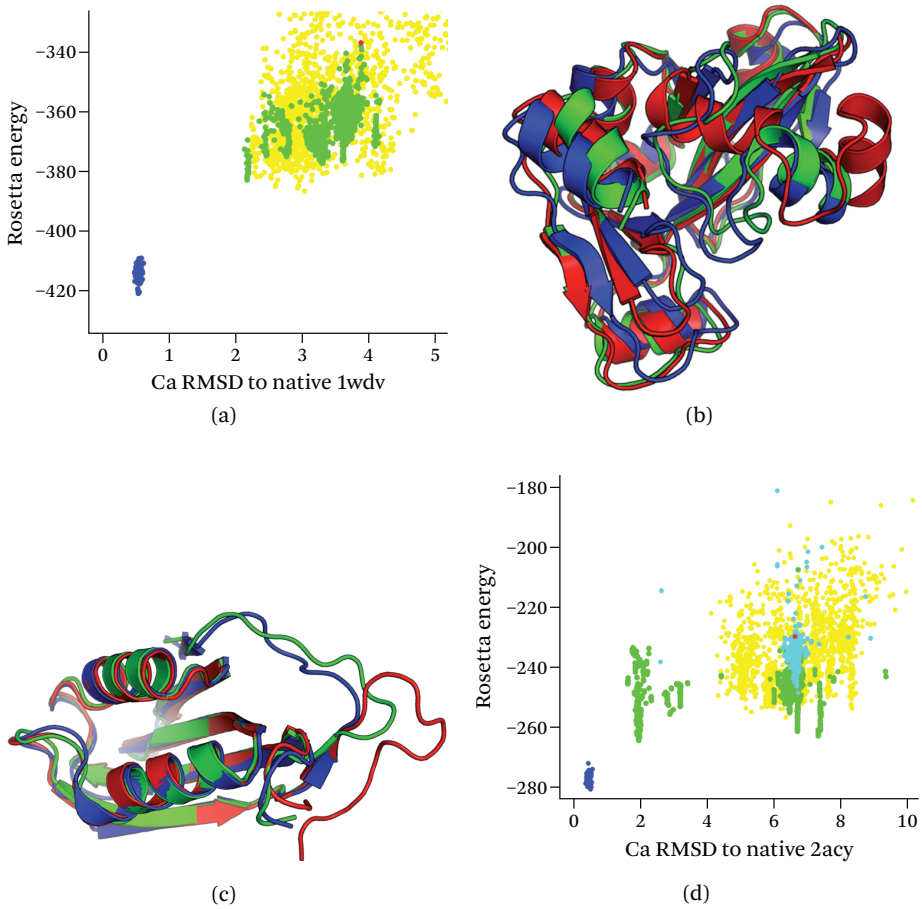
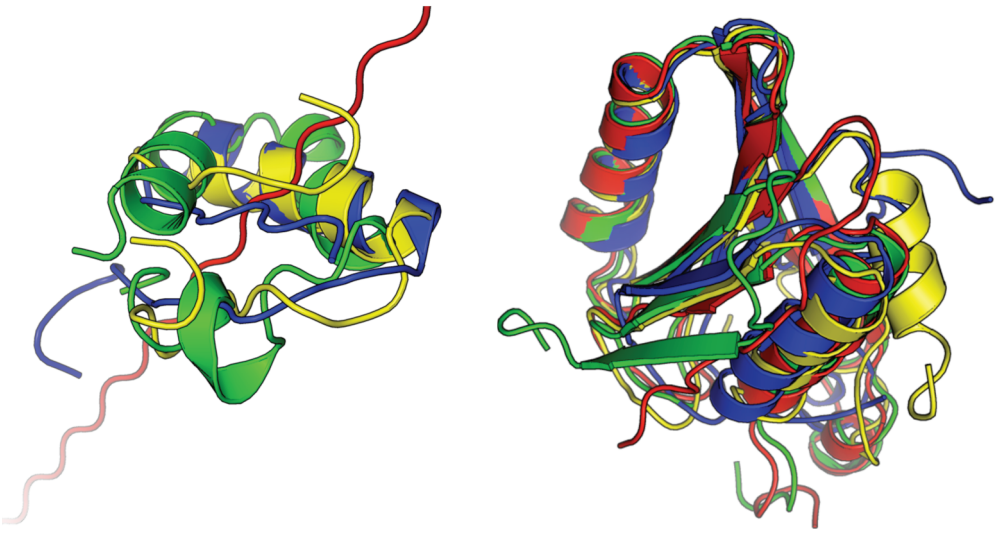


Figure 4.15 **Additional successful Foldit puzzles (*non-blind trials*).** More successful Foldit predictions. (a) RMSD plot for Foldit puzzle 985988. Starting puzzle shown as red dot. Relaxed natives are shown as blue cloud. Rosetta's rebuild and refine predictions are shown in yellow (note that the lowest scoring Rosetta predictions are over 3 Å away from the native structure). Foldit predictions are shown in green. (b) Top scoring Foldit prediction for Foldit puzzle 985988 (lowest green point in a). The native is shown in blue, the starting puzzle in red, and the best scoring Foldit prediction is shown in green. The very difficult region on the right of the protein, where the starting puzzle has two helices that are connected by a short loop, was completely rebuilt by this Foldit player and is very close to the native. (c) Top scoring Foldit prediction for Foldit puzzle 986127 (lowest green point in d). The native is shown in blue, the starting puzzle in red, and the top scoring Foldit prediction is shown in green. The C-terminus at the top of the protein, which was incorrectly placed in the starting Rosetta model (on the right in red), was correctly moved by this Foldit player and is very close to the native. (d) RMSD plot for Foldit puzzle 986127. Starting puzzle shown as red dot. Relaxed natives are shown as blue cloud. Rosetta's rebuild and refine predictions are shown in yellow (note that the lowest scoring Rosetta predictions are over 6 Å away from the native structure). Foldit beginner predictions are shown in cyan with all Foldit predictions in green. (Figure from Cooper et al. [2010a])



(a) (b)

Figure 4.16 **Rosetta outperforming Foldit players (*blind trials*).** Comparing Foldit players to Rosetta rebuild and refine. Native structures are shown in blue. Starting Foldit puzzles are shown in red. The best scoring Foldit predictions are shown in green. The best scoring Rosetta rebuild and refine predictions given the same starting model are shown in yellow. (a) "Freestyle" puzzle 986844 started as an extended chain (shown in red) and the top scoring Foldit solution was unable to capture the native fold. Rosetta rebuild and refine's best scoring solution had trouble with the termini, but was able to correctly fold the helix. (b) The starting structure for puzzle 986961 (shown in red) was the final CASD prediction submitted by the Baker group for CASD target AtT13, built using CS-Rosetta [Shen et al. 2008]. The best scoring Foldit prediction used the C-terminus to form a strand, similar to the starting structure. Rosetta rebuild and refine's best scoring solution was able to correctly move the C-terminal end to the other side of the helix. (Figure from Cooper et al. [2010a])

Table 4.4 gives the percentage of players who were able to successfully restructure select blind Foldit puzzles.

4.5.1 First Strand Swap Example

Previous to posting the strand swap puzzle shown in Figure 4.8, we had come across a Rosetta prediction that had incorrectly swapped two of the beta strands and subsequent calculations were unable to correctly swap them back. We posted this first strand swap puzzle (Figure 4.14), expecting it to be a challenging puzzle. Indeed, beginning Foldit players were unable to fix the strand swap and never got within 3 Å of the na-

Table 4.4 **Percentages of players successful.** For certain Foldit puzzles in the blind test set (Table 4.3), we identified specific scenarios that could be evaluated for success. We calculated the percentages of Foldit players (including beginners) who successfully fixed the problems in each of these puzzles. (Table from Cooper et al. [2010a])

Puzzle ID	Type of Restructuring Operation Scenario	Percent of Players Successful
987088	Register shift (Figure 4.7(a)	9%
987088	Remodeled loop (Figure 4.7(b)	7%
986836	Rotated helix (Figure 4.7(c)	19%
987145	Two remodeled loops (Figure 4.7(d)	5%
987076	Helix rotation and remodeled loop (Figure 4.7(e)	4%
986875	Strand swap (Figure 4.8)	8%

tive solution (cyan in Figure 4.14(c). We were pleasantly surprised to find that several Foldit players were able to correctly fix the strand swap, with Foldit solutions getting 1.1 Å from the native structure. The top scoring Foldit solution fixed the strand swap perfectly and got 1.3 Å from the native (Figure 4.14(d).

Since we did not have an introductory level teaching Foldit players how to swap strands, we posted the same Rosetta prediction again as a Quest to the Native puzzle with the native conformation (Figure 4.14(a) provided as a guide within the puzzle. With this particular Quest to the Native (Figure 4.4(d) players were able to learn the necessary Foldit tools required to swap strands and most players were able to get close to the native.

4.5.2 Player Contribution and Expertise

Looking at the set of 208 Foldit puzzles run at the time of the rebuild and refine comparison (mostly non-blind), 95% of the score improvements on most puzzles are done by less than 10 people, the median number being 5 people and the mean 6. However, these players are different from puzzle to puzzle; e.g., there are 72 distinct top players over these puzzles and 262 distinct players counting the top 3. So while no more than perhaps 500 people have really driven Foldit (at least in the very restricted sense of improvements to the score), the expertise acquired in the game appears to be diversified, instead of concentrated with a few individuals. This is likely related to the significant variation in human strategies that are required to solve different puzzles. We also note that advancement towards the eventual best solution in each puzzle cannot be analyzed just on direct score improvement, as the score does not consider many social aspects

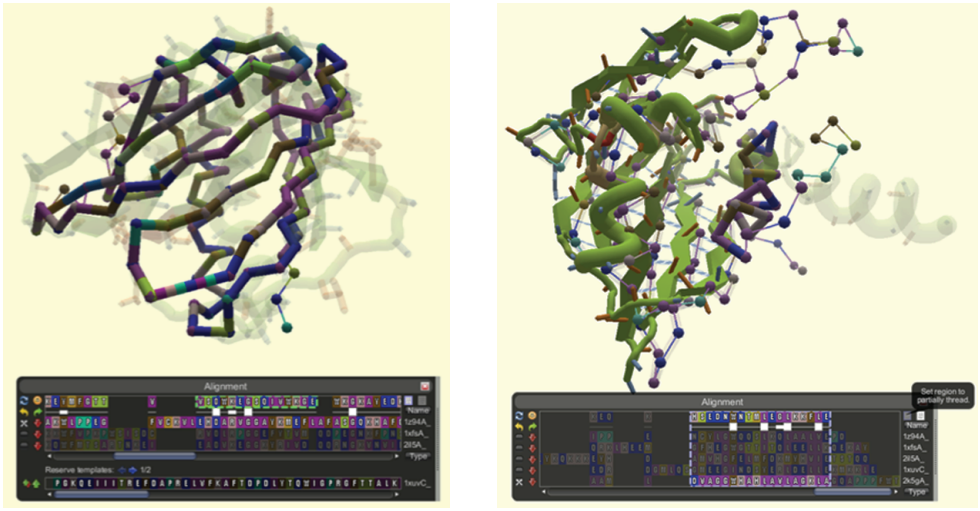
of the game. A false direction of one person may provide crucial insight to the eventual advancement of another person. Further formal analysis of collective expertise is required to fully understand this process. In addition, it is not the case that those players with little biochemistry experience are simply improving upon the solutions found by those with biochemistry expertise. In our survey of biochemistry background, we found that the top five responding soloists of all time (players that have reached top solutions without improving upon other players' solutions) have no more than a high school level of biochemistry experience. It is also possible for players with no biochemistry experience to rise through the ranks. A new player with no biochemistry background, who joined in the middle of the rebuild and refine trial set presented in this work, was able to progress to be the third place soloist overall.

From the puzzles in this set, the average amount of time spent per player on non-blind prediction puzzles open to all players was roughly 155 min. For all Quest to the Native puzzles, that average was 216 min., and for blind puzzles, the average was 249. There is also a series of "Beginner" puzzles intended to ease the transition from the introductory levels to the open science puzzles, which only new players are allowed to compete in. The average time per player on these puzzles was 29 min. Thus, it would appear that more experienced players spend more time per puzzle, and more time was spent on the blind puzzles than non-blind ones.

4.6 Alignment Tool and CASP9

Many real-world protein-modeling problems are amenable to comparative modeling starting from the structures of homologous proteins. To make use of homology modeling techniques in Foldit, we introduced a new capability called the Alignment Tool, which allows players to manually move alignments and thread their sequence onto the structures of known homologs (Figure 4.17(a). Players are also able to carry out partial threading, which allows threading only a specific region from one template, rather than the entire template, making it possible to combine different regions from multiple templates into a single hybrid structure (Figure 4.17(b). Players can also load in previously saved Foldit player solutions as templates by using template loading, which allows partial threading and hybridizing with their previous solutions.

Our aim was for Foldit players to use these new tools to solve real-world problems; the Critical Assessment of Techniques for Protein Structure Prediction (CASP) experiment was an ideal venue to test this. CASP is a biennial experiment of protein structure prediction methods where the amino-acid sequences of structures that are close to being experimentally determined—referred to as CASP targets—are posted for groups around the world to predict the native structure. Each group taking part in CASP is al-



(a) (b)

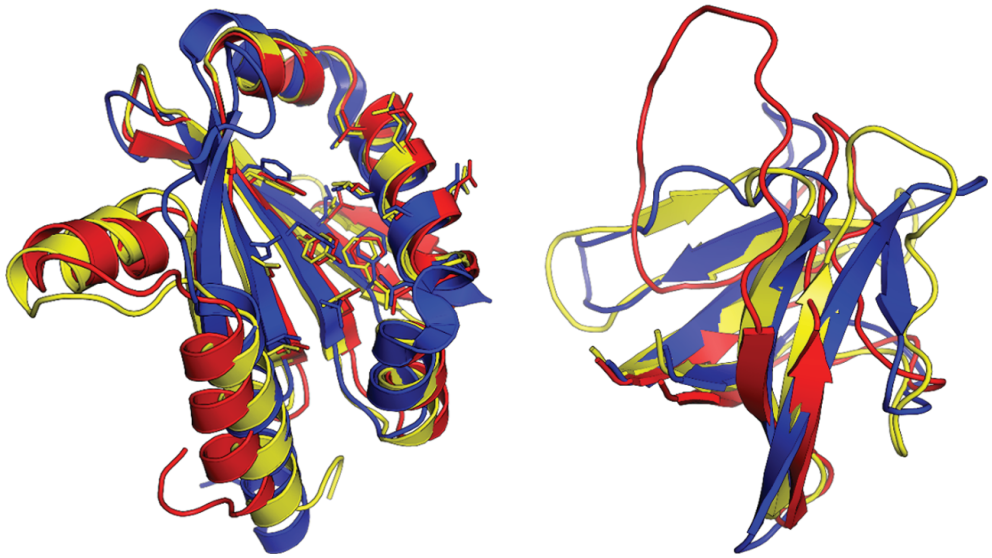
Figure 4.17 **Screenshots of the Foldit Alignment Tool.** The Alignment Tool allows Foldit players to load in different templates and manually move alignments. Players are then able to thread their sequence onto the structures of these known homologs. (a) When a template is selected, the aligned regions are represented as cylinders in the game while any unaligned regions are shown as spheres connected by lines; these graphical representations change in real time as players select residues and move the alignments around in the Alignment Tool. (b) During CASP9 Foldit players requested the ability to thread only a specific region from one template so partial threading was added to the Alignment Tool; this allows players to combine different regions from multiple templates into one hybrid model. (Figure from Khatib et al. [2011b])

lowed to submit five different predictions for each sequence. Foldit participated as an independent group during CASP9 and made predictions for the targets with fewer than 165 residues that the CASP organizers did not indicate as oligomeric. For targets with identifiable homologous protein structures—the Template Based Modeling category—Foldit players were given different alignments to templates predicted by the HHpred server [Söding et al. 2005] via the new Alignment Tool. Despite these new additions to the game, the performance of Foldit players over all CASP9 Template Based Modeling targets was not as good as the best performing methods, which made better use of information from homologous structures; extensive energy minimization by Foldit players perturbed peripheral portions of the chain away from the conformations present in homologs.

For prediction problems where there were no identifiable homologous protein structures—the CASP9 Free Modeling category—Foldit players were given the five Rosetta Server CASP9 submissions (which were publicly available to other prediction groups) as starting points, using the Alignment Tool. Here, all five starting models were available, allowing players to use partial threading to combine different features of the Rosetta models. In this Free Modeling category, some of the shortcomings of the Foldit predictions became clear. The main problem was a lack of diversity in the conformational space explored by Foldit players, because the starting models given to the players were already minimized with the same Rosetta energy function used by Foldit. This made it very difficult for Foldit players to get out of these local minima, and the only way for the players to improve their Foldit scores was to make very small changes ("tunneling" to the nearest local minimum) to the starting structures. However, this tunneling did lead to one of the most spectacular successes in the CASP9 experiment.

De novo structure prediction remains an exceptionally challenging problem, and very few predictions with atomic accuracy have been made in the history of the CASP experiment. For CASP9 target T0581, starting from an extended chain, the Rosetta Server, which carried out a large-scale search for the lowest-energy structure using computing power from Rosetta@home volunteers, produced a remarkably accurate model (Figure 4.18(a), compare red to blue). However, the server ranked this model fourth out of the five submissions. The Foldit Void Crushers group correctly selected this near-native model and further improved it by accurately moving the terminal helix, producing the best model for this target of any group, and one of the best overall predictions at CASP9 [Kinch et al. 2011] (Figure 4.18(a), compare yellow and blue). Thus, in a situation where one model out of several is in a near-native conformation, Foldit players can recognize it and improve it to become the best model. Unfortunately, for the other Free Modeling targets, there were no similarly outstanding Rosetta Server starting models so Foldit players simply tunneled to the nearest incorrect local minima.

The CASP9 Refinement category provides groups with the best prediction made for selected targets, and challenges groups to improve them further. Foldit participated in the Refinement category for all non-oligomeric targets. Many refinement targets at CASP9 were models created using Rosetta, resulting in a similar "tunneling" problem as with the initial Free Modeling predictions. Using the lessons learned from those targets, we tried presenting problems in a way that encouraged Foldit players to make more dramatic changes to the starting models. Initially, a strict RMSD condition was applied on refinement puzzles: the Foldit score would not count unless the prediction was different enough from the starting refinement model. However, Foldit players found it difficult to improve upon the starting model while improving the score because the Rosetta models had already been minimized using the Rosetta energy function.



(a) (b)

Figure 4.18 **Successful CASP9 predictions by the Void Crushers Foldit group.** (a) Starting from the fourth ranked Rosetta Server model (red) for CASP9 target T0581, members of the Void Crushers Foldit group (yellow) were able to bring it closer to the later determined crystal structure (blue). This model was the best-ranked prediction made by any CASP9 group for Free Modeling target T0581. (b) Starting from a modified Rosetta model built using the Alignment Tool (red), members of the Void Crushers Foldit group generated a model (yellow) considerably closer to the later determined crystal structure (blue). This was the best-ranked prediction made by any CASP9 group for refinement target TR624. (Figure from Khatib et al. [2011b])

For the very last CASP9 refinement target, TR624, Foldit players struggled with this same problem; the starting model was Rosetta-optimized and very few players were able to satisfy the RMSD conditions while at the same time finding lower energies.

Instead of imposing RMSD conditions, we decided to perturb the structure out of its energy minimum. We used the Alignment Tool to align the regions the CASP organizers identified as correct, and threaded the sequence onto the correctly aligned portions of the starting structure.

The unaligned portions were rebuilt randomly, initially with a poor energy, encouraging diversification of models in the incorrect regions, while maintaining the favorable interactions in regions known to be near-native. Using this modified model as a start to a new puzzle, Foldit players were able to sample closer to the native fold by

rebuilding several incorrect loops (Figure 4.18(b), compare yellow and blue). The submitted top scoring Foldit prediction by the Void Crushers group for this puzzle also used the Alignment Tool to partially thread one of their previous solutions and was the best ranked model for the most difficult refinement challenge in CASP9, TR624 [MacCallum et al. 2011].

4.7 Solution of Crystal Structure

Following the failure of a wide range of attempts to solve the crystal structure of monomeric M-PMV retroviral protease by molecular replacement, we challenged players of the protein folding game Foldit to produce accurate models of the protein. Remarkably, Foldit players were able to generate models of sufficient quality for successful molecular replacement and subsequent structure determination. The refined structure provides new insights for design of anti-retroviral drugs.

The most important problem solved by Foldit players to date came after CASP9 and involves the M-PMV (Mason-Pfizer Monkey Virus) retroviral protease. Retroviral proteases (PR) play a critical role in viral maturation and proliferation and are the focus of intensive anti-viral drug development work [Mastrolorenzo et al. 2007]. All previously determined crystal structures of retroviral proteases show the biologically active homodimeric form [Wlodawer and Gustchina 2000]; prevention of PR dimerization has been proposed as a mechanism for disruption of PR activity [Koh et al. 2007] and a drug design avenue for antiretrovirals. The retroviral protease of M-PMV, an AIDS-causing monkey virus, crystallizes as a monomer, but despite the availability of several crystal forms researchers for over a decade have been unable to solve the structure by molecular replacement (MR) using either homodimer-derived models, or an NMR model of the protein monomer [Veverka et al. 2003]. A recent approach using density- and energy-guided structure refinement [DiMaio et al. 2011] was also unsuccessful at determining a solution, despite a success rate of over 50% in cases where similarly good homologous template structures were available. Of the unsolved cases presented in DiMaio et al. [2011], this was the only one suitable (non-oligimeric, and of small enough size, 114 residues) for use in Foldit.

To determine if human intuition could succeed where automated methods had failed, we challenged Foldit players to build accurate models of M-PMV PR starting from the NMR coordinates (which had failed in MR tests).

When the three-week competition concluded, we screened the top-scoring Foldit models using Phaser [Mccoy et al. 2007] to determine whether any were of sufficient quality for MR. Remarkably, despite the complete failure of all previous approaches,

several solutions by the Foldit team Contenders produced phase estimates that were good enough to allow a rapid solution of the crystal structure.

We provided Foldit players with the ten different NMR models which all scored poorly using Rosetta's energy function so that players would not be trapped in local energy minima, and included all ten NMR models as templates in the Alignment Tool. The improvement in model accuracy by the Foldit Contenders group is illustrated in Figure 4.19(a). As game play progressed (x axis), model accuracy and suitability for MR, as assessed by the Phaser log-likelihood gain (LLG) (y axis) increased, with several notable jumps (arrows). Figure 4.19(b–d) illustrates some of the breakthrough models produced by Foldit players. Foldit player spvincent (yellow) used partial threading with the Alignment Tool and quickly improved the starting NMR model (red) to have much better agreement with the afterwards determined crystal structure (blue). Another teammate, grabhorn, was able to improve spvincent's model (magenta), particularly in the core of the protein, and mimi was able to generate an even more accurate model (green) by correctly tucking in the loop at the top left. The LLG of this model was high enough to allow its unambiguous identification as a likely MR solution among the vast number of Foldit models and standard autobuilding and structure refinement methods showed within hours that the solution was almost certainly correct.

Using the Foldit solution, the final refined structure was completed a few days later. Of particular interest in this monomeric retropepsin structure is the molecular surface that normally forms the dimer interface in homodimeric retroviral protease molecules. There is a considerable backbone rearrangement in this area in which a flap curls over the half-active site in the monomer (Figure 4.20). Additionally, the N- and C-termini are totally disordered highlighting the absence of a dimerization interface. These features provide exciting opportunities for retroviral drug design, including anti HIV drugs; compounds that bind to the surface formed in the monomer but not the dimer should shift the equilibrium in favor of the former, which is catalytically inactive.

4.8 Conclusion

This chapter has discussed the framework's application to the prediction of natural protein structures. We showed that game players, in certain cases, can predict structures more accurately than computational methods and perform well against other methods in worldwide competition. We also showed that this framework has allowed the solution of a previously unsolvable open problem in structural biochemistry.

In proof of concept tests, Foldit players were able to solve protein structure refinement problems in which backbone rearrangement was necessary to achieve the burial

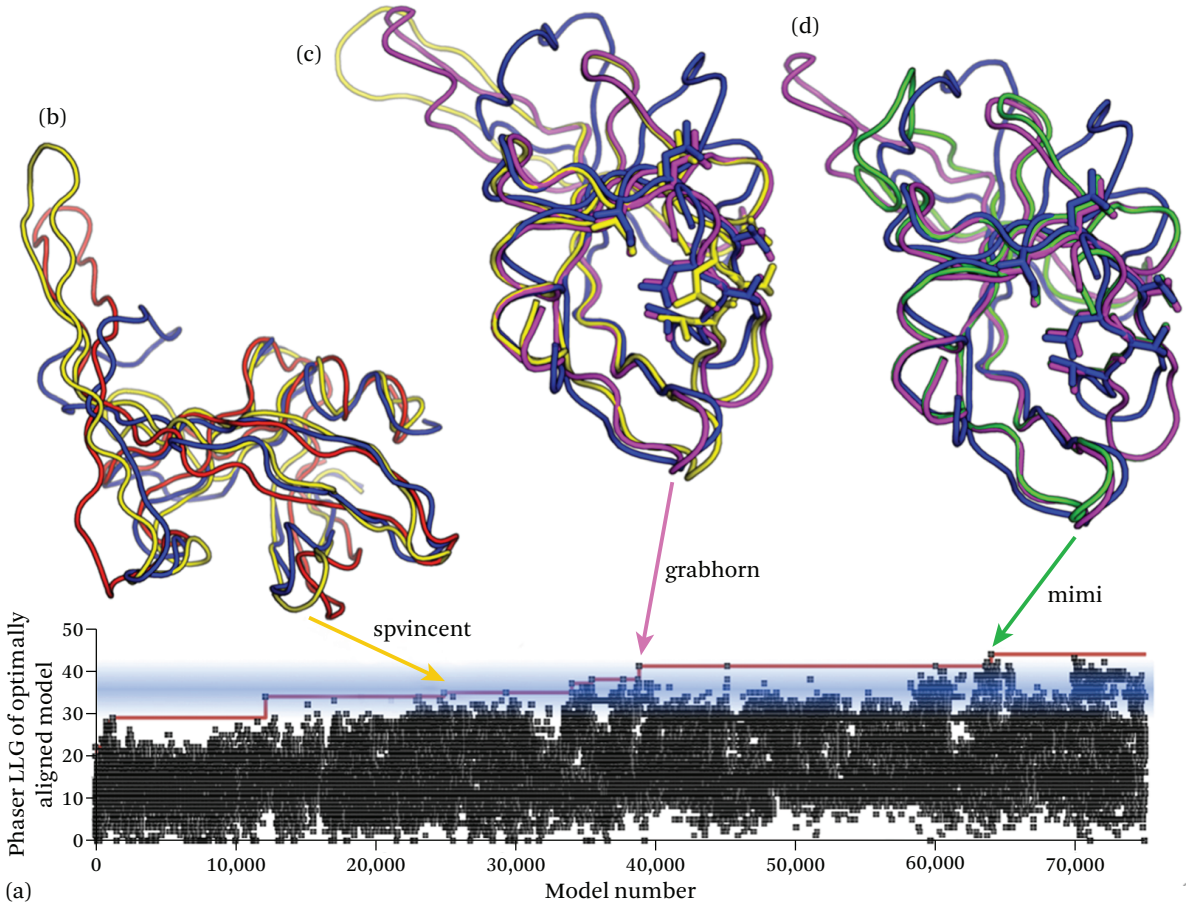


Figure 4.19 **M-PMV retroviral protease structure improvement by the Contenders Foldit group.** (a) Progress of structure refinement over the first 16 days of game play. The x-axis shows progression in time, while the y-axis shows the Phaser log-likelihood (LLG) of each model in a near-native orientation. To identify a solution as correct by molecular replacement using Phaser, the model must have an LLG better than the best random models. The distribution of these best random predictions is indicated by the intensity of the pale blue band. (Because almost all the models are too poor to allow correct placement in the unit cell, Phaser LLGs are calculated after optimal superposition of each model onto the solved crystal structure and rigid-body optimization.) (b) Starting from a quite inaccurate NMR model (red), spvincent generated a model (yellow) considerably more similar to the later determined crystal structure (blue) in the beta-strand region. (c) grabhorn generated a model (magenta) considerably closer to the crystal structure than spvincent's model. (d) mimi generated a model (green) still more accurate (in the loop region at the top left) that provided an unambiguous molecular replacement solution which allowed rapid determination of the ultimate crystal structure (blue). (Figure from Khatib et al. [2011b])

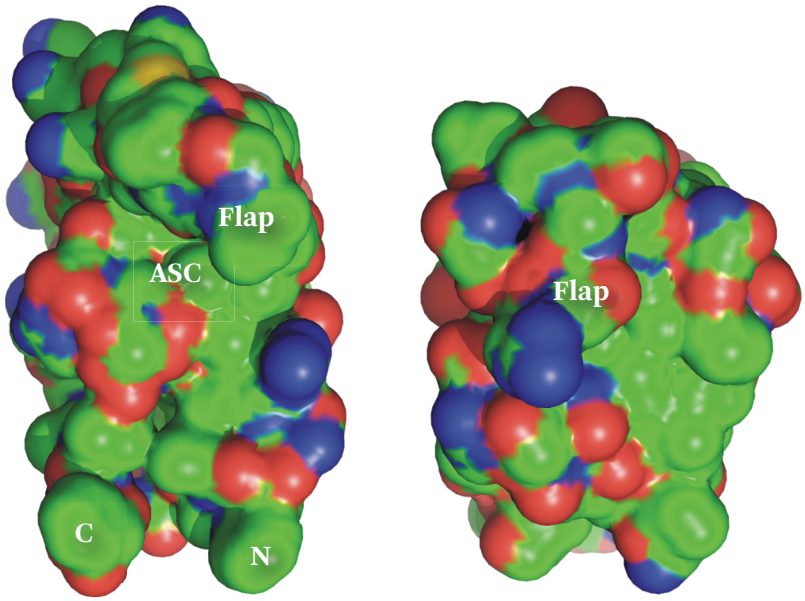


Figure 4.20 **CPK representation of retropepsin surface.** (a) The surface of HIV-1 PR protomer extracted from the dimeric molecule (PDB 3hvp), as seen from the direction of the removed dimerization partner. (b) M-PMV PR monomer shown in the same orientation and scale. In this view, the N- and C-termini (missing in M-PMV PR) are at the bottom and the flap loops are at the top. The active-site cavity (ASC) is clearly seen between the flap and the body of the HIV-1 PR molecule. In M-PMV PR the cavity is completely covered by the curled flap. (Figure from Khatib et al. [2011b])

of hydrophobic residues [Cooper et al. 2010a]. Foldit players have also performed well in specific refinement cases in CASP competitions [Cooper et al. 2010b].

The critical role of Foldit players in the solution of the M-PMV PR structure shows the power of online games to channel human intuition and three-dimensional pattern-matching skills to solve challenging scientific problems [Khatib et al. 2011b]. While much attention has recently been given to the potential of crowd sourcing and game playing, this is the first instance we are aware of in which online gamers solved a specific longstanding scientific problem. These results indicate the potential for integrating video games into the real-world scientific experimental process: the ingenuity of game players is a formidable force that if properly directed can likely help solve a wide range of scientific problems.

5 Protein Design

5.1 Introduction

This chapter discusses applying the framework to take advantage of player creativity to design proteins. These designed proteins do not exist in nature and can be experimentally tested for effectiveness. The creativity of players is useful for coming up with structures that are both novel and effective. The structures produced by gameplay can aid in the successful synthesis of designs that are more effective than previous structures.

The design of novel biological molecules can lead to cures for diseases, more efficient biofuels, and a better understanding of living organisms. The design of such complex biochemical molecules such as proteins is a challenging problem. However, these problems are primarily structural in nature; they depend on the spatial relationships between the atoms involved and how they interact. Thus, we believe that there is the potential for anyone can make a contribution to these problems using innate spatial reasoning ability. We discuss Foldit, an online game that allows players to compete by folding proteins, and the iterative approach we took to add protein design into the Foldit framework. In order to solve real biochemistry problems using a video game, we took an iterative approach to creating the tools and visualizations necessary for protein design in Foldit by starting with protein design scientists and improving the game based on their feedback. Once Foldit design features were released we incorporated the results that came back from the players into this iterative loop.

In our approach, we allow players to design novel structures through the game and evaluate the results both computationally and experimentally. We discuss early results for two cases of protein design: redesigning a protein's central core to improve stability, and redesigning a protein's exterior regions to improve binding of a small molecule. In both cases iteration allowed us to improve the initial results. Iterative approaches have been used for protein design automation [Dahiyat and Mayo 1996]; however, we are iterating in a process with video game players helping to provide the designs.

In spite of its difficulty, protein design offers many benefits to society. Proteins can be vaccines and inhibitors against disease, enzymes to catalyze useful reactions, and DNA manipulators. Coming up with new designs is often a combination of massive

computational resources and scientists' input. The human aspect of this problem is essential: scientists must use their knowledge and spatial reasoning ability for design to be successful. As these design problems involve the interactions between biochemical structures, spatial reasoning is used to determine the three-dimensional relationships of the novel protein. We believe that because all humans have an innate spatial reasoning ability, there is potential for anyone to reason about the structures contained in design problems. However, those without biological training will not have the knowledge of scientists. Instead, we will leverage scientists' knowledge and use that to shape the rules of a game, in order to lead those without training to successful designs.

5.2 Framework Extension

5.2.1 Foldit

Foldit's initial development was focused on protein structure prediction—players would try to find the structure of naturally occurring proteins. To allow players to design novel proteins, new interactions needed to be added. Foldit's interface is designed around different modes, so a new "Design Mode" was added, in which interactions relevant to design could take place. In this mode, players can manually mutate between individual amino acids, by selecting one from a pie menu Figure 5.4, effectively changing the sequence of the protein. Players can also manually insert or delete amino acids, changing the length of the protein and its sequence. Players are also given the ability to allow the computer to automatically select the best scoring amino acids where they choose. On top of this, the original prediction based interactions and optimizations remain present, so players can search for the best scoring structure for the new sequence.

Puzzles required new parameters for design in order to constrain what types of design could occur, in order to make the results relevant to the problem at hand. As not every amino acid can always be mutated, a puzzle can specify which amino acids can and cannot change from the starting structure. Further, puzzles can specify where amino acids can be inserted and deleted, and what the minimum and maximum length of the sequence can be. Areas than can be designed appear brighter than the surrounding ones. Also, some areas can be locked by the puzzle, preventing their degrees of freedom from changing. The purpose of this is twofold: on one hand, it prevents regions of the protein whose structure is critical to maintain from being modified, and on the other, focuses players' attention on the regions where their changes will be the most useful. Such locked areas appear grayed out.

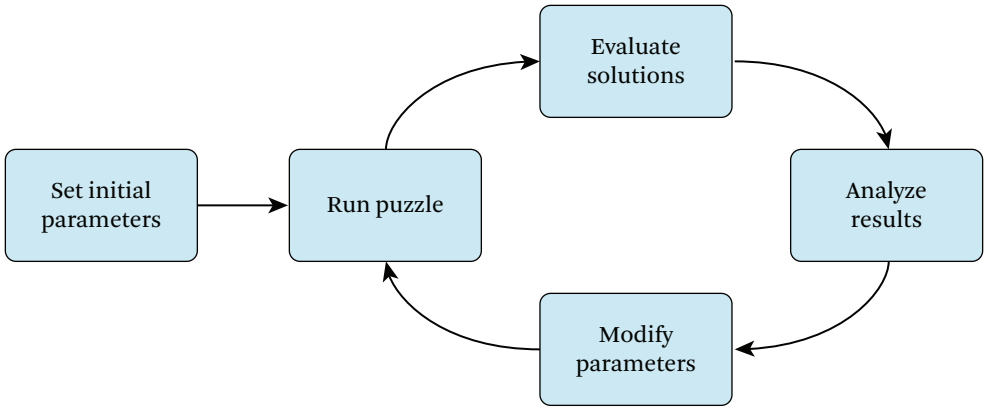


Figure 5.1 **An overview of the iterative approach taken for protein design.**

5.2.2 Iteration Strategy

Our basic iterative approach is as follows. First, a puzzle's parameters are set, and the puzzle is posted. Once the puzzle is closed, the resulting player designed structures are gathered and evaluated. This evaluation could take a variety of forms. Using the information received from the evaluation, the parameters for a new puzzle can be set to improve the results or try a different approach. This may involve anything from adjusting constants to adding new features to the game itself. The process can be repeated until satisfactory results are gained. An overview of the process is given in Figure 5.1.

The first steps are to set the initial puzzle parameters and run the puzzle. These would be decided on by the scientist whose problem was to be used in the puzzle. There is some goal that is desired to be achieved by the results of the puzzle, for example, the protein should become more stable, or it should bind another molecule more tightly. The parameters should be set with this in mind. As mentioned previously, puzzles typically run for about a week, although more that one puzzle may be online at a time. We have heuristically found this to be a reasonable length of time for puzzles to run, however, further analysis could determine a better length.

The next step is to evaluate the solutions. Given the large number of raw structures that can be generated, and the fact that resources for evaluation (such as scientist time or wetlab equipment) may be limited, it is necessary to reduce the these to a manageable number. We have found some methods of sorting and filtering solutions to be generally useful. We sort the solutions by their overall score, then go through the solutions in order filtering many out. First, we filter by sequence; discarding solutions

whose sequence has been seen before with better score. Second, we filter by structure, discarding solutions whose structures are very similar to higher scoring structures; we do this by ordering the structures by score, then proceeding to filter out any structures too close to a higher scoring structure. Rather than filtering, it may also be useful to cluster solutions based on structure; however, we have found the filtering approach to work as it is fast and flexible. Methods of sorting and filtering specific to each puzzle are also useful. The remaining solutions can then be evaluated more closely. The purpose of this evaluation is to determine if the solutions have achieved the initial goal set out in the puzzle. The evaluation could be carried out in a number of different ways, from qualitative examination by a scientist, to computational verification, to experimental synthesis.

Next, the results are analyzed. In analyzing the results, we wish to determine why the goal was or was not met. We have found that often the goal is not met due to a puzzle's parameters being too open; that is, the players are able to change from the starting structure too much. Often a designer will look at solutions and see that the players have made some change that is not acceptable to the design's goals because there were implicit constraints on the design that were not enforced by the puzzle's parameters.

With the results of the analysis, new parameters for the puzzle are chosen. The new puzzle can then be run and the process repeated.

5.3 Science Transfer

Several modifications needed to be made to the base tools and visualizations to support protein design. We needed to have finer grained control over which parts of the protein could be changed to map design problems into the game and focus player effort where it would be most useful. We needed to prevent certain regions of the protein from changing, and also specify which regions were and were not designable. We also needed to give players the tools necessary to make design-related changes to the protein.

5.3.1 Visualizations

We made changes to the visualization of the protein to communicate to the players the different properties of the protein with respect to how they could change it, shown in Figure 5.2. Designable regions, where it was possible to change amino acid or insert or remove residues, are shown brighter, to draw attention to them. Locked regions, which could not be changed at all, are faded out to make them less noticeable. The remaining regions, which behave as they normally would in a prediction puzzle, look

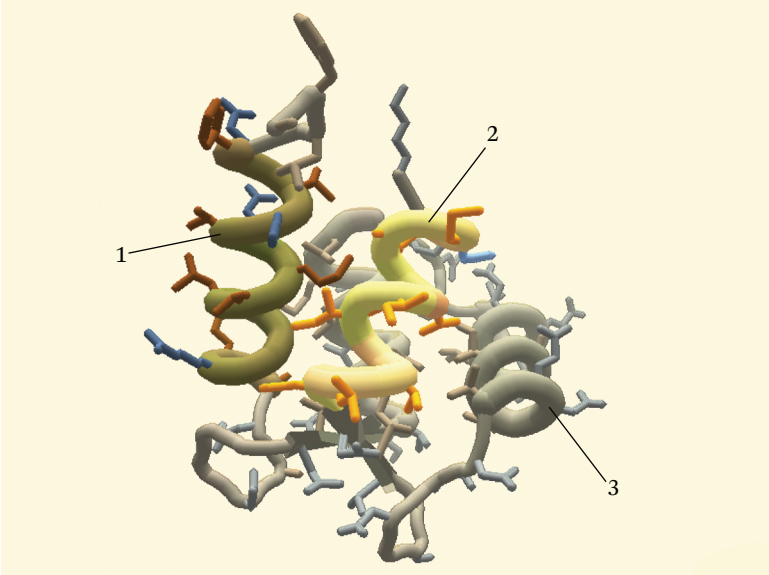
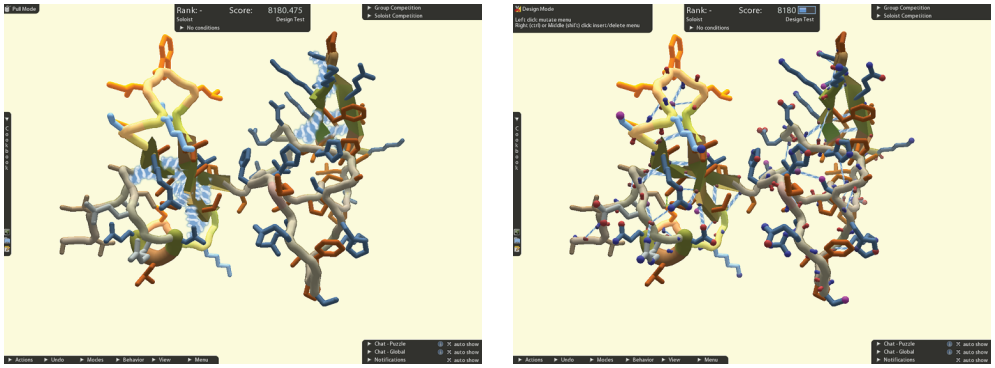


Figure 5.2 **Different properties of residues in design puzzles.** Regions of normal brightness behave (arrow 1) as they would in prediction puzzles; they can move but not be designed. Brighter regions (arrow 2) can also be designed. Regions that are faded out (arrow 3) are locked and cannot be moved or designed.

the same. Unfortunately, it is not possible to indicate designability by just recoloring the sidechain, as glycine could not be seen without the addition of some geometry; we had also initially tried to recolor sidechains that could be designed as green, this however obscured the hydrophobicity coloring.

Another aspect of visualization that is particularly important for protein design is hydrogen bonding. The creation of hydrogen bonds to stabilize structures is often critical to protein design. For prediction puzzles, we only showed hydrogen bonds when they connected sheets, showed all hydrogen bonds the same thickness, and did not show donor or acceptor atoms (which can create hydrogen bonds).

Changes in hydrogen bonds, which were motivated by protein design, are shown in Figure 5.3. For design puzzles, we added extra view options that allowed all hydrogen bonds to be shown, based on if they are connected to sheets, helices, loops, or non-protein structures. Similarly, there are options to show donor and acceptor atoms to help in the creation of hydrogen bonds. The strength of each hydrogen bond is shown in its thickness.



(a) (b)

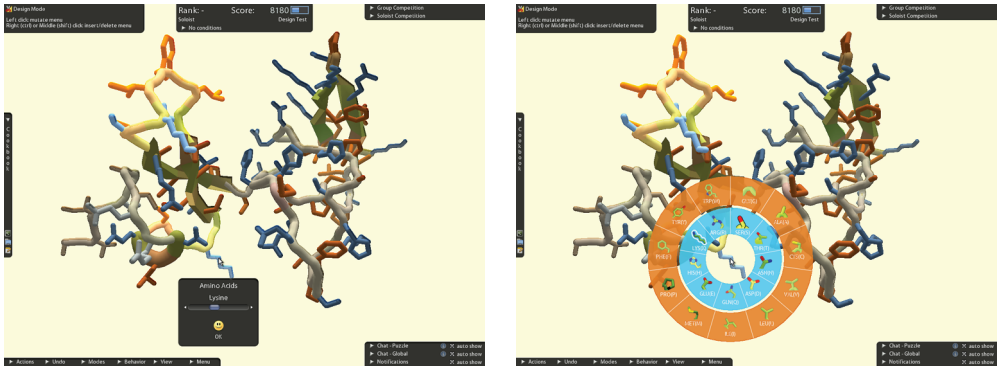
Figure 5.3 **Example of the refinement of a visualization: hydrogen bonds.** (a) Initially hydrogen bonds were only shown on sheets, and fixed-width. (b) The current version allows players to show different sets of bonds, show atoms which can form these bonds, and their width is proportional to their strength.

5.3.2 Tools

The primary new tool to allow design in puzzle was to allow players to change amino acids on a protein. This was accomplished through the addition of a new Design Mode. When in design mode, left clicking the protein will bring up the mutate menu. This menu went through redesigns before being released, shown in Figure 5.4. Initially, it was a simple slider with amino acid names. This design was refined over time. The final design has many improvements: the pie menu has pictures to show the geometry and atomic properties of each amino acid; the amino acids are separated into two rings for hydrophobics and hydrophilics; as the menu is large, there is an opening in the middle to see the selected residue through and the menu will fade out when the mouse is not over it.

Right clicking brings up a simple pie menu for adding or deleting residues.

We added an Auto Mutate tool, similar to the Shake tool, which repacks but also redesigns residues. When we initially implemented the tool, we simply allowed residues to be designed. However, running this on the entire protein at once proved to be too slow due to the combinatorial nature or designing. We therefore changed the algorithm to repack and redesign smaller neighborhoods of the protein iteratively. This allows the player to see partial results and also to cancel the optimization early.



(a) (b)

Figure 5.4 **Example of the refinement of a tool: the mutation menu.** (a) The first iteration was simply a slider that allowed the player to choose an amino acid. (b) The current version is a pie menu.

At this point we also added specific tools for running Wiggle on sidechains and backbone independently.

5.3.3 Conditions

In order to ensure that player designs had specific desirable properties, we added *conditions* to the game. Conditions represent qualifications that a player's solution needs to meet before it will receive credit and register on the leaderboard.

Conditions are general and can vary from puzzle to puzzle as needed. For example, they may be limiting the number of mutations, requiring specific atoms to hydrogen bond, or limiting how much a structure can move from its starting configuration.

5.4 Introductory Levels

In order to introduce the new concepts relevant to design, we added several new introductory levels. See Figure 5.5 for an example. These levels are only reachable after completing the basic prediction levels, so that players who reach them are somewhat familiar with the protein manipulation tools in Foldit before beginning design. Later, we added levels to further introduce ligand-related concepts. Information about the design levels can be found in Table 5.1.

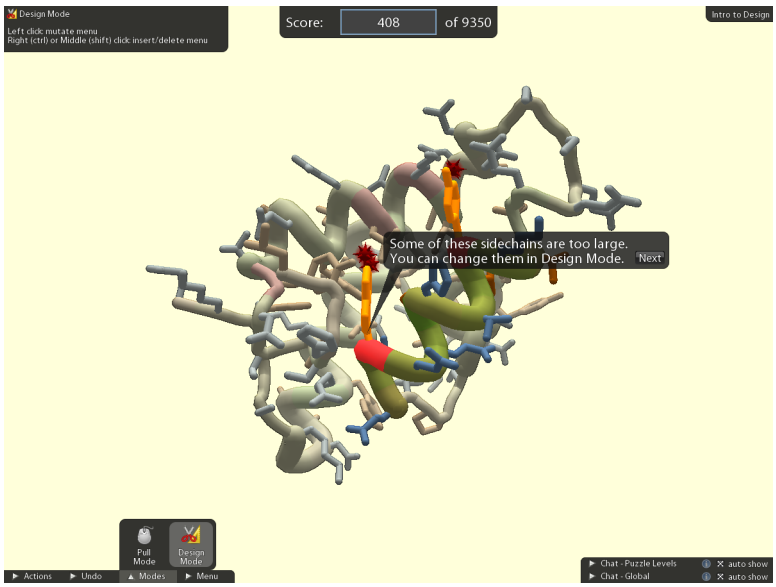


Figure 5.5 **A design introductory level.**

Table 5.1 **Design introductory levels.** Given are their order, name, and the main concepts introduced

Number	Title	New Concepts
5-1	Intro to Design	Design mode, individual residue mutation, areas locked by puzzle
5-2	Swappin' Sidechains	
5-3	Mass Mutate	Automatic mutation
5-4	Insertion and Deletion	Insertion and deletion of residues
6-1	Ligand Debut	Ligands
6-2	Ligand Constraints	Catalytic constraints

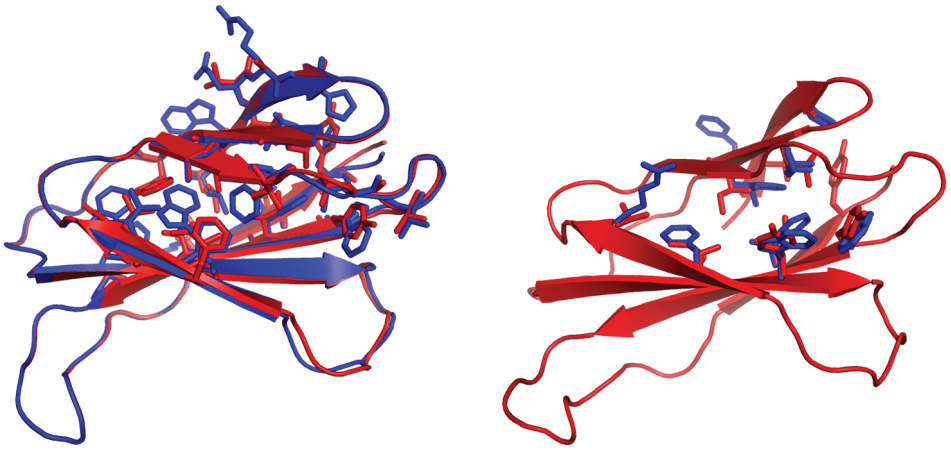
5.5 Examples

5.5.1 Fibronectin

The purpose of the fibronectin puzzle series was to allow players to redesign the core of the protein to improve stability. Fibronectin has been used as a scaffold for protein design, with the potential for hosting antibody loops [Olson and Roberts 2007]. These puzzles were based on the structure of the 10th domain of fibronectin found in PDB structure 1FNF, which is composed only of sheets (and loops). A number of loops on the structure are generally modified for binding, and it has been reported that some of these loops are structurally important to the stability of the fibronectin. If the core of the fibronectin could be better stabilized on its own, the molecule could then tolerate more changes in the binding loops, thus uncoupling these important loops from the rest of the protein. By uncoupling these loops, it would be easier to experiment with different loop mutations, allowing for more diverse experimental selection of binders.

The first puzzle posted in the series was 178: Core and Tail Design (ID 986568). The puzzle was setup to allow players to mutate in the core of the protein while allowing the parts of the backbone that were not part of the binding loops to move as well. One end of the protein could be extended by several residues, but we found that but most players did not alter the length. In modeling the interactions between the residues, this puzzle used a set of weights for the energy function terms derived specifically for all-beta proteins [Hu et al. 2008] with a reduced repulsive term. This setup allows players to sample different sequences that can achieve higher packing density in the core. The original sequence in the core was also substituted with alanines so that the player solutions were not influenced by the native core; any potential biases towards the native sequence is encoded in the backbone geometry alone. This bias encoded by the native scaffold was further reduced by allowing the backbone torsional angles to change by the players.

Upon completion of the first puzzle, we filtered the results by rescoring the models against the standard forcefield used for Rosetta prediction and the void volume metric that measures packing in the core [Sheffler and Baker 2009]. We found that players packed the core with significantly higher number of aromatic amino acids than the native. To adjust for clashes introduced when using the weights derived from fixed backbone structures on flexible backbone designs, we sorted the models on the two measures described above and picked candidates for experimental testing if they ranked highly in both measures. One design was selected for experimental testing, shown in Figure 5.6a. Although it also had an abundance of aromatic amino acids when compared to the wildtype, it was well-packed and not clashing. It scored reasonably well,



(a) (b)

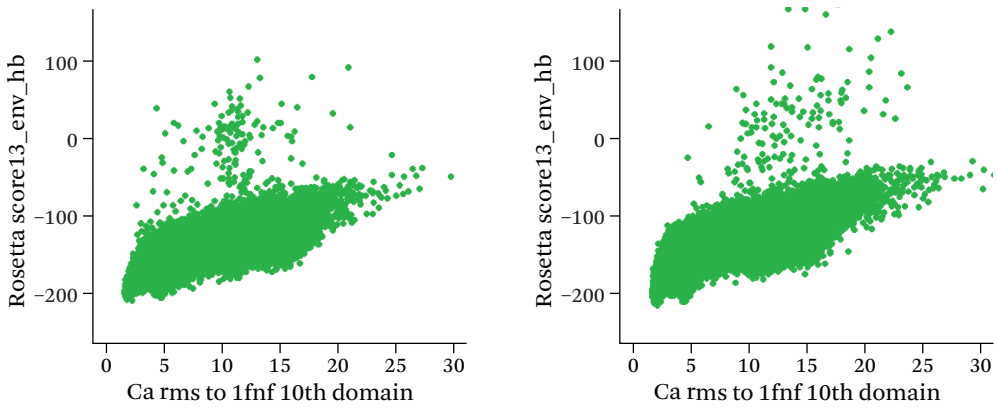
Figure 5.6 **Player fibronectin designs.** (a) The player design from the first round fibronectin puzzle that was tested in the wetlab is shown in blue. The native 1FNF structure given to as a starting fibronectin scaffold players for fibronectin design puzzles, shown in red for comparison. Amino acids which could be mutated are shown in more detail. (b) Individual amino acid point mutations from the second round shown on one structure, in blue. Starting scaffold shown in red for comparison.

and that meant the extra atoms from mutating residues to aromatic amino acids were making favorable contacts.

To verify the feasibility of the most promising structure, we ran two massive prediction runs on Rosetta@home, one with the designed sequence, and one with the native sequence as a control. The results are shown in Figure 5.7. Given that the designed sequence folded in the prediction runs, we went forward with synthesizing the sequence in the wetlab. Although computationally validated, it turned out that this design was not experimentally soluble—likely due to the number of aromatic residues put in the core. Although the residues are relatively neutral for their presence on beta-sheet structures, the design might have become overly insoluble such that its folding kinetics ware disrupted by molecules aggregating in solution.

To celebrate the first player-designed structure to be tested in the wetlab, a trophy was donated to the designing player by 3D Molecular Designs.[1] The trophy is shown in Figure 5.8.

1. http://www.3dmoleculardesigns.com/; last retrieved May 2014.



(a) Foldit sequence

(b) Native sequence

Figure 5.7 **Computational verification of the fibronectin design.** The player designed Foldit sequence produces an energy landscape similar to the native sequence.

Figure 5.8 **Trophy for the player who created the tested 1FNF design.**

With this information, we posted another puzzle, 223: Core and Tail Design 2 (ID 987028). To address this issues from the previous puzzle's designs, we reduced the degrees of freedom players could sample by (1) restricting the backbone to their starting position with a harmonic potential, (2) scoring the puzzle with the standard Rosetta parameters instead of the one with reduced repulsive forces, (3) constraining the sequence space to only allow mutations to amino acids observed for this particular fold at a significant level, and (4) disallowing extension of the end of the protein. We still presented the starting structure with an alanine only core. The puzzle was set up to test the various constraints we can applied to the puzzle, and observe responses from designs generated by players. We expected to have designs that are more native-like and thus will still be soluble when produced in the wetlab. Since sequences were still allowed to drift away from the native if a better alternative can be found, more stable sequences could still be designed.

We again filtered the results of the puzzle. Looking at the top designs, the changes made were much more conservative, changing only a few residues from native per design, shown in Figure 5.6b. Due to this fact, we searched the literature to see if similar mutations had been made. We found that similar mutations had been previously tested [Cota et al. 2000] and thus decided that it was not necessary to experimentally test the player designs.

5.5.2 Diels-Alder

The purpose of this series of puzzles was to redesign surface loops of a computationally designed enzyme in order to improve the molecular contacts between the protein and the small molecule it had been designed to bind. The enzyme of interest in this case was designed to bind two independent small molecules and catalyze the formation of a carbon-carbon bond between them in a chemical process known as the "Diels-Alder" reaction. The Diels-Alder reaction is a cornerstone in organic synthesis (it was awarded the 1950 Nobel Prize in Chemistry), but to date no naturally occurring enzyme has been proven to catalyze a biomolecular Diels-Alder reaction. Unfortunately, while showing significant Diels-Alderase activity, the relative catalytic efficiency of this enzyme is approximately 10,000-fold lower than what is commonly observed for enzymes found in nature. In order to improve this enzyme further we believe additional contacts need to be made to the small molecules the enzyme is binding. Additional contacts with the small molecules will potentially allow this enzyme to stabilize the "transition state" of the reaction, resulting in enhanced catalysis.

In this case, the scientists were using a custom built version of the Foldit game that allowed them to load their work into the game. The scientists are able to use all the same intuitive tools as players for their work, and when desired, they can post what

they are working on as a puzzle for Foldit players. This can help the scientists to get new ideas for structures, narrow down the range of structures to consider for testing, or confirm the feasibility of certain structures. Upon receiving solutions from the players, the scientists can then analyze player-created structures and resume work with this new data. For this Diels-Alder experimentation cycle all design work was done in Foldit, either by scientists in their custom version of the game or by players in the standard version.

We integrated Foldit and the players into the cycle of computational design and experimental testing, running several iterations of combined scientist and player computational design, validated by experimental enzyme assays.

In order to introduce new contacts we have decided to borrow some tricks from nature. When naturally occurring enzymes alter binding it is often done through the modification of surface loops that interact with the small molecule. Therefore we decided to try and engineer surface loops in order to introduce additional protein-ligand (in this case the ligand is the two small molecules the enzyme is binding) contacts. To engineer the surface loops we used the Foldit interface and an interactive player-scientist refinement cycle. For the first puzzle that was released we allowed the ligand to move in addition to several surface loops, but we kept the core of the protein fixed in space. Players were then encouraged to modify the loops that had been allowed to move in order to "cover the ligand". Upon review of the first round of solutions from the game *192: Cover the Ligand* (ID 986712), we found that many players had not only altered the loops, but had moved the ligand into physically unrealistic positions and conformations. Therefore we decided to release a second round of puzzles to the Foldit players. In *195: Cover the Ligand 2* (ID 39082) we allowed the same surface loops to be engineered by players, but this time kept the ligand fixed in space. Of the resulting structures submitted by players we clustered them based on sequence similarity and structural similarity and manually analyzed a representative structure from each cluster, on the order of 50 structures.

Upon further inspection of the selected structures, we found that while the location of the loop was such that the new contacts would be made to the ligand, the amino acid sequence used to create the loop conformation was unlikely to form a stable structure. In particular, there appeared to be an excess of the amino acid glycine, which is inherently highly flexible and leads to disordered protein structures. So while players had made models with several new contacts to the ligand, due to the use of glycines it was highly unlikely that these structures would actually remain in the conformation as modeled by the players. Therefore, the next task was to take these designs and replace the glycines with new amino acids such that the new ligand contacts and general loop structure form the Foldit player would remain intact, in addition to introducing

new protein-protein contacts that would increase of the probability of having the loop structured as modeled when synthesized in the wetlab.

From the structures originally selected, a set of three player backbone topologies were found in which we were able to remove the excess glycines without significantly disrupting the new contacts that were made to the ligand. Variations of these three backbone topologies were designed by scientists by replacing amino acids and varying the backbone length. These were synthesized in the laboratory and experimentally tested for their ability to bind the desired ligands. After several rounds of authoring libraries and experimental testing based on these backbone topologies, it was found that some designs were able to increase overall enzyme activity. These designs had placed a helix near the ligand.

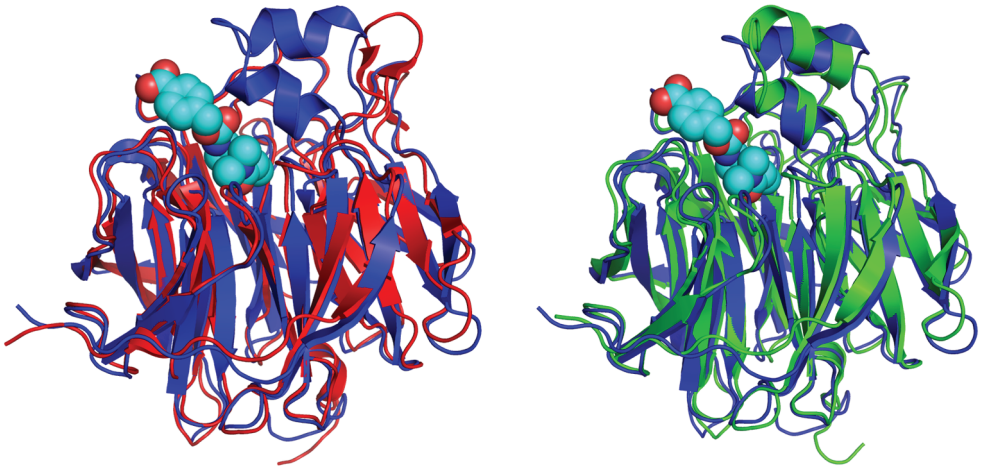
Although these designs had improved over the original, and the helix appeared to be a positive change, there was a loop neighboring the helix that looked like it could be improved. Thus we posted a followup puzzle, *366: Back Me Up 1* (ID 988581), requesting that the players redesign that loop into a more stable helix-loop-helix motif. The resulting player-produced structures were promising and able to create the desired motif. Again, the most promising backbone structures identified by manual examination were used to author libraries of structures of varying length and sequence for experimental validation, and the best were found to give an increase in activity.

We then iterated with several more puzzles, *395: Improve the Loop* (ID 989188), *401: Quick Loop Puzzle* (ID 989312), and *418: Back Me Up 2* (ID 989669), giving players good starting backbones from previous rounds to verify that the structure was correct, fine-tune it, and use results to author libraries for testing. In the end, the most successful design created by players and scientists that was tested is shown in Figure 5.9a. This final design consisted of a 13 residue insertion, the creation of two new helices, and a roughly 20-fold increase in activity over the starting enzyme. A solved crystal structure (shown in Figure 5.9b) confirmed the form of the computationally designed structure.

5.6 Conclusion

This chapter has discussed the design of novel proteins using the framework. We have shown players' creativity useful for coming up with ideas for novel and effective structures. We have shown that players, working together with scientists, are able to design protein structures with increased effectiveness over previously existing structures.

The framework developed with Foldit was able to be extended from an initial focus on protein structure prediction to protein design. The addition of new tools, visualizations, and levels, game players the ability to not just look for protein structures that exist in nature, but design entirely new ones. An iterative process of refinement was



(a) (b)

Figure 5.9 **Foldit player and scientist Diels-Alder co-design.** (a) The most successful Diels-Alder backbone designed by players, used by scientists to generate a library for experimental testing in the wetlab. The player design is shown in blue, and the ligand is shown with spheres. The starting scaffold shown in red. The modification of the loop in the upper right allows additional contacts with the ligand. (b) Comparison of the designed structure (in blue) with a solved crystal structure (in green). Note the similarity of the two structures, particularly in the designed helical regions.

used to improve the game and the protein design puzzles themselves to move towards more useful results.

The puzzles run to redesign the core of fibronectin showed that players were able to come up with designs that were feasible to be tested in the wetlab and similar to those that had been tested experimentally before. Through the process of refining puzzle constraints the players were able to come up with designs that were more likely to work.

The successful redesign of the Diels-Alder enzyme by Foldit players and scientists shows the potential for integrating video game players into the scientific discovery process and the experimental process of scientists. Again, the iterative process of refining puzzles was able to guide players toward more successful protein designs. The drastic departure from the starting scaffold by players indicates that players have the potential to try fearless designs to find truly novel protein structures.

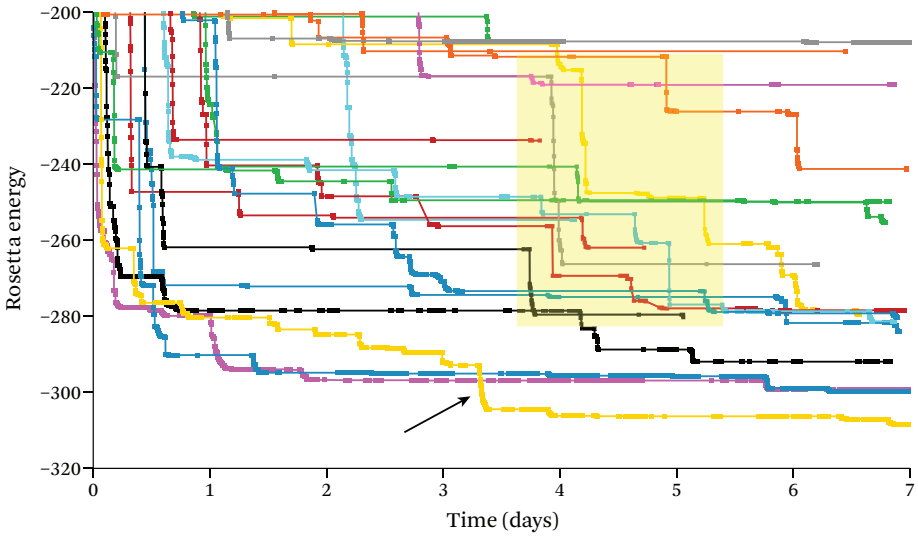
6 Protein Structure Refinement Algorithms

6.1 Introduction

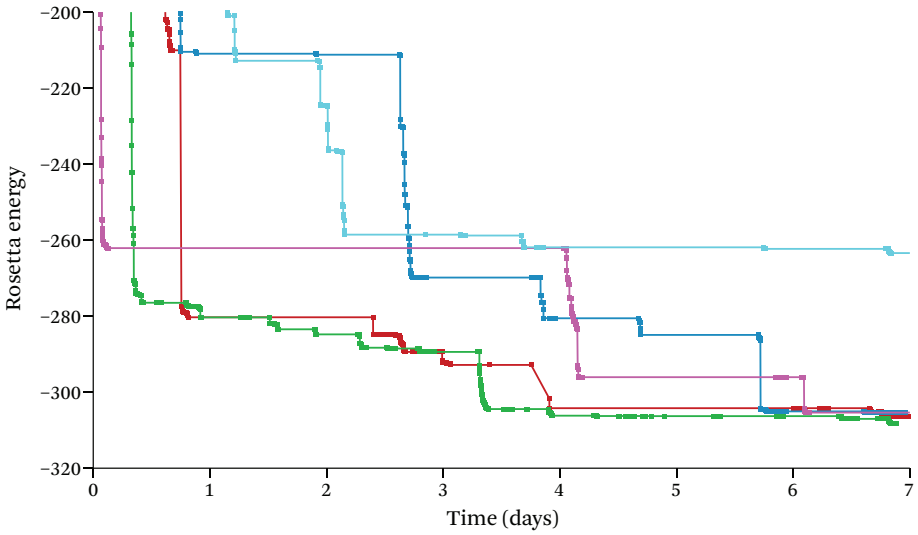
This chapter discusses the use of the framework to allow players to codify, automate, and share their expertise in the form of gameplay macros. Through social evolution of these macros, players develop specialized algorithms with a variety of purposes. In some cases, these algorithms surpass the effectiveness of scientist-developed algorithms.

As interesting as the Foldit predictions themselves is the complexity, variation, and creativity of the human search process. Foldit gameplay supports both competition and collaboration between players. For collaboration, players can share structures with their group members to work serially, and help each other out with strategies and tips through the game's chat function, or across the wiki. The competition and collaboration create a large social aspect to the game, which alters the aggregate search progress of Foldit and heightens player motivation. As groups compete for higher rankings and discover new structures, other groups appear to be motivated to play more (Figure 6.1a), and within groups the exchange of solutions can help other members catch up to the leaders (Figure 6.1b).

Humans use a much more varied range of exploration methods than computers. Different players use different move sequences, both according to the puzzle type and throughout the duration of a puzzle (Figure 6.2a). For example, some players prefer to manually adjust sidechains; some will forego large amounts of continuous minimization at the beginning of a puzzle, but increase it as the puzzle progresses; and some prefer a more direct approach and use more rubber bands when the puzzle begins from an extended chain. Within teams, there is often a division of labor; some players specialize in early stage openings, others in middle and end game polishing. Our informal investigation revealed a fascinating array of thought processes, insights and previously unexplored methodologies developed solely through Foldit gameplay. We have summarized several player-generated algorithms from an informal survey of top players in Table 6.1.



(a) Puzzle 987034 groups



(b) Puzzle 987034 group 985857

Figure 6.1 **Competition and collaboration.** (a) A chart of the lowest energy found by each group for puzzle 987034. The y-axis is Rosetta energy, and the x-axis is time from the puzzle start in days. The thicker line denotes when a member of the group was actively playing. Notice how the breakthrough of the cyan group (denoted by the arrow) during the third day sets off a chain reaction of energy improvements (shown in the yellow box) and how there is a frantic increase in playing just before the puzzle expires. (b) A similar plot for each individual player in group 985857 for the same puzzle. Note how players with worse energy are able to quickly catch up to teammates with better energies. (Figure from Cooper et al. [2010a])

Figure 6.2 **Player move preferences.** (a) Different Foldit players take different approaches to solving the same problem. Each circle represents the move type frequencies used in the top solution produced by each player in different time frames: the inner denotes the first hour, the middle denotes the first day, and the outer denotes the puzzle's entire duration. Each color represents a different type of move that can be made in the game. The left column reflects player move types for puzzles that start relatively close to the native topology. The right column reflects player move types for puzzles that start from a fully extended conformation. Each row represents a different Foldit player. Each player's preferred move types across each puzzle class are distinct from one another, yet a player's preferences are similar for both classes of puzzles. Also note that the move preferences change over the lifetime of a puzzle; local minimize is heavily preferred by the end of puzzles but not by all players at the beginning. The move types preferences are very different from Rosetta's current best automated protocol, rebuild and refine, shown in (b). (Figure from Cooper et al. [2010a])

Table 6.1 **Summary of player-developed algorithms.** (Table from Cooper et al. [2010a])

Category	Summary
Sidechain centric	Force core sidechain into new rotamer. Shake remaining sidechains. Unfreeze sidechain, wiggle, shake, wiggle.
	Move sidechains, automatically readjust the backbone.
	Sidechain flipping. Turn repulsive way down, then shake, then increase to 1 and wiggle/shake. Alternatively, reduce repulsive and wiggle, then increase and shake/wiggle. "Compress" recipe.
Backbone centric	Push, shake, wiggle,shake, wiggle until score stabilizes. Then rebuild plus bands.
	Reduce repulsive, shake, increase repulsive, wiggle. Twist and straighten ss elements. Script for locally optimizing 10 residue chunks. Rebuild all loops at least once.
	Use tweak tool to flip sheets with repulsive reduced, then rebuild end loops.
	Put in bands between sidechains, then wiggle and shake. Vary the repulsive strength. Wiggle internal segments with ends frozen.
	Drag backbone into voids based on positions of sidechains that could fill voids. Use bands to drag backbone, rebuilds to reposition, then shake and wiggle for cleanup. In the end game, move sidechains, optimize, then sometimes replace with original.
	Put bands in every Nth residue with recipe/ reduce repulsive. Wiggle. Remove bands, increase repulsive. Shake. Wiggle. Identify high energy sidechains, replace with original sidechain conformations individually. Basic idea is to get new sidechain conformations by overcompacting the protein, then to mix these with original sidechain conformations guided by the per residue energies.
Problem centric	Search for exposed hydrophobics. Push protein around, focusing on backbone, to get more compact and bury hydrophobics. Then wiggle sidechains, then wiggle backbone, then wiggle all. Then global shake. Then perturb sidechains, tweak helices. Compact protein using bands. For refinement, change secondary structure, rebuild, use bands.

In addition, any expertise that these players have developed is through the score function, the Foldit interface, and the aggregate knowledge of the Foldit player community. As we have shown, very few of them have a biochemical background. Thus, the process of expertise development in Foldit is radically different from that of traditional biochemical research: it emerged through free-form collective exploration within the game framework, rather than through lectures and book reading [Ericsson 2009].

As games grow in complexity, gameplay needs to provide players with powerful means of managing this complexity. One approach is to give tools for automation to players. This gives players the opportunity for customization and allows them to approach the game at a higher level. World of Warcraft[1] allows macro commands and also a rich scripting language for creating addons. Second Life[2] has a scripting language that allows users to define the behavior of objects. Final Fantasy XII[3] introduced the gambit system, which allowed players to automate many of the details of battle.

In the context of automation, we would like to investigate several questions. Can players formalize elements of their strategies? What effects to sharing and other social elements have on strategy development? Can sharing automated strategies help to spread expertise to new players? And can we use player strategies to improve automated methods?

The *Foldit cookbook* was developed to aid players in systematizing their strategies and integrating them into the social environment of a scientific discovery game. In this chapter we show the potential of automation in a social context for propagating the expertise of top Foldit players and increasing the overall collective problem solving skills of the predominantly non-scientist Foldit player population. In addition, we discuss our experience deploying the cookbook to the Foldit community at large, present analyses of two data sets gathered from players who used the cookbook, and conjecture directions for future improvement of in-game automation in scientific discovery games.

6.2 Related Work

Programming generally requires sophisticated skills and training. Extensive prior work has presented new programming languages to make the task easier and require less training. Among them, visual programming languages [Boshernitsan and Downes 2010] have been popular, particularly in educational contexts. They often take the form

1. http://us.battle.net/wow/en/; last retrieved May 2014.
2. http://secondlife.com/; last retrieved May 2014.
3. http://www.finalfantasyxii.com/; last retrieved May 2014.

of assembling blocks of code, such as in Scratch [Maloney et al. 2010] and StarLogo TNG [Wang et al. 2006]. Alice [Conway et al. 2000] is a visual programming language designed to teach students programming through storytelling. Kodu [MacLaurin 2009, MacLaurin 2011] is a visual programming language designed for young children to design and program video games. The Foldit cookbook design faces the same challenges in supporting user programming and uses similar visual programming language techniques. The recent language Toque [Tarkan et al. 2010] is particularly interesting for its use of cooking metaphors, which the Foldit cookbook also employs.

Analysis of data from online games is an expanding field, giving insights into the dynamics of online games and the communities around them. Recently, Lewis [Lewis and Wardrip-Fruin 2010] gathered a large data set on World of Warcraft by crawling web pages. This data was used to answer questions about the game world, such as which classes level faster or die more often. Williams [Williams et al. 2008] was given access to a large amount of data on EverQuest 2, which was used to discover the profiles of online gamers. In this chapter, we perform data analysis gathered from Foldit players using the cookbook.

How users share their software customization and collaborate in problem solving has also been studied in other domains. Mackay [Mackay 1990] studied patterns of sharing customizable software through a study of users at two research sites working with two different kinds of customizable software. Three groups of sharers were found: the experts who created customization from scratch, the translators who created simplified and more task-specific versions of the expert customizations, and the rest of the organization who adopted customizations from the translators. Berlin and Jeffries [Berlin and Jeffries 1992] studied interactions between experts and apprentices in learning and collaborative problem solving. One primary strategy found in their work was "copy and experiment": the apprentices find something similar to what they want, copy it, and then try to customize it to their specific needs. In our Foldit cookbook, the sharing activities occur in a much larger scale and in a much more distributed setting. Our analyses discover some of these same behaviors in scientific discovery games and also make new observations about group-based sharing and ratings.

6.3 Overview

6.3.1 Cookbook

Soon after Foldit's release, the players created a wiki.[4] One section of the wiki was devoted to player strategies; players began requesting the addition of automation tools

4. http://foldit.wikia.com/; last retrieved May 2014.

(a) (b)

Figure 6.3 **Screenshots of the cookbook and recipe editor for a GUI recipe.** The cookbook interface is shown in (a). Recipes are divided into sections based on whether they have been shared. When the mouse hovers over a recipe name, buttons for running, editing, and deleting appear. The GUI recipe editor is shown in (b). The visual blocks that make up the recipe take up the main section of the window. Buttons for saving, loading, sharing, and other options are along the bottom. (Figure from Cooper et al. [2011])

so that they could more easily carry out their strategies. We also intended to infer strategies from the Foldit players and use them to improve fully automatic approaches. Rather than performing machine learning on the playtraces of Foldit players, we decided that the players themselves would likely be much better at systematic abstraction of their strategies. For this reason we added the *cookbook* to Foldit. The cookbook allowed players to write, share, and run *recipes*, which were automated version of their strategies.

The initial interface for writing recipes was a simple block-based visual programming interface, illustrated in Figure 6.3b. Using this interface, a recipe is created by adding *steps* from a pulldown menu. Steps include game moves such as shaking, wiggling, adding bands, freezing, or restoring saved solutions. Steps can be deleted or reordered as desired. Each step may have some number of *ingredients* which need to be specified. These can include the pieces of the protein to apply the step to or the number of iterations to execute an optimization. It is worth noting that this language does not have support for conditions or loops. Recipes created with this interface are called *GUI recipes*.

After the addition of the cookbook with GUI recipes, players additionally requested the expressive power of a full scripting language. We therefore added an alternative text-based interface, using the Lua scripting language.[5] The base Lua language was

5. http://www.lua.org/; last retrieved May 2014.

augmented with custom functions to execute game moves and query game state. The Lua interface has many more available moves and the ability to control the flow of the recipe, allowing much more expressive recipes to be written. Recipes created with this interface are called *script recipes*.

Players are able to upload their recipes to the game servers to share with other players. They can share a recipe globally with all players or only with players in their group. The Foldit website has an interface for players to search for shared recipes. Each recipe has a webpage with information about the recipe, including its title, description, author, and comments. Each recipe additionally has a five-star rating. Players can rate a recipe on its webpage and are also reminded to rate, if they have not yet, when running a recipe in-game. If a recipe has been created by modifying a previous recipe, the recipe it is derived from (its parent) is listed on the webpage; any derived recipes (its children) are also listed. There is a button on the webpage that will add the recipe to the player's cookbook. The player who shared a recipe can make new revisions of it after it has been uploaded. These revisions will be considered the same recipe, and only the newest revision can be downloaded.

While playing Foldit, a list of the recipes in a player's cookbook are given in a collapsible sidebar in the game, shown in Figure 6.3a. As an introduction for players, the cookbook comes with four example GUI recipes and a button that links to a video introducing the cookbook. There is also documentation available for the steps and function calls, as well as text bubbles that appear explaining each step the first time it is used. The player can easily run, edit, or delete recipes from the cookbook. While running, the cookbook is replaced by text that gives the progress of the current recipe. Individual steps within the recipe can be canceled, or the entire recipe can be canceled.

6.4 CASP9 Analysis

To understand the social aspects of the cookbook, we analyzed data gathered from the puzzles run for CASP9. CASP is a biannual event in which teams compete to most accurately predict protein structures. Foldit ran a series of CASP9 related structure prediction puzzles over a period of several months, from May 6 to August 19, 2010. Basic information about the puzzles, players, and recipes used can be found in Table 6.2. In order to ignore recipes that were used a very small number of times, our CASP9 analyses consider only the recipes that were run at least 10 times.

6.4.1 Recipe Sharing

As Foldit players are able to share recipes with other players, we were interested in observing the patterns of use of recipes written by other players. This would allow us to determine if, and how, players were sharing recipes and using shared recipes.

Table 6.2 **Overview of data from the CASP9 data set.** (Table from Cooper et al. [2011])

Puzzles	56
Recipes	5488
Recipes run $\geq$ 10 times	1139
Recipes run $\geq$ 100 times	233
Recipes run $\geq$ 1000 times	26
Recipes run $\geq$ 10000 times	1
Total times recipes were run	158682
Players	3590
Players who ran a recipe	771
Players who wrote a recipe	568
Recipes written during CASP9	5202

In order to understand how players use shared recipes, we analyzed the patterns of use of recipes written by players themselves and those written by other players. We considered all players that either used a recipe during CASP9 or wrote a recipe that was used. For each of these players we computed the following statistics: total number of recipes used, total number of recipes written, number of times the players used recipes written by others, and number of times other players used recipes written by the player.

This led us to identify two distinct classes of recipe users: those who mostly used recipes that they wrote, and those who mostly used recipes written by others. We calculated what percentage of total recipe uses by each player were recipes written by that player. Figure 6.4 shows the histogram of the usage of a player's own recipes by 10% bins.

The large number of players in the leftmost bin correspond to players who primarily used recipes written by other players. The gray area identifies the number of pure consumers, players who wrote no recipes at all. Combined, these 246 players used recipes over 36,000 times. It is worth noting that 13 of the pure consumers account for more than half of these uses.

The rightmost bin corresponds to players who mostly used recipes they wrote themselves. Looking more closely at players who ran only their own recipes, Figure 6.5 shows the number of times that recipes were used by their authors, broken down by their sharing status. Clearly, these players tend to have an arsenal of recipes that they keep to themselves.

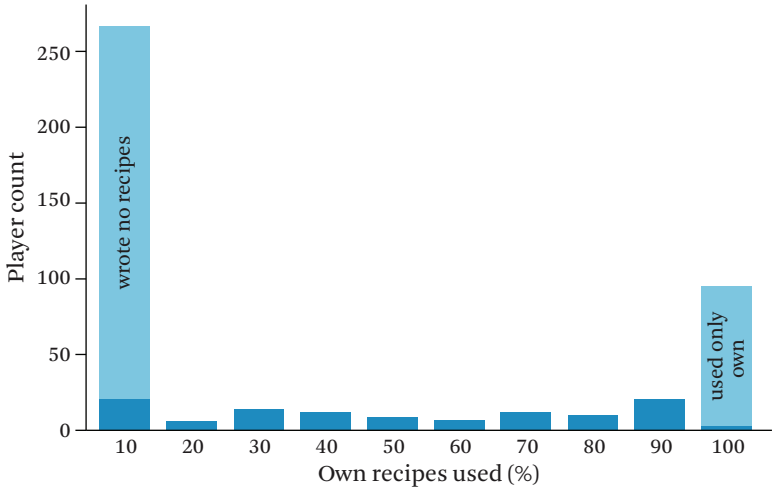


Figure 6.4 **Histogram of recipe users by the percentages of recipes used that they wrote themselves for recipes in the CASP9 data set.** The lighter bar on the left shows players who used only other players' recipes; the presence of this bar indicates that the expertise of players who write recipes can be used by players who do not write recipes. The lighter bar on the right shows players who used only their own recipes. (Figure from Cooper et al. [2011])

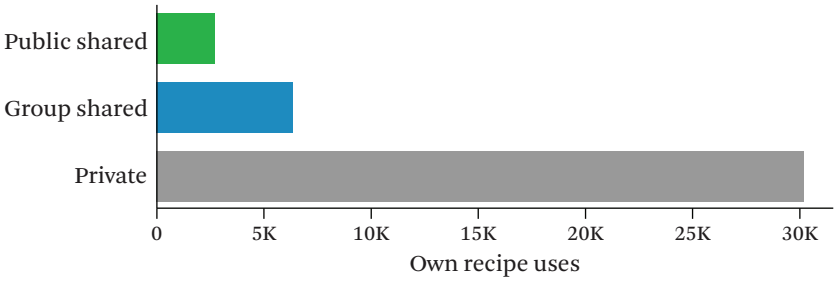


Figure 6.5 **Sharing status of recipes run by players who run only their own recipes in the CASP9 data set.** Most of the recipe uses are of private recipes, indicating that there is room to improve incentives for sharing recipes. (Figure from Cooper et al. [2011])

This data shows that the recipe sharing system is being used by the players and that there are distinct patterns of use among players. The high number of pure consumers indicates that recipes are being used by players who do not write them, and thus have the potential to facilitate the exchange of expertise from players who are writing recipes. However, there are a number of players who are running recipes that they

have not shared. More could be done to more to encourage these players to share their recipes, potentially by enabling players to earn points for others using their recipes.

6.4.2 Inheritance Relationships

Given that players were using and sharing recipes, we wanted to observe if players were modifying those recipes and what patterns of inheritance they produced. When a player shares a recipe, any player can then modify that recipe and create a new recipe that becomes a child of the original. Players can also modify their own private recipes to produce children. This allows for different variations of a similar concept to be produced, and it provides an opportunity for others to improve on a player's creation.

To observe patterns of how players were modifying other recipes, we looked at the number of descendants for each recipe. A descendant is any recipe that can be traced back through its parents to an original recipe. It is worth noting that it is possible for someone to take a shared recipe, rewrite it from scratch (with or without modifications), and call it their own. This would not show up as a descendant in our data set. Most recipes in the data set had no descendants, but there were over 250 recipes with one or more of them and 16 recipes with over 10 descendants (Figure 6.6).

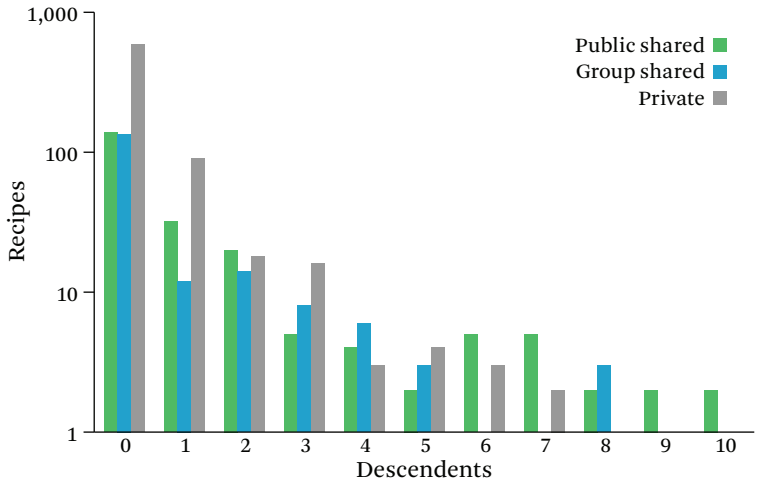


Figure 6.6 **A histogram of the number of recipes from the CASP9 data set by number of descendants.** There were 9 public shared, 3 group shared, and 4 private recipes with more than 10 descendants. While most recipes have no descendants, some recipes led to a variety of modifications and customizations, creating many descendants. (Figure from Cooper et al. [2011])

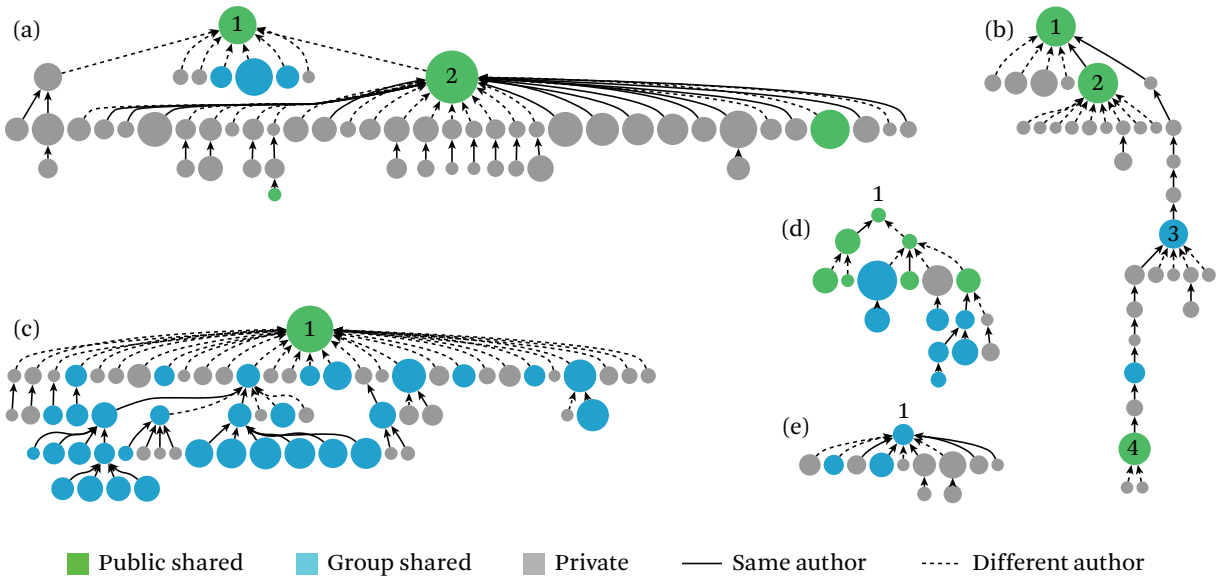


Figure 6.7 **Example hierarchies of recipe descendants from the CASP9 data set.** A child is created when a player edits a previously existing recipe. Size is logarithm of number of uses of a recipe. The ability to modify recipes shared by other players led to a variety of interesting modification patterns. (Figure from Cooper et al. [2011])

To investigate further, we looked at the inheritance hierarchies for the recipes with the most descendants. Several interesting patterns are shown in Figure 6.7. Panel (a) in Figure 6.7 is an example of the most-used recipe during CASP9, *Blue Fuse v1.1* (a, 2), which was derived from another public recipe, *Acid Tweeker v0.5* (a, 1). *Blue Fuse v1.1* has many descendants, but almost half of them are from the same author of *Blue Fuse v1.1* (shown as solid lines). The same is true for the *Loop Rebuild* family of recipes shown in panel (b). *Loop Rebuild 1.0* (b, 1) and *Loop Rebuild 1.1* (b, 2) were heavily used recipes during CASP9, with many other players creating different versions of both (shown as dotted arrows). The author of *Loop Rebuild 1.0* produced many descendants of it, sharing two of them with his group along the way (shown in cyan, such as *Loop Rebuild 1.9* (b, 3)) before finally sharing *Loop Rebuild 2.0* (b, 4) with the entire Foldit community. The hierarchy for *Blue Fuse v1.1* is broad, showing experimentation from a common starting point, while the hierarchy for *Loop Rebuild 1.0* is deep, showing iteration and refinement.

Some recipe writers publicly share their creation and do not create any modifications afterward. Panel (c) in Figure 6.7 shows an example of a popular recipe with many direct children, none of which were written by the original author. *Quake* (c, 1), which was the second most-used recipe during CASP9, has many different descendants, yet none of these modifications were made public. Many of these *Quake* descendants were shared within groups, but not with the rest of the Foldit community. The recipe *simple wiggle by ones* (d, 1) has many publicly shared descendants that ended up being used even more than the original parent during CASP9. The recipe *Lua EAR 3-5 LSR* (e, 1) was shared with the author's group and led to the creation of more group shared recipes.

Through this analysis it appears that players are taking advantage of the ability to modify and adapt the recipes of other players. Publicly shared recipes have the potential to create many more descendants with wide inheritance hierarchies, but keeping recipes private also provided value to players and allowed more localized refinement.

6.4.3 Ratings

The Foldit cookbook gives players the option of rating the recipes they use. Players can rate on the webpage for a recipe or in game via a popup. We wanted to determine if players were rating the recipes they used and what kind of ratings they were giving.

The number of users of a recipe who rated the recipe is shown in Figure 6.8. We found that it was difficult to get Foldit players to rate each other's recipes. While recipes with more unique users have more ratings, many users did not rate the recipes they used. Even with a pop-up to solicit ratings after using a new recipe several times, many users still did not rate the recipes.

We used a 5-star rating system, with 5 being the highest rating. However, we found those who did rate recipes mostly rated them with 5 stars (Figure 6.9). It is possible that players simply stop using recipes that they do not like, without bothering to rank them. Another possible explanation is that players do not want to negatively rate other players' creations, for fear that it might cause an author to no longer share their future recipes.

To improve feedback from ratings, we could require players to rate a recipe after using it many times before they are allowed to use it again or delete it. Perhaps a simple question, such as "Do you like this recipe?" with "Yes" and "No" buttons, or a "Like" feature might have yielded more ratings. A simple choice might have made it easier for players to decide how to rate. It might also be possible to have the option to vote always available in game while a recipe is running, rather than only popping up occasionally.

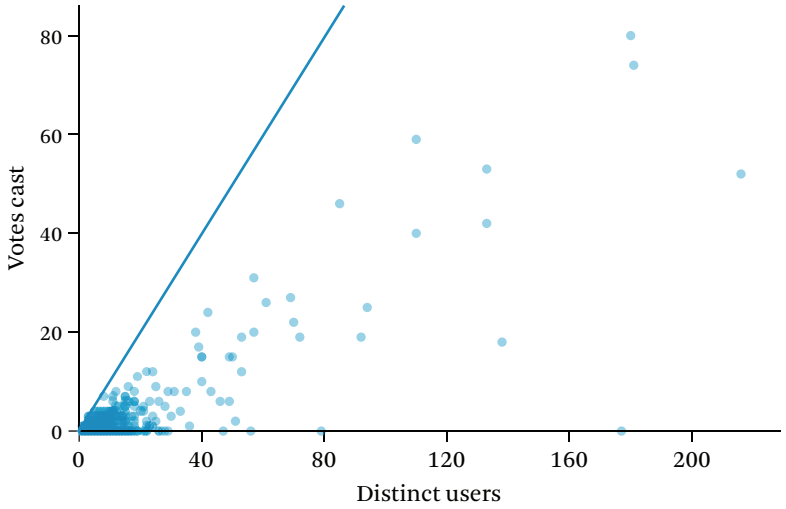


Figure 6.8 **Number of distinct users of a recipe and number of votes they cast for recipes in the CASP9 data set.** The line shows where points would be if every user rated. The average number of votes cast per user of a recipe was 0.25. Many recipe users do not rate recipes that they run. (Figure from Cooper et al. [2011])

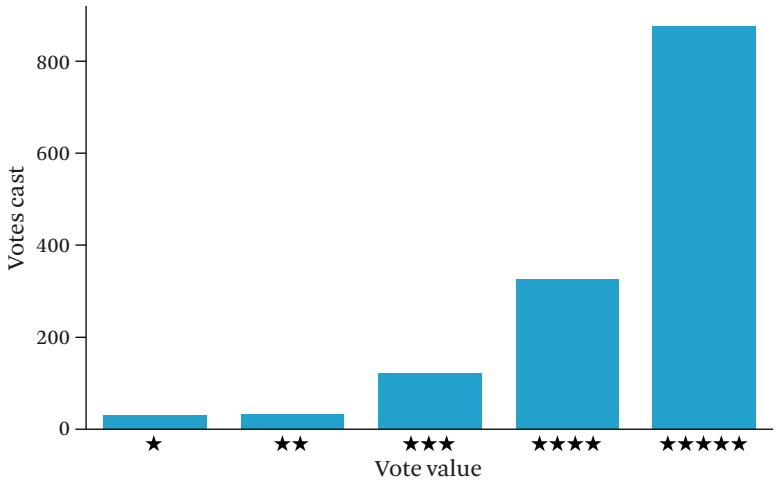


Figure 6.9 **Distribution of votes by rating for recipes in the CASP9 data set.** Most players who cast votes gave 5 stars. Players did not take advantage of the full range of ratings available, indicating a simpler rating system could have been used. (Figure from Cooper et al. [2011])

6.5 Script Recipe Adoption Analysis

We analyzed data around the time of the introduction of script recipes, to discover how the introduction of script recipes impacted the existing GUI recipes. Script recipes give players the option to write recipes based on the Lua scripting language in addition to the original GUI recipes, and were released several months later. The cookbook editor defaults to creating a GUI recipe, and there are two "New" buttons (one for each type of recipe). This data was gathered from July 2009 to April 2010.

Figure 6.10 shows numbers for recipes written and used during this time. We can see that script recipes quickly became popular to use. However, it took many weeks

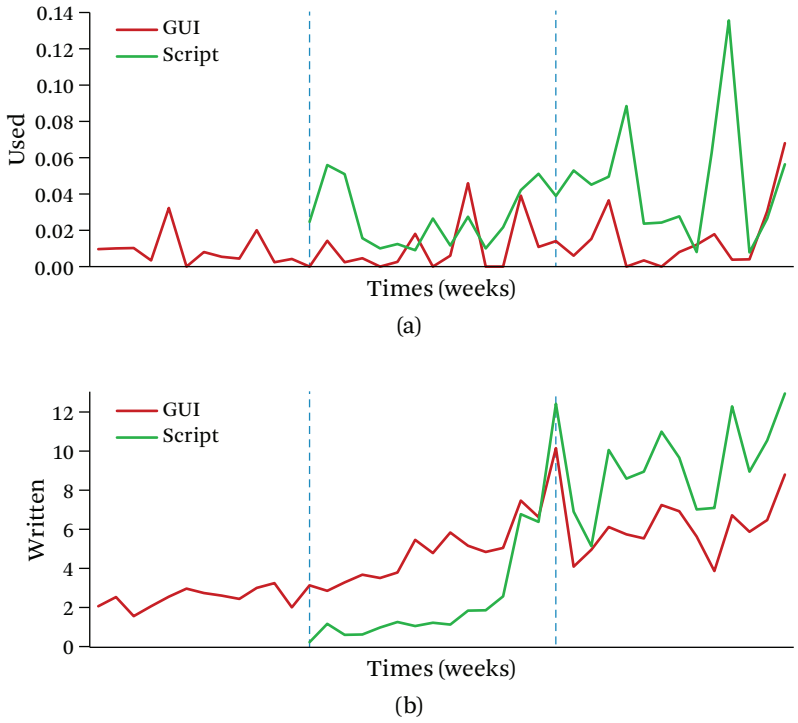


Figure 6.10 **Comparison of GUI and script recipes around the time of the introduction of scripting** Data is binned by week and normalized by number of unique competing players. Shown are (a) number of recipes written and (b) number of recipes run. The first vertical bar indicates when script recipes were introduced. The second indicates a refinement of the web interface for recipes. Even after the introduction of script recipes, GUI recipes continued to be written and used, indicating that both interfaces were useful. (Figure from Cooper et al. [2011])

before their popularity overtook the popularity of GUI recipes. In addition, it appears the use of GUI recipes remained popular and did not decline. Further, both script and GUI recipes continued to be authored at this point.

This indicates that GUI recipes continue to be useful even though the more powerful script recipes have been introduced. Therefore, it can be useful to provide both simpler visual methods and more advanced automation capabilities in scientific discovery games.

6.6 Algorithm Categories

Different algorithms were used with very different frequencies during the experiment. Some are designated by the authors as "public" and are available for use by all Foldit players, whereas others are "private" and are only used by the creator or their Foldit team. The distribution of recipe usage among different players is shown in Figure 6.11 for the 26 recipes that were run over 1,000 times. Some recipes, such as the one represented by the leftmost bar, were used many times by many different players, while others, such as the one represented by the pink bar in the middle, were used by only one player who chose not to share it with other players. Not surprisingly, the frequency of usage, indicated by the height of the bars, was significantly higher for the publicly shared recipes than the private ones.

We studied these popular player algorithms and found that they fell into four main categories: (1) perturb and minimize, (2) aggressive rebuilding, (3) local optimize, and (4) set constraints. The first category goes beyond the deterministic minimize function provided to Foldit players, which has the disadvantage of readily being trapped in local minima, by adding in perturbations to lead the minimizer in different directions. The second category uses the rebuild tool, which performs fragment insertion with loop closure, to search different areas of conformation space; these recipes are often run for long periods of time as they are designed to rebuild entire regions of a protein rather than just refining them.

The third category of recipes performs local minimizations along the protein backbone in order to improve the Rosetta energy for every segment of a protein. The final category of recipes assigns constraints such as pairing beta strands, randomly introducing rubber bands or changing the secondary structure assignment of a given protein to guide subsequent optimization. Foldit players use these algorithms to augment rather than to substitute for human strategizing. Players in essence perform a problem-informed search over the space of strategies, and use recipes to encode the specific lower-level strategies. Different algorithms are employed at different stages in game play. Figure 6.12 shows the frequency of recipe use in the opening, midgame, and

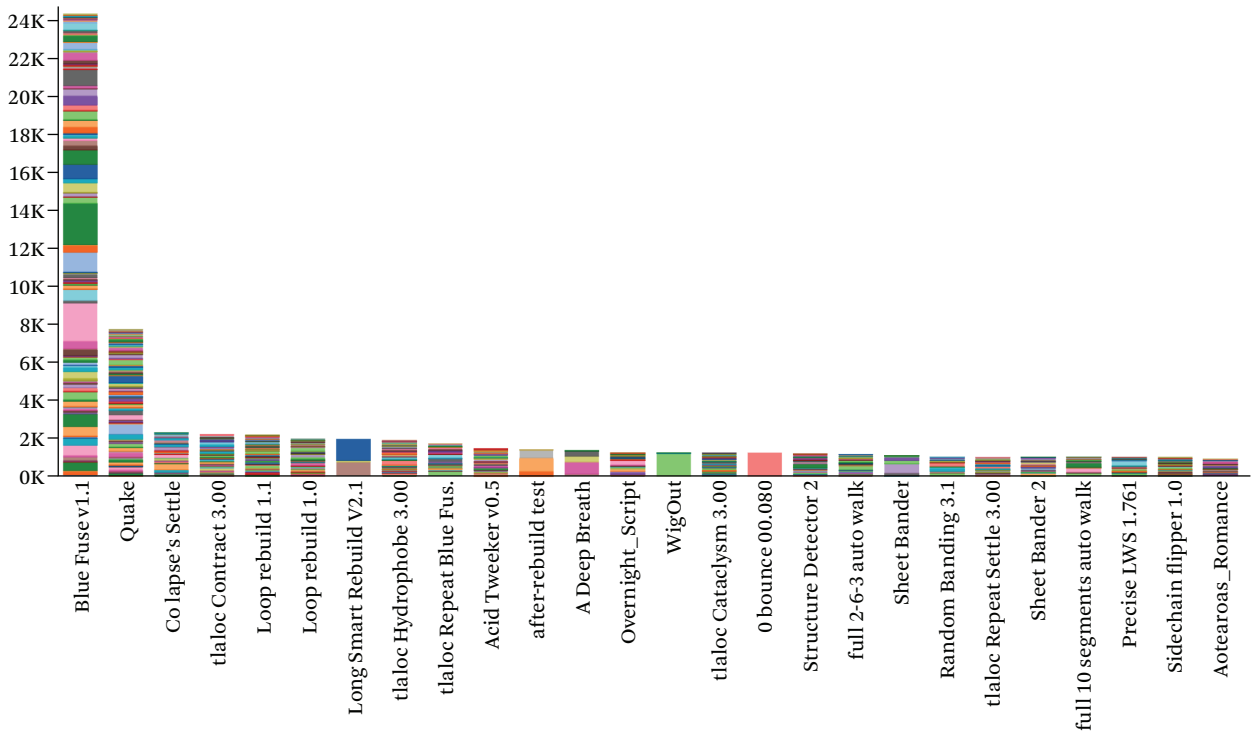


Figure 6.11 **Foldit recipes are used with very different frequencies.** Each bar on the x-axis represents a different Foldit recipe with the height of the bar denoting the number of times that recipe was used. Each color represents a different Foldit player who used that recipe. Recipes shown with only one color indicate that certain Foldit players did not share that private recipe with other players. *Blue Fuse v1.1*, at the very left was used over 24,000 times, more than 3 times as much as the next highest used recipe, *Quake*. (Figure from Khatib et al. [2011a])

endgame. Expert players exhibit different patterns of recipe use than the player population as a whole.

Local optimize recipes are heavily used in the endgame than at the beginning.

6.7 Context Dependence

Human strategic judgment plays an important role in choosing when and how to employ the recipes. Many of the aggressive rebuilding recipes, for example, require specifying which region of the protein to rebuild as it is rarely useful to entirely rebuild a protein chain from beginning to end. Most recipes heavily rely on the interactive aspects

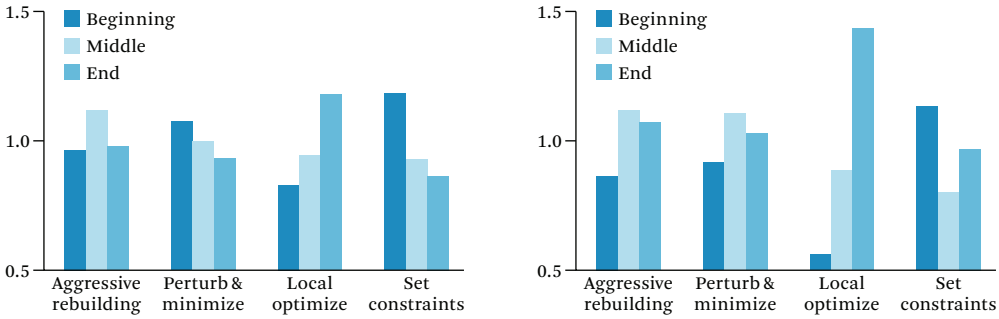


Figure 6.12 **Foldit players employ different recipes at different stages of game play.** The frequency of usage for each recipe category is shown for three different stages of game play (beginning in black, middle in beige, end in brown) for (a) all players and (b) top-ranked players. Local optimize recipes are more heavily used as gameplay progresses by all players, but even more so by top players. (Figure from Khatib et al. [2011a])

of the game, and are used less as automated tools and more as power tools that serve as an extension of the player's strategy. There is no recipe that takes a starting Foldit puzzle as input and is able to output a final model that ranks in the top 10 without any human intervention. Foldit players must guide recipes to perform the required task at each particular stage of the folding process. The local optimize recipes that walk along the protein backbone performing local minimizations are useless on the initial state of a Foldit puzzle that starts in an extended chain configuration, for example, because a successful prediction will no longer have that backbone in an extended conformation and will require a new round of local optimization. There are many recipes designed to be interacted with minimally; many of them are configured to run overnight effectively serving as alternative fully computational methods. These CPU intensive recipes can be useful for searching various low-energy regions of conformation space, but alone are not sufficient to outperform recipes used to support the human-lead exploration. These longer algorithms are often launched by top players once they have converged to a low-energy solution and are unable to improve their predictions any further, yet still want to search nearby regions of conformational space before local optimization.

This makes it difficult to quantify the usefulness and potential success of individual recipes since they all require very different protein states as inputs for optimal performance.

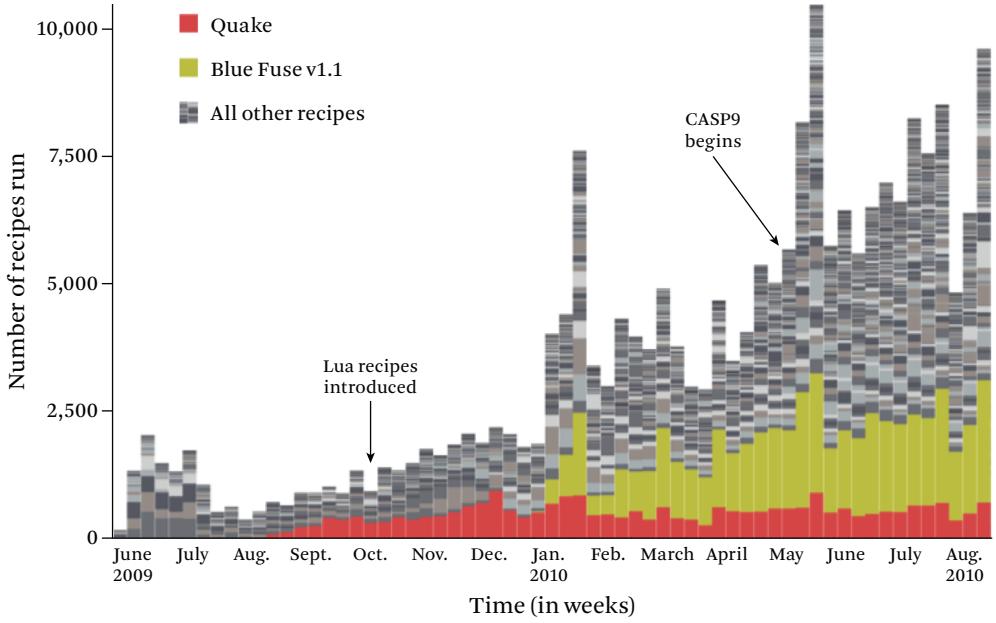


Figure 6.13 **Rapid proliferation of Foldit recipes between June 2009 and August 2010.** Each different shade of gray represents an individual recipe with the size of the gray bar denoting how many times that recipe was run that particular week. The most heavily used script recipe is *Blue Fuse v1.1*, shown in yellow. The most popular GUI recipe is *Quake*, shown in red, which has been used consistently by Foldit players weekly since its creation. (Figure from Khatib et al. [2011a])

6.8 Recipe Evolution

There was a rapid evolution of recipe use in the Foldit player population over the period from June 2009 (when the first GUI recipes were first introduced) through the end of August 2010. Figure 6.13 shows the number of distinct recipes run every week across this time period, with different shades of gray representing different recipes. In addition to the many individual recipes represented in gray, two recipes stand out as being heavily used: *Quake* (in red) and *Blue Fuse v1.1* (in yellow). The essential features of the algorithms contained within these two most popular recipes are similar. *Quake* is a GUI recipe that repeatedly optimizes the sidechain conformations and minimizes the sidechain and backbone torsion angles, increasing and decreasing the strength of an applied set of constraints to promote annealing of the structure. The *Blue Fuse v1.1* algorithm is conceptually similar to but simpler than *Quake*: rather than varying the

strength of artificial constraint functions, the strength of atom-atom repulsive interactions is alternately increased and decreased in repeated cycles of sidechain packing and complete torsion angle minimization.

In both cases, an initially poorly packed structure is alternatively compressed (by minimization with strong constraints or weak repulsive interactions) and then relaxed (by minimization with weak constraints or strong repulsive interactions); this alternation appears to be effective in locating low energy compact but well packed conformations. Foldit algorithms evolve in a social fashion, with popular recipes copied and then elaborated upon.

While the evolutionary mechanisms by which new algorithms are created (copying and variation) are clear, the mechanisms through which individual recipes spread through the population are less so. The popularity of *Quake* and *Blue Fuse v1.1* could result from individual player experimentation, player communication through chat, or player communication within teams. Interviews with Foldit players suggest that the primary mechanism by which successful algorithms take over the population is word of mouth. Players test and then recommend new recipes to others and rate recipes on the Foldit website. The reputation of the recipe author is also a factor for deciding whether a new recipe is worth trying or not. During this same time period, researchers in the Baker group were attempting to improve the core optimization methods within the Rosetta structure prediction and design program. A breakthrough was made with the *Fast Relax* algorithm, which achieves more effective energy optimization in less time than previous Rosetta methods. Benchmarks have shown *Fast Relax* to be considerably more efficient then an older algorithm, *Classic Relax* [Kuhlman et al. 2003] used since 2002 (Figure 6.15, compare green to red), and it is now used in all de novo and homology modeling methods as well as enzyme design strategies. The *Fast Relax* algorithm is comprised of interlaced cycles of full combinatorial repacking of the side-chains of the protein (repack) and gradient-based minimization of the backbone and sidechain degrees of freedom (minimize). These cycles are repeated 4 times while the strength of the repulsive weight is varied; starting at a low weight of 2% of the final repulsive weight, the weight is ramped over 4 cycles of repack/minimize to its final value, comprising a full *Fast Relax* round. Typically, 5–15 rounds of repulsive weight annealing are applied to a given structure and the lowest energy structure encountered (only full repulsive weight structures are eligible) is finally kept as the result. There is a striking similarity between the *Fast Relax* algorithm developed by scientists and the *Blue Fuse v1.1* algorithm developed by Foldit game players. These similarities are evident in the algorithm comparison shown in Figure 6.14. Both algorithms ramp the repulsive weight up and down while repacking and minimizing the structure, ultimately selecting as the algorithm output the lowest energy structure encountered. There are

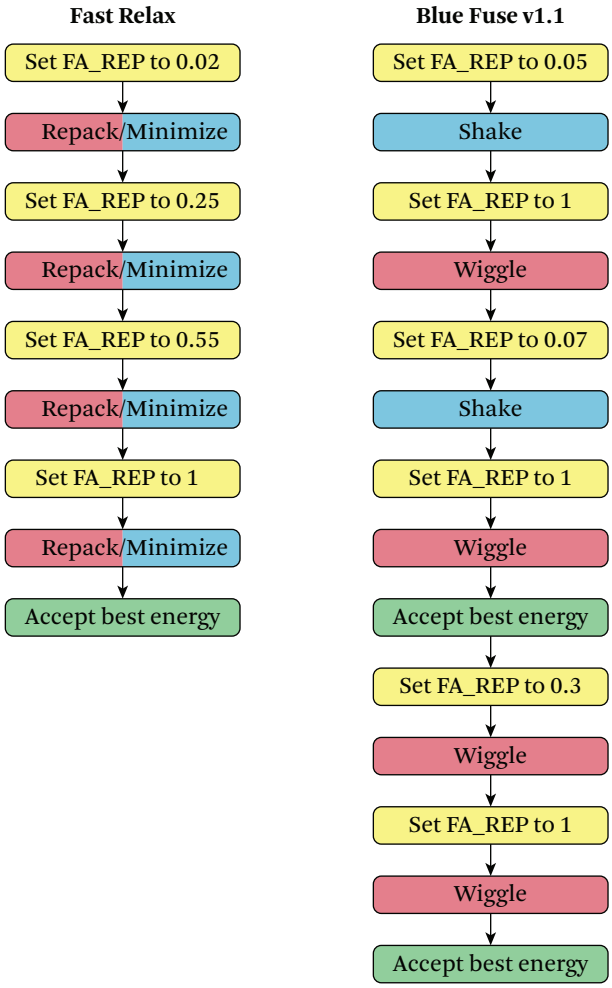


Figure 6.14 **Comparison of Foldit recipes to Rosetta algorithms developed by scientists.** Similarity between the unpublished Rosetta protocol *Fast Relax* and the Foldit recipe *Blue Fuse v1.1*. Equivalent steps in the two algorithms are represented with the same color. In Rosetta, the *Fast Relax* protocol is repeated 5 times by default, with a user-defined flag `-relax:thorough` resulting in a longer cycle of 15 rounds. (Figure from Khatib et al. [2011a])

minor differences—*Blue Fuse v1.1* begins with a low repulsive weight and only performs a shake before minimizing the structure at the standard weight while *Fast Relax* does a repack/minimize cycle at each stage—but the similarities far outweigh the differences. The Foldit players' discovery is in a rich tradition of softening the repulsive forces to enhance sampling in protein folding calculations. The earliest methods replaced entire subsets of atoms with simplified centroid sidechains [Levitt and Warshel 1975], and subsequent methods softened repulsive interactions in full atom representations [Levitt 1983]. *Classic Relax* expanded on this approach by gradually ramping up the repulsive term during the course of a simulation. Through social evolution of recipes, Foldit players independently rediscovered the utility of initially softening the repulsive interactions. They went beyond previous approaches by introducing the sawtooth-like repeated annealing of the strength of the repulsive interactions. This sawtooth profile likely induces repeated compaction and expansion of the protein chain, which evidently helps access new energy minima.

6.9 Performance Comparison

To determine how the *Blue Fuse v1.1* algorithm compared to both the *Classic Relax* and *Fast Relax* protocols, we ran all 3 algorithms on an in-house standard benchmark set consisting of 62 proteins with a range of structural diversity, including monomeric as well as multimeric structures with and without ligands. For each protein we included both native and close-to-native structures as well as non-native Rosetta models for a total of 6,758 structures. Each method was run on every input structure and average energies were extracted and analyzed, recording the improvement in Rosetta energy as a function of CPU time. Figure 6.15a shows that the *Blue Fuse v1.1* algorithm (in blue) performs more efficiently than *Classic Relax* (in red), but not as well as the newer *Fast Relax* protocol (in green). We tested other recipes, but found none that minimized the energy as quickly as *Blue Fuse v1.1*. Many recipes were able to find lower energies than *Blue Fuse v1.1* but took much longer to do so; this is not surprising as time is not at premium for most Foldit players. There is no penalty for a player to launch a recipe overnight, or before going out for the day, and if the recipe gains them any points by the time the player returns then it was productive. While repeated iterations of *Blue Fuse v1.1* fail to decrease the energy further, Foldit players discovered that by combining different recipes together with *Blue Fuse v1.1* lower energies can be achieved than with any of them alone (for example, *A Deep Breath* in Figure 6.15a). Human ingenuity complicates efforts to incorporate the rich new algorithms developed by Foldit players into automated methods such as Rosetta. Foldit players not only use different algorithms at different stages in game play, as noted earlier, but also employ them in specific subsets of situations.

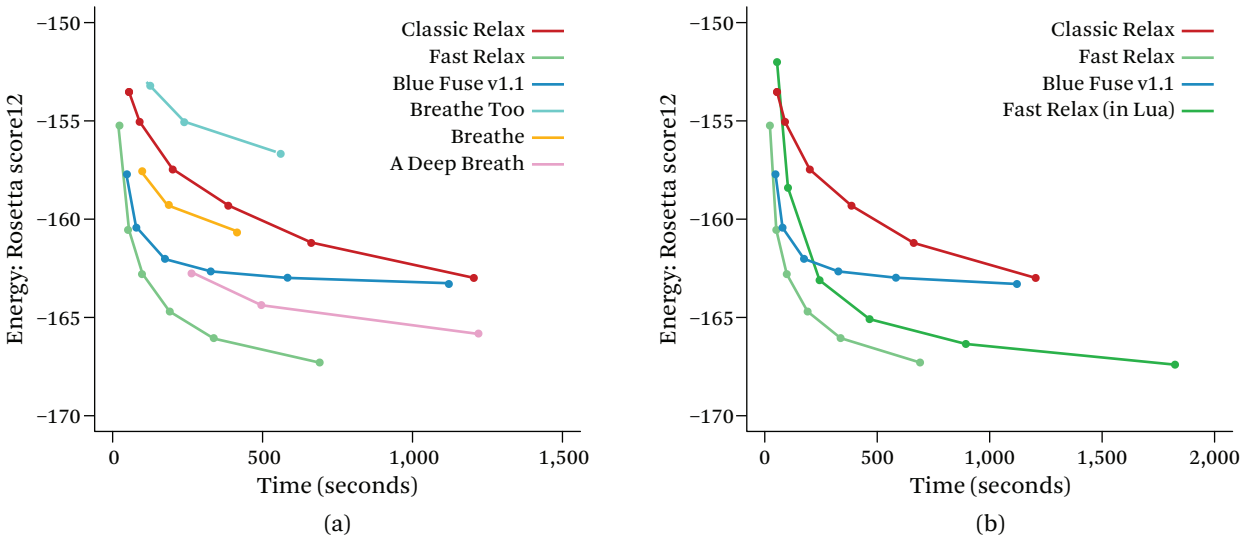


Figure 6.15 **Performance comparison between Rosetta *Relax* protocols and Foldit recipes.** After running each algorithm on a benchmark set of 6758 models, from 62 different proteins, we calculated the average energy and runtime for each method. The x-axis denotes the total runtime in seconds and the y-axis represents the Rosetta energy. (a) Foldit recipe *Blue Fuse v1.1* is able to sample lower energies than *Classic Relax* and reach these energies in much less time, but *Fast Relax* is even faster and more effective at energy optimization. *A Deep Breath* combines two other recipes (*Breathe* and *Breathe Too*) with *Blue Fuse v1.1*, resulting in a longer average runtime but making it able to sample lower energies than *Blue Fuse v1.1*. (b) Comparison to a reimplementation of *Fast Relax* in Foldit's Lua scripting interface. The Lua version of *Fast Relax* is less effective than the original version, and at short runtimes, less effective than *Blue Fuse v1.1*. (Figure from Khatib et al. [2011a])

The authors of *A Deep Breath*, for example, recommend adding rubber bands to a protein before running while other recipes are designed to be run after using the rebuild tool, such as after-rebuild test. Foldit players thus can recognize the precise circumstances under which a particular recipe may be useful, and choose from the growing palette of recipes accordingly. All recipes in the set constraints category, for example, are designed to be run before launching other recipes or manually manipulating the protein.

We also reimplemented *Fast Relax* in Foldit's Lua scripting interface for comparison in Figure 6.15b. This would be the implementation the players would have had if they had implemented the algorithm. The implementation of *Fast Relax* in Lua is

less effective than the original version. For short execution times, *Blue Fuse v1.1* (implemented in Lua) is able to sample lower energies than *Fast Relax* (in Lua). We found the average player's execution time for *Blue Fuse v1.1* to be around 122 s. For this execution time, *Blue Fuse v1.1* is able to sample lower energies than *Fast Relax* (in Lua). Given their scripting interface and common use case, the Foldit players were able to write a more effective algorithm.

6.10 Conclusion

This chapter discussed how players codify, automate, and share their expertise within the framework, in the form of gameplay macros. We showed how players develop effective algorithms through a process of social evolution. We showed that these algorithms, in some cases, surpass the effectiveness of scientist-developed algorithms.

Foldit tackles a difficult scientific problem, requiring players to master many different skills in order to successfully fold protein structures. Players have come up with various strategies to perform well in Foldit [Cooper et al. 2011]. Allowing Foldit players to create and run recipes was a method of codifying these strategies, and including the ability to share recipes has been instrumental in disseminating player strategies. Facilitating the sharing and modification of other's recipes is a method of distributing expertise. Players are easily able to run shared recipes without the prerequisite of being able to write them and may come up with different fruitful ways of using them that the authors may not have considered. Conversely, a player may not be the best at folding proteins but is a valuable asset to other Foldit players if they produce useful recipes.

We found that although most recipes have no descendants, there are still many shared recipes that have been modified by other players, often leading to more descendants. One way to increase these numbers would be to incentivize recipe sharing. This could be done by allowing players to earn points for others using their recipes, but this would require hiding the details of another players' recipe or they could easily be copied.

Despite the use of shared recipes, we found it difficult to get players to rate recipes they used. If players did rate, they was typically positive ratings. We would like to experiment with different incentives to encourage players to rate recipes they use, and simplify the rating system to reflect these usage patterns.

We found that both the visual GUI recipes and text-based script recipes were written and used despite vast differences in interface and expressiveness. This indicates that support for multiple authoring styles can be beneficial. We would like to further examine why players choose one interface over another and what kinds of recipes or styles of writing are best supported by each interface.

Our exploration of the ability of Foldit players to codify folding algorithms spawned a flurry of creative activity. There was a striking social evolution of Foldit player recipes in the year after their introduction. Particularly remarkable is the convergence of this evolution and its similarity to an unpublished algorithm developed by scientific experts over the same period of time that achieves faster and more effective energy optimization than previous methods [Khatib et al. 2011a]. Foldit players have been at a considerable disadvantage compared to scientists in developing new algorithms as the scripting language only exposes a small fraction of the variables and base algorithms in the Rosetta codebase. We are now generalizing the scripting language to allow control over more of these features, and look forward to learning what Foldit player ingenuity can do with these additional capabilities. More generally, the explosion of Foldit algorithms and the convergence on the best algorithm discovered thus far by scientists highlights the potential of scientific discovery games for complex problem solving and contributions to science and technology.

7 Conclusion

7.1 Contributions

This book has presented a framework for using a game to enable non-scientists to solve problems in order to make a contribution to a scientific domain. The framework takes a coevolution approach, allowing adaptation of both the players and the game, to progress towards scientific discovery. With this approach, we showed that players have the ability to make novel contributions to challenging proteomics problems in the field of biochemistry.

This work identifies useful criteria for determining the kinds of problems would lend themselves to such contributions by non-scientists, and an approach to map them onto games. The framework was expanded from an initial focus on protein structure prediction to additionally encompass protein design and the social development of protein refinement algorithms. We found that with a large population of engaged non-scientists, there exists the potential for some to discover novel scientific breakthroughs not previously possible.

In designing Foldit we sought to maximize both engagement by a wide range of players (a requirement common to all games), and the scientific relevance of the game outcomes (unique to Foldit). We fine-tuned the game through continuous coevolution based on observations of player activity and feedback, taking approaches from players who did well and making them accessible to all players. Most of the tools available to players today are a product of this refinement. They either did not initially exist or have undergone major revision. The introductory levels were also iteratively tuned to reduce player attrition due to difficulty or lack of engagement. Just as Foldit players gained expertise by playing Foldit, both individually and collectively, the game itself adapted to players' best practices and skill sets. We suspect that this process of coadaptation of game and players should be applicable to similar scientific discovery games.

7.2 Future Work

There is still much to be learned about the basis for human achievement with Foldit, which will require more specific analysis of how players acquire domain expertise

through gameplay, and can discover promising solutions. Such insights could also lead to improved automated algorithms for protein structure prediction.

One large question that this work has only begun to explore is the formalization of successful human strategies to solve the complex geometric task of protein structure prediction, given a set of tools. This is difficult because the general problem of protein structure prediction, and the specific problem of determining what is wrong with a particular predicted structure, is largely unknown. Furthermore, human strategies and their success appear to stem from their diversity and variability. It is not clear if humans' meta-pattern recognition ability allows them to intuit the answer, even when computational biologists cannot formulate the answer. Quantifying various problems with protein structures is an enormous challenge, one that computational biologists are still struggling with currently. We are in the process of analyzing player approaches and formalizing their variations towards improving current automated methods.

The contributions toward challenging proteomics problems by Foldit players demonstrates the considerable potential of a hybrid human-computer optimization framework in the form of a massively multiplayer game. The approach should be readily extendable to related problems, both in biochemistry and other scientific domains where human 3D structural problem solving can be leveraged. Our results suggest that scientific advancement is possible if even a small fraction of the energy that goes into playing computer games can be channeled into scientific discovery.

Bibliography

Peter Adriaenssens, Ephrem Eggermont, Karel Pyck, W. Boeckx, and B. Gilles. The video invasion of rehabilitation. *Burns Incl. Therm. Inj.*, 14 (5): 417–419, 1988. 7

Luis Von Ahn, Manuel Blum, and John Langford. CAPTCHA: Using hard AI problems for security. In *Proc. Int. Conf. Theory and Application of Cryptographic Techniques*, pages 294–311, 2003. 6

Mike Ambinder. Valve's approach to playtesting: The application of empiricism. Game Developer's Conference, March 2009. 11

David Anderson, Emily Anderson, Neal Lesh, Joe Marks, Brian Mirtich, David Ratajczak, and Kathy Ryall. Human-guided simple search. In *Proc. 17th National Conf. on Artificial Intelligence and 12th Innovative Applications of Artificial Intelligence Conf.*, pages 209–216, 2000. 6

David P. Anderson. BOINC: A system for public-resource computing and storage. In *Proc. IEEE/ACM International Workshop on Grid Computing*, pages 4–10, 2004. DOI: 10.1109/GRID.2004.14. 5

Christian B. Anfinsen. Principles that govern the folding of protein chains. *Science*, 181: 223–230, 1973. DOI: 10.1126/science.181.4096.223. 12, 34

Per Backlund, Henrik Engström, Cecilia Hammar, Mikael Johannesson, and Mikael Lebram. Sidh—a game based firefighter training simulation. In *Proc. 11th Int. Conf. Information Visualization*, pages 899–907, 2007. DOI: 10.1109/IV.2007.100. 7

Lucy M. Berlin and Robin Jeffries. Consultants and apprentices: Observations about learning and collaborative problem solving. In *Proc 1992 Conf. on Computer Supported Cooperative Work*, pages 130–137, 1992. DOI: 10.1145/143457.143471. 88

Marat Boshernitsan and Michael Downes. Visual programming languages: A survey. Technical Report CSD-04-1368, Department of Electrical Engineering and Computer Science, University of California, Berkeley, 2010. Available from http://techreports.lib.berkeley.edu/accessPages/CSD-04-1368.html; last retrieved May 2014. 87

Philip Bradley, Kira M. Misura, and David Baker. Toward high-resolution de novo structure prediction for small proteins. *Science*, 309 (5742): 1868–1871, September 2005. DOI: 10.1126/science.1113801. 13, 34

Ruth M. J. Byrne and P. N. Johnson-Laird. Spatial reasoning. *J. Memory and Language*, 28: 564–575, 1989. 1

Brookshire D. Conner, Scott S. Snibbe, Kenneth P. Herndon, Daniel C. Robbins, Robert C. Zeleznik, and Andries van Dam. Three-dimensional widgets. In *Proc. 1992 Symposium on Interactive 3D Graphics*, pages 183–188, 1992. DOI: 10.1145/147156.147199. 8

Matthew Conway, Steve Audia, Tommy Burnette, Dennis Cosgrove, and Kevin Christiansen. Alice: Lessons learned from building a 3D system for novices. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 486–493, 2000. DOI: 10.1145/332040.332481. 88

Seth Cooper, Firas Khatib, Adrien Treuille, Janos Barbero, Jeehyung Lee, Michael Beenen, Andrew Leaver-Fay, David Baker, Zoran Popović, and Foldit Players. Predicting protein structures with a multiplayer online game. *Nature*, 466 (7307): 756–760, August 2010a. DOI: 10.1038/nature09304. 27, 28, 29, 30, 35, 36, 37, 38, 39, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 65, 84, 85, 86

Seth Cooper, Adrien Treuille, Janos Barbero, Andrew Leaver-Fay, Kathleen Tuite, Firas Khatib, Alex Cho Snyder, Michael Beenen, David Salesin, David Baker, Zoran Popović, and Foldit Players. The challenge of designing scientific discovery games. In *Proc. 5th Int. Conf. on the Foundations of Digital Games*, pages 40–47, 2010b. DOI: 10.1145/1822348.1822354 . 13, 14, 15, 24, 25, 32, 45, 65

Seth Cooper, Firas Khatib, Ilya Makedon, Hao Lu, Janos Barbero, James Fogarty, Zoran Popović, and Foldit Players. Analysis of social gameplay macros in the Foldit cookbook. In *Proc. 6th Int. Conf. on the Foundations of Digital Games*, pages 9–14, 2011. DOI: 10.1145/2159365.2159367. 89, 91, 92, 93, 94, 96, 97, 106

Robert B. Corey and Linus Pauling. Molecular models of amino acids, peptides, and proteins. *Review of Scientific Instruments*, 24: 621–627, 1953. 8

Ernesto Cota, Stefan J. Hamill, Susan B. Fowler, and Jane Clarke. Two proteins with the same structure respond very differently to mutation: The role of plasticity in protein stability. *J. Molecular Biology*, 302 (3): 713–725, 2000. DOI: 10.1006/jmbi.2000.4053. 78

Charles Cusack, Chris Martens, and Priyanshu Mutreja. Volunteer computing using casual games. In *Proc. of Future Play Int. Conf. on the Future of Game Design and Technology*, 2006. 6

B. I. Dahiyat and S. L. Mayo. Protein design automation. *Protein Science*, 5 (5): 895–903, 1996. 67

Rhiju Das and David Baker. Macromolecular modeling with Rosetta. *Annual Review of Biochemistry*, 77 (1): 363–382, 2008. DOI: 10.1146/annurev.biochem.77.062906.171838. 34

Frank DiMaio, Thomas C. Terwilliger, Randy J. Read, Alexander Wlodawer, Gustav Oberdorfer, Ulrike Wagner, Eugene Valkov, Assaf Alon, Deborah Fass, Herbert L. Axelrod, Debanu Das, Sergey M. Vorobiev, Hideo Iwai, P. Raj Pokkuluri, and David Baker. Improved molecular replacement by density- and energy-guided protein structure optimization. *Nature*, 473: 540–543, 2011. DOI: 10.1038/nature09964. 62

K. Anders Ericsson. *Development of Professional Expertise: Toward Measurement of Expert Performance and Design of Optimal Learning Environments*. Cambridge, 2009. 87

Tracy Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Morgan Kaufmann, 1 edition, 2008. 11

Mark Griffiths. Video games and health. *BMJ*, 331: 122–123, July 2005. DOI: 10.1136/bmj.331.7509.122. 7

Xiaozhen Hu, Huanchen Wang, Hengming Ke, and Brian Kuhlman. Computer-based redesign of a β-sandwich protein suggests that extensive negative design is not required for *de novo* β-sheet design. *Structure*, 16 (12): 1799–1805, 2008. DOI: 10.1016/j.str.2008.09.013. 75

David Kennerly. Better game design through data mining. http://www.gamasutra.com/view/feature/2816/better_game_design_through_data_.php, 2003; last retrieved May 2014. 11

Firas Khatib, Seth Cooper, Michael Tyka, Kefan Xu, Ilya Makedon, Zoran Popović, David Baker, and Foldit Players. Algorithm discovery by protein folding game players. *Proc. National Academy of Sciences*, 108 (47): 18949–18953, 2011a. DOI: 10.1073/pnas.1115898108. 99, 100, 101, 103, 105, 107

Firas Khatib, Frank DiMaio, Foldit Contenders Group, Foldit Void Crushers Group, Seth Cooper, Maciej Kazmierczyk, Miroslaw Gilski, Szymon Krzywda, Helena Zabranska, Iva Pichova, James Thompson, Zoran Popović, Mariusz Jaskolski, and David Baker. Crystal structure of monomeric retroviral protease solved by protein folding game players. *Nature Structural and Molecular Biology*, 18: 1175–1177, 2011b. DOI: 10.1038/nsmb.2119. 59, 61, 64, 65

Lisa Kinch, Shuo Yong Shi, Qian Cong, Hua Cheng, Yuxing Liao, and Nick V. Grishin. CASP9 assessment of free modeling target predictions. *Proteins: Structure, Function, and Bioinformatics*, 79: 59–73, 2011. DOI: 10.1002/prot.23181. 60

B. Knispel et al. Pulsar discovery by global volunteer computing. *Science*, 329 (5997): 1305, 2010. DOI: 10.1126/science.1195253 . 5

Y. Koh, S. Matsumi, D. Das, M. Amano, D. A. Davis, J. Li, S. Leschenko, A. Baldridge, T. Shioda, R. Yarchoan, A. K. Ghosh, and H. Mitsuya. Potent inhibition of HIV-1

replication by novel non-peptidyl small molecule inhibitors of protease dimerization. *J. Biological Chemistry*, 282 (39): 28709–28720, 2007. DOI: 10.1074/jbc.M703938200. 62

B. Kuhlman, G. Dantas, G.C. Ireton, G. Varani, B.L. Stoddard, and D. Baker. Design of a novel globular protein fold with atomic-level accuracy. *Science*, 302 (5649): 1364–1368, 2003. 13, 102

N. Lesh, J. Marks, A. McMahon, and M. Mitzenmacher. New heuristic and interactive approaches to 2D rectangular strip packing. *J. Experimental Algorithmics*, 10, 2005. DOI: 10.1145/1064546.1083322. 6

M. Levitt and A. Warshel. Computer simulation of protein folding. *Nature*, 253 (5494): 694–698, 1975. DOI: 10.1038/253694a0. 104

Michael Levitt. Protein folding by restrained energy minimization and molecular dynamics. *J. Molecular Biology*, 170: 723–764, 1983. 104

Chris Lewis and Noah Wardrip-Fruin. Mining game statistics from web services: A World of Warcraft armory case study. In *Proc. 5th Int. Conf. on the Foundations of Digital Games*, pages 100–107, 2010. DOI: 10.1145/1822348.1822362. 88

J. MacCallum et al. Assessment of the protein structure refinement category in CASP9. *Proteins: Structure, Function, and Bioinformatics*, 79 (S10): 74–90, 2011. DOI: 10.1002/prot.23131. 62

Wendy E. Mackay. Patterns of sharing customizable software. In *Proc 1990 Conf. on Computer Supported Cooperative Work*, pages 209–221, 1990. DOI: 10.1145/99332.99356. 88

Matt MacLaurin. Kodu: End-user programming and design for games. In *Proc. 4th Int. Conf. on the Foundations of Digital Games*, 2009. DOI: 10.1145/1536513.1536516. 88

Matthew B. MacLaurin. The design of Kodu: A tiny visual programming language for children on the Xbox 360. In *Proc. 38th ACM SIGACT-SIGPLAN Symp. on Principles of Programming Languages*, pages 241–246, 2011. DOI: 10.1145/1926385.1926413. 88

John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The Scratch programming language and environment. *ACM Trans. Computing Education*, 10: 16:1–16:15, 2010. DOI: 10.1145/1868358.1868363. 88

Adam Martin, Brooke Thompson, Tom Chatfield, Andrea Phillips, Brooke Thompson, Bryan Alexander, Christy Dena, and Nova Barlow. 2006 alternate reality games white paper, 2006. Available from http://www.igda.org/arg/resources/IGDA-AlternateRealityGames-Whitepaper-2006.pdf; last retrieved May 2014. 7

A. Mastrolorenzo, S. Rusconi, A. Scozzafava, G. Barbaro, and C. T. Supuran. Inhibitors of HIV-1 protease: Current state of the art 10 years after their introduction. From antiretroviral drugs to antifungal, antibacterial and antitumor agents based on aspartic protease inhibitors. *Current Medicinal Chemistry*, 14 (26): 2734–2748, 2007. 62

Airlie J. Mccoy, Ralf W. Grosse-Kunstleve, Paul D. Adams, Martyn D. Winn, Laurent C. Storoni, and Randy J. Read. Phaser crystallographic software. *J. Applied Crystallography*, 40: 658–674, 2007. DOI: 10.1107/S0021889807021206. 62

John Moult, Krzysztof Fidelis, Andriy Kryshtafovych, Burkhard Rost, and Anna Tramontano. Critical assessment of methods of protein structure prediction—round VIII. *Proteins: Structure, Function, and Bioinformatics*, 77 (S9): 1–4, 2009. DOI: 10.1002/prot.22589. 42

Vijay Natarajan, Patrice Koehl, Yusu Wang, and Bernd Hamann. Visual analysis of biomolecular surfaces. In Lars Linsen, Hans Hagen, and Bernd Hamann, editors, *Visualization in Medicine and Life Sciences*, pages 237–255. Springer, 2008. DOI: 10.1007/978-3-540-72630-2_14. 8

Anders C. Olson and Richard W. Roberts. Design, expression, and stability of a diverse protein library based on the human fibronectin type III domain. *Protein Science*, 16 (3): 476–484, 2007. DOI: 10.1110/ps.062498407. 75

Vijay S. Pande, Ian Baker, Jarrod Chapman, Sidney P. Elmer, Siraj Khaliq, Stefan M. Larson, Young Min Rhee, Michael R. Shirts, Christopher D. Snow, Eric J. Sorin, and Bojan Zagrovic. Atomistic protein folding simulations on the submillisecond time scale using worldwide distributed computing. *Biopolymers*, 68: 91–109, 2003. DOI: 10.1002/bip.10219. 8

Bin Qian, Srivatsan Raman, Rhiju Das, Philip Bradley, Airlie J. McCoy, Randy J. Read, and David Baker. High-resolution structure prediction and the crystallographic phase problem. *Nature*, 450 (7167): 259–264, 2007. DOI: 10.1038/nature06249. 13, 34, 44

Srivatsan Raman, Robert Vernon, James Thompson, Michael Tyka, Ruslan Sadreyev, Jimin Pei, David Kim, Elizabeth Kellogg, Frank DiMaio, Oliver Lange, Lisa Kinch, Will Sheffler, Bong-Hyun Kim, Rhiju Das, Nick V. Grishin, and David Baker. Structure prediction for CASP8 with all-atom refinement using Rosetta. *Proteins: Structure, Function, and Bioinformatics*, 77 (S9): 89–99, 2009. DOI: 10.1002/prot.22540. 44

Judith Ramey, Ted Boren, Elisabeth Cuddihy, Joe Dumas, Zhiwei Guan, Maaike J. van den Haak, and Menno D. T. De Jong. Does think aloud work? How do we know? In *CHI '06 extended abstracts on Human factors in computing systems*, pages 45–48, 2006. DOI: 10.1145/1125451.1125464. 24

Randy J. Read and Gayatri Chavali. Assessment of CASP7 predictions in the high accuracy template-based modeling category. *Proteins: Structure, Function, and Bioinformatics*, 69 (S8): 27–37, 2007. DOI: 10.1002/prot.21662. 40

C. A. Rohl, C. E. Strauss, K. M. Misura, and D. Baker. Protein structure prediction using Rosetta. *Methods in Enzymology*, 383: 66–93, 2004. 8, 13, 33

Antonio Rosato, Anurag Bagaria, David Baker, Benjamin Bardiaux, Andrea Cavalli, Jurgen F. Doreleijers, Andrea Giachetti, Paul Guerry, Peter Guntert, Torsten Herrmann,

Yuanpeng J. Huang, Hendrik R. A. Jonker, Binchen Mao, Therese E. Malliavin, Gaetano T. Montelione, Michael Nilges, Srivatsan Raman, Gijs van der Schot, Wim F. Vranken, Geerten W. Vuister, and Alexandre M. J. J. Bonvin. CASD-NMR: Critical assessment of automated structure determination by NMR. *Nature Methods*, 6 (9): 625–626, 2009. 46

Jesse Schell. *The Art of Game Design: A Book of Lenses*. 2008, Morgan Kaufmann. 15

Will Sheffler and David Baker. RosettaHoles: Rapid assessment of protein core packing for structure prediction, refinement, design, and validation. *Protein Science*, 18 (1): 229–239, 2009. DOI: 10.1002/pro.8. 75

Yang Shen, Oliver Lange, Frank Delaglio, Paolo Rossi, James M. Aramini, Gaohua Liu, Alexander Eletsky, Yibing Wu, Kiran K. Singarapu, Alexander Lemak, Alexandr Ignatchenko, Cheryl H. Arrowsmith, Thomas Szyperski, Gaetano T. Montelione, David Baker, and Ad Bax. Consistent blind protein structure generation from NMR chemical shift data. *Proc. National Academy of Sciences*, 105 (12): 4685–4690, 2008. DOI: 10.1073/pnas.0800256105. 56
56

Johannes Söding, Andreas Biegert, and Andrei N. Lupas. The HHpred interactive server for protein homology detection and structure prediction. *Nucleic Acids Research*, 33 (2): W244–W248, 2005. 59

D. A. Stainforth, T. Aina, C. Christensen, M. Collins, N. Faull, D. J. Frame, J. A. Kettleborough, S. Knight, A. Martin, J. M. Murphy, C. Plani, D. Sexton, L. A. Smith, R. A. Spicer, A. J. Thorpe, and M. R. Allen. Uncertainty in predictions of the climate response to rising levels of greenhouse gases. *Nature*, 433: 403–406+, 2005. DOI: 10.1038/nature03301. 5

Woodruff T. Sullivan III, Dan Werthimer, Stuart Bowyer, Jeff Cobb, David Gedye, and David Anderson. A new major SETI project based on Project Serendip data and 100,000 personal computers. In *Proc. 5th Intl. Conf. on Bioastronomy*, 1997. 5

Mark C. Surles, Jane S. Richardson, David C. Richardson, and Fred P. Brooks. Sculpting proteins interactively: Continual energy minimization embedded in a graphical modeling system. *Protein Science*, 3: 198–210, 1994. 8

Tarja Susi, Mikael Johannesson, and Per Backlund. Serious games—An overview. Technical Report HS-IKI-TR-07-001, School of Humanities and Informatics, University of Skövde, Sweden, February 2007. Available from http://www.diva-portal.org/smash/get/diva2:2416/FULLTEXT01.pdf; last retrieved May 2014. 7

Sureyya Tarkan, Vibha Sazawal, Allison Druin, Evan Golub, Elizabeth M. Bonsignore, Greg Walsh, and Zeina Atrash. Toque: Designing a cooking-based programming language for and with children. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 2417–2426, 2010. 88

Daniel Terdiman. I Love Bees game a surprise hit. *Wired*, October 2004. 7

Michael Tress, Iakes Ezkurdia, Osvaldo Graña, Gonzalo López, and Alfonso Valencia. Assessment of predictions submitted for the CASP6 comparative modeling category. *Proteins: Structure, Function, and Bioinformatics*, 61 (S7): 27–45, 2005. 40

Barbara Tversky. Cognitive maps, cognitive collages, and spatial mental models. In Andrew U. Frank and Irene Campari, editors, *Spatial Information Theory: A Theoretical Basis for GIS*, volume 716 of *Lecture Notes in Computer Science*, pages 14–24. Springer, 1993. DOI: 10.1007/3-540-57207-4_2. 1

V. Veverka, H. Bauerova, A. Zabransky, J. Lang, T. Ruml, I. Pichova, and R. Hrabal. Three-dimensional structure of a monomeric form of a retroviral protease. *J. Molecular Biology*, 333: 771–780, 2003. DOI: 10.1016/j.jmb.2003.08.049. 62

Luis von Ahn and Laura Dabbish. Labeling images with a computer game. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 319–326, 2004. DOI: 10.1145/985692.985733. 6, 33

Luis von Ahn, Ruoran Liu, and Manuel Blum. Peekaboom: A game for locating objects in images. In *Proc. SIGCHI Conf. on Human Factors in Computing Systems*, pages 55–64, New York, NY, USA, 2006. DOI: 10.1145/1124772.1124782. 6, 33

Kevin Wang, Corey McCaffrey, Daniel Wendel, and Eric Klopfer. 3D game design with programming blocks in StarLogo TNG. In *Proc. 7th Int. Conf. on Learning Sciences*, pages 1008–1009, 2006. 88

Andrew J. Westphal, Sasa Bajt, Hans A. Bechtel, Janet Borg, D. Brownlee, Anna L. Butterworth, George Cody, C. Floss, George J. Flynn, Z. Gainsforth, Anton Kearsley, Larry R. Nittler, Scott A. Sandford, F.J. Stadermann, Thomas Stephan, Rhonda M. Stroud, P. Tsou, Tolek Tyliszczak, Michael Zolensky, and et al. Non-destructive search for interstellar dust using synchrotron microprobes. In *AIP Conference Proc.*, volume 1221, pages 131–138, 2010. 6, 33

Dmitri Williams, Nick Yee, and Scott E. Caplan. Who plays, how much, and why? Debunking the stereotypical gamer profile. *J. Computer-Mediated Communication*, 13 (4): 993–1018, 2008. DOI: 10.1111/j.1083-6101.2008.00428.x. 88

A. Wlodawer and A. Gustchina. Structural and biochemical studies of retroviral proteases. *Biochimica et Biophysica Acta*, 1477: 16–34, 2000. 62

Nick Yee. Motivations for play in online games. *CyberPsychology and Behavior*, 9 (6): 772–775, 2006. DOI: 10.1089/cpb.2006.9.772. 27

A. Zemla. LGA: A method for finding 3D similarities in protein structures. *Nucleic Acids Research*, 31 (13): 3370–3374, 2003. DOI: 10.1093/nar/gkg571. 44, 45

Michael Zenke. Land of the free: The rise of the tiny MMO. *Game Developer Magazine*, pages 7–12, June/July 2008. 7

Chao Zhang and Sung-Hou Kim. Overview of structural genomics: From structure to function. *Current Opinion in Chemical Biology*, 7: 28–32, February 2003. DOI: 10.1016/S1367-5931(02)00015-7. 2

Author's Biography

Seth Cooper

Seth Cooper is the creative director of the Center for Game Science at the University of Washington. He received his Ph.D. in Computer Science and Engineering from the University of Washington. His dissertation, *A Framework for Scientific Discovery through Video Games,* advised by Zoran Popović, won the ACM Doctoral Dissertation Award in 2011. His current research focuses on using video games to solve difficult scientific problems, and he has delivered TED talks on the topic. He is co-creator of the scientific discovery game Foldit and early math educational games including Refraction and Treefrog Treasure. He has also done research in real-time animation for games, often in the motion capture lab. He has previously worked at Square Enix, Electronic Arts, Pixar Animation Studios, and the UC Berkeley Space Sciences Laboratory (on BOINC, the Berkeley Open Infrastructure for Network Computing).