Parallel Experiment Planning: Macromolecular Crystallization Case Study

by

Vanathi Gopalakrishnan

B.E., Bangalore University, 1988

M.S., University of Pittsburgh, 1991

Submitted to the Graduate Faculty of

Arts and Sciences in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

1999

UNIVERSITY OF PITTSBURGH

FACULTY OF ARTS AND SCIENCES

This dissertation was presented

by

Vanathi Gopalakrishnan

It was defended on

November 17 1999

and approved by

Prof. Bruce G. Buchanan (Chairman)

Prof. John M. Rosenberg

Prof. Johanna D. Moore

Prof. Siegfried Treu

Committee Chairperson(s)

Parallel Experiment Planning:
Macromolecular Crystallization Case Study

Vanathi Gopalakrishnan, Ph.D.
University of Pittsburgh, 1999

In macromolecular crystallization [43, 44], a large parameter space must be searched in order to find at least one set of experimental conditions that produce a good quality crystal of a macromolecule. Few heuristics exist since the observable effects of experiments change over time. Hence, crystallographers typically resort to trial-and-error experimentation. The nature of experimentation is tedious since several experiments have to be set up in parallel in order to search efficiently.

In this dissertation a Parallel Experiment Planning (PEP) framework is developed that: (1) provides a computational representation and set of tools to manage information about parallel experiments (or trials), and (2) can provide intelligent assistance for decision-making by suggesting likely places in search space for new trials and portions of space that are unlikely to yield results so that they could be closed.

Sufficiency of the PEP framework is demonstrated by developing a prototype implementation of the complex environment that supports parallel experiment planning. This prototype PEP system is evaluated by experiments with human decision-making agents who can manipulate the environment to achieve their goals. The framework and system are evaluated with respect to usability, completeness or sufficiency, utility, time complexity, and flexibility.

This work may also be viewed as including reinforcement learning wherein there exists an agent that tries to learn a policy that can achieve its goals by manipulating a complex environment and receiving feedback from it. Much of prior work in reinforcement learning in artificial intelligence uses only a real-valued feedback or reward in sequential probes of the environment [65, 66, 6, 41]. Within the framework developed here, an agent receives more than one type of feedback, which includes a set of symbolic rules, and receives feedback from multiple probes of (copies of) the same environment. Symbolic feedback amounts to a description of the boundary between observed classes of partial results that can be utilized to prune the vast search space, and thus is more useful than point values.

Preface

Dedicated to the change of seasons that occur in the continuity of research.

I would like to thank Raj Reddy and Anu Reddy first and foremost for welcoming me to the U.S.A., hosting me so unforgetably well, and introducing me to Bruce Buchanan.

It is wonderful to be in the company of great people – teachers and scholars who exemplify virtues such as humility, patience and gentleness in their everyday life and interactions with people. I am indeed fortunate to be associated with Bruce Buchanan for the past 10 years. He has been very patient and understanding throughout this entire journey that I embarked upon in order to obtain a doctorate degree in computer science. He has molded and shaped my thinking by his subtle remarks. I am very grateful to Bruce for his firm belief in my abilities and for helping me conceptualize and express the ideas that form the basis of this dissertation. I would also like to thank him for getting me interested in computational biology.

Bruce introduced me to John Rosenberg from the biological sciences department at Pitt 5 years ago, and my interests changed rapidly since then. John loves science, does science and teaches science. It is impossible to not love science after you have had a chance to talk to him. So, even though I might have been able to obtain my degree sooner if I had actually worked on something related to my Master's project, I decided to work toward a dissertation that would demonstrate ideas that had some utility in a complicated real-world problem, namely macromolecular crystallization. This, by the way, happens to be John Rosenberg's expertise. I am very grateful for the generous amount of time and encouragement that John provided throughout the development and refinement of the work described in this dissertation.

Johanna Moore has always encouraged me. She and I used to be among the people working in MIB until the wee morning hours during my first years at Pitt. Johanna gave me great advice about doing applied AI work that I followed. I am very grateful to Johanna for her belief in me and the encouragement and support I have always received from her.

Sig Treu has helped me in many ways. My Master's project was demonstrated using a simulation language called HSL that was developed here at Pitt by Pete Sanderson and others who were Sig's students. I am grateful to Sig for helping me express my ideas more eloquently by asking me important questions related to the problem and my approach.

I would like to especially thank my fellow occupants of the Keck Lab – Ethan Benatan and Dan Hennessy for all the fun and learning that happened when we were together. I sure miss those days. Discussions with Foster Provost and Scott Clearwater are also missed as they were informative as well as insightful. These discussions have helped me understand the main contributions of several research papers. John Aronis has helped me understand very clearly the difference between neat and scruffy AI.

Special thanks to Erik Sassaman, who helped me encode some of the mathematical functions required for the simulation using Numerical recipes in C.

Devika Subramanian, who is currently at Rice University, made a profound impression upon me. I learnt so much during her brief stay in Pittsburgh. I am extremely grateful to her for her guidance and support during the initial stages of the crystallization project.

Jude Shavlik enabled me to spend a summer at Madison, Wisconsin. I am very thankful to him for exposing me to various aspects of and projects in computational biology. I enjoyed my conversations with Mark Craven and am happy to discover that we happen to have similar professional interests.

I would like to thank my fellow graduate students here at Pitt, particularly Rastislav Bodik, Nilufer Onder and Ioannis for their ready help at all times. Lucille Jarzynka, Keena Walker, Loretta Shabatura and Jeannie Bakosh are wonderful people and I am grateful for their encouragement and support during this process.

I would like to thank my friends and family for being supportive through all the difficult times. I am very fortunate to be born of parents that are as supportive as mine. My father, R. V. Gopalakrishnan, is a value engineering expert with a doctorate degee in the pure siences from Glasgow University. He became an operation-induced paraplegic 12 years ago in India, but his enthusiasm and determination have enabled him to continue to contribute to society through his expertise despite being in a wheelchair. My father knew Raj Reddy during his undergraduate days and made sure that I would be well taken care of as I made my entry into an unknown foreign land.

My mother, Indira Gopalakrishnan, is an amazing individual with remarkable gifts. She is endowed with beauty, intelligence, courage, cheerfulness and spontaneous wit. Through humor, she can bring cheer and laughter into any room. The last 12 years have placed undue stress on her person, but through her patience, perseverance and hard work, she has single-handedly helped my father become more independent. She has been the most positive influence in my life. Without her help, I would have been able to neither start nor complete this work.

I owe a lot to my grandparents who simply showered their love upon me. I always reflect upon their greatness for inspiration and strength. Many thanks to my loving brothers, R. G. Srinivas and Mohanaraman for frequently nudging me on towards my goals. I am grateful to my brother-in-law Subramani and his wife Shanti for helping me move back to Pittsburgh so that I could finish this dissertation. Thanks to Ram, Lauren, Rose and Dolly who have inspired and helped me in several ways.

Lastly, I am extremely grateful to my husband Ganesh Mani and daughter Serena. They have made plenty of sacrifices and provided the love, support and encouragement I needed in order to complete this dissertation.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Problem

In experimental science, a very large parameter space needs to be searched to find one or more sets of conditions that define a satisfactory solution. Such searches have to be performed taking into account resource limitations, which exist in almost all real-world problems. We have focussed our research on one such problem, namely, that of experiment design in macromolecular crystallization. In this domain, the goal of experimentation is to obtain a good quality (X-ray diffractible) crystal of a macromolecule (protein, DNA, or protein-DNA complexes). Such a crystal can be obtained only under certain specific conditions, which vary from macromolecule to macromolecule, that arise due to the mixing of different chemical compounds under varying conditions of physical factors such as temperature, pressure and gravity. There are some specific features related to growing crystals of macromolecules that make this problem particularly interesting. These include:

1. effects of actions changing over time, that is, partial results of experiments vary over time. For example, we could observe a crystal in an experimental apparatus on one day, but a week later it could have dissolved completely.

2. imprecise evaluation of partial results. There exists only a crude local evaluation function (based on visual inspection) for partial result determination of each experiment.

3. a large degree of interdependence among the variables – the relationships of their interaction is largely unknown.

4. the tedious nature of experimentation. Long hours are spent in repeatedly pipetting solutions into experimental apparatus since typically several (50 to 100) experiments are performed in parallel.

Tradeoffs are necessary in order to successfully search a large multi-dimensional parameter space to find a small region that yields a good quality crystal of a macromolecule.

Given these problem characteristics, one central question is the following: Can we infer global strategies for designing several experiments in parallel, that can lead to a satisficing solution, given (a) limited resources with costs associated with use of each resource, (b) effort involved in performing each experiment,(c) partial results that vary over time and (d) only a crude local evaluation function (that provides partial result for an experiment at any given time of observation)?

1.2 The Approach

This research describes a framework, called the PEP framework, within which global strategies can be tried by a human designer, and provides reflective statistical summaries of the choices made for experimentation. The framework includes a predictive model that can be used for simulating an experimental outcome at a particular time. Different scenarios that are representative of the difference in response behavior of hypothetical proteins in various physico-chemical environments over time can be produced using simulation. The framework is general enough for use without a simulator - actual observations can be noted instead. The usefulness of the framework is greatly increased, however, with the use of simulation, since the implementation can be used as an intelligent decision-making aid and as a tool for training novice crystallographers to devise strategies for parallel experimentation for different types of macromolecules.

The framework uses terms that have the following definitions:

1. *Experiment*: A combination of conditions (values for control parameters) that result in a series of outcomes at different times and consume some set of resources.

2. *Trial*: A group or population of experiments that are performed in parallel.

3. *Project*: the problem (e.g. the protein to be crystallized).

4. *Global Information*: Information about the constants and variables for a project, as well as experimental record. This record consists of statistical summaries, and boundary information that represents the evolving boundary between classes of observable partial results.

5. *Local policy*[1] *(or search heuristic)*: the basis for defining a new trial after observing partial results.

6. *Global strategy (or search protocol)*: the abstract description of a series of trials leading to the goal (such as a good quality crystal).

If we were to try to represent the crystallization problem as an AI planning problem, it would have to be cast as a reactive planning problem, as the environment is dynamically changing.

[1] A complete mapping from states to actions describes a policy. A policy therefore represents a simple reflex agent that knows what action to perform based on what state it is in.

The biggest challenge would arise in how to (re)plan based on evaluation of partial results. Thus, we would need an evaluation function that can assign probabilities to different regions of the search space of experiments based on observation and evaluation of partial results over a period of time. These factors make this problem very different from traditional AI planning problems (For some well-known planning frameworks in the AI literature see [23, 51, 42, 19, 64, 1]).

Another point to note is that computational research in the domain of macromolecular crystallization has begun only recently [27, 62, 32]. It is important to note the distinction between macromolecular crystallization and X-ray crystallography. In macromolecular crystallization, the goal is to obtain a crystal from protein or other macromolecule that exists in solution. X-ray crystallography refers to the domain of analysis of the X-ray diffraction patterns that are obtained by diffracting a crystal that has been obtained by macromolecular crystallization techniques. X-ray crystallography involves Fourier transforms and other analysis of X-ray image data, which is computationally intensive and has been researched for several decades [38, 24, 68]. This dissertation focuses on the more fundamental question of how to grow a crystal of a macromolecule, which is the first step before X-ray crystallography can be used to obtain the 3-dimensional structure of the macromolecule.

The process of obtaining a well diffracting crystal of a macromolecule may involve months or years of tedious experimentation in the laboratory in order to discern the physical and chemical environments that are conducive to such growth. Proteins and other macromolecules tend to be quite specific in terms of their response behaviors, making it difficult to develop very general crystallization protocols that have high rates of success with respect to obtaining good crystals in a reasonable amount of time and without expenditure of lots of material, namely protein.

Generating and collecting data for the purposes of analysis is the most important step in any scientific study. Currently, insufficient real data (from actual experiments performed in the laboratory) are available to warrant important conclusions using the framework described in this dissertation. Therefore, considerable effort has been invested into the creation of a predictive model that is representative of hypothetical protein response behavior under varying experimental conditions. The model has been designed with a fairly flexible representation to enable easy tuning for generating different scenarios for reflective evaluation.

The simulator of hypothetical macromolecules developed and evaluated in Chapter 5 could be used for planning actual experiments in the laboratory. Figure 1.1 depicts the general idea of how the PEP framework with the simulator and generalization program could be used in order to reduce the amount of resources, time and effort spent on crystallizing a new protein in the laboratory. The assumption is that we can describe as input to the simulator, solubility tables that describe the

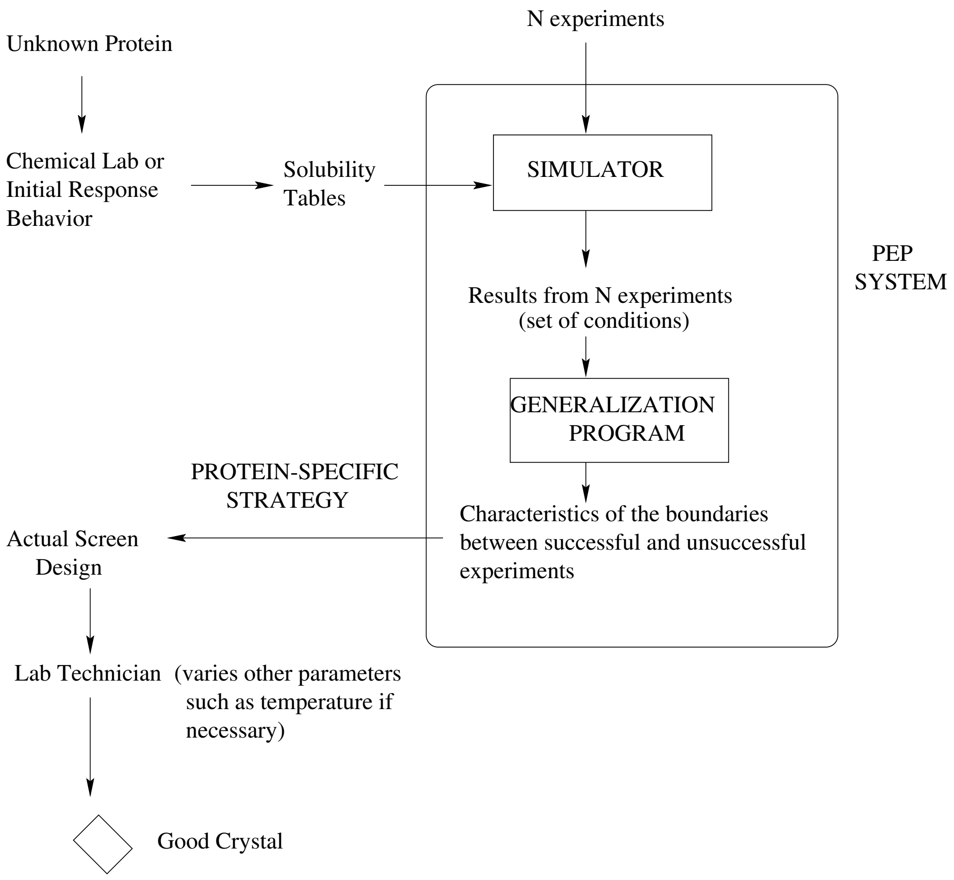


Figure 1.1: The general idea for providing intelligent assistance

solubility[2] of each solid phase of a protein with respect to individual control variables such as the concentrations of a particular precipitating agent, or salt. The output from the simulator is a real number that represents the quality of the best result obtained from an experiment. This number is easily translated into an observable class such as clear[3], crystal or precipitate that represents an observation of an experiment at a particular time in the laboratory under a microscope.

Thus, even with an approximate simulator that can model interactions within a reduced set of dimensions that exist in the real-world, it is possible to run several thousands of simulated experiments and obtain results at any time of observation. These results can be analyzed using a generalization program to learn the general characteristics of the boundary between successful and unsuccessful experiments. The boundary characterizes the values of control variables that are more likely to yield crystals in the laboratory. These could be used as initial conditions by crystallographers, who can then vary other parameters that were not represented in the model to obtain a good crystal of the new protein.

The problem of macromolecular crystal growing is viewed as heuristic search over a parameter space. This is the most natural representation of the parallel experiment planning problem as each experiment is independent and is not influenced by any other experiment. The task is then to be able to learn a good heuristic evaluation function that can get us to the goal of being able to grow a macromolecular crystal given some resource constraints. Due to the complex nature of the problem and the non-existence of heuristic rules that can be expressed computationally, we are faced with the problem of learning a suitable heuristic evaluation function by trial-and-error experimentation.

We already know that experimentation in the laboratory is both tedious and time-consuming. This research develops an approximate model of the process of macromolecular crystal nucleation and growth that can be used to simulate experiments. This model is used within the framework shown in Figure 1.2. This figure depicts a model of the environment for planning parallel experiments that has been developed after studying the way in which experiments are performed in the crystal growing domain. The environment depicts the kinds of feedback that could be made available to an agent (human or other) in order to facilitate learning that could help in future problem-solving.

The parallel experiment planning (PEP) framework is designed to be open-ended in terms of the kinds of analyses that can be performed with the partial data that are collected. Statistical summaries of partial results are certainly a major aid to decision making. Also, inherent in this framework is the ability to keep track of resources that are being consumed by each experiment. This information is very useful to the agent (person) making the decision as to whether to design a new trial, and if so, how many experiments to perform, or whether to wait and observe one or more

[2] Solubility of a solid phase of protein is defined as the concentration of the protein in equilibrium with solid phase of interest given ideal thermodynamic conditions.

[3] no visible change in solution

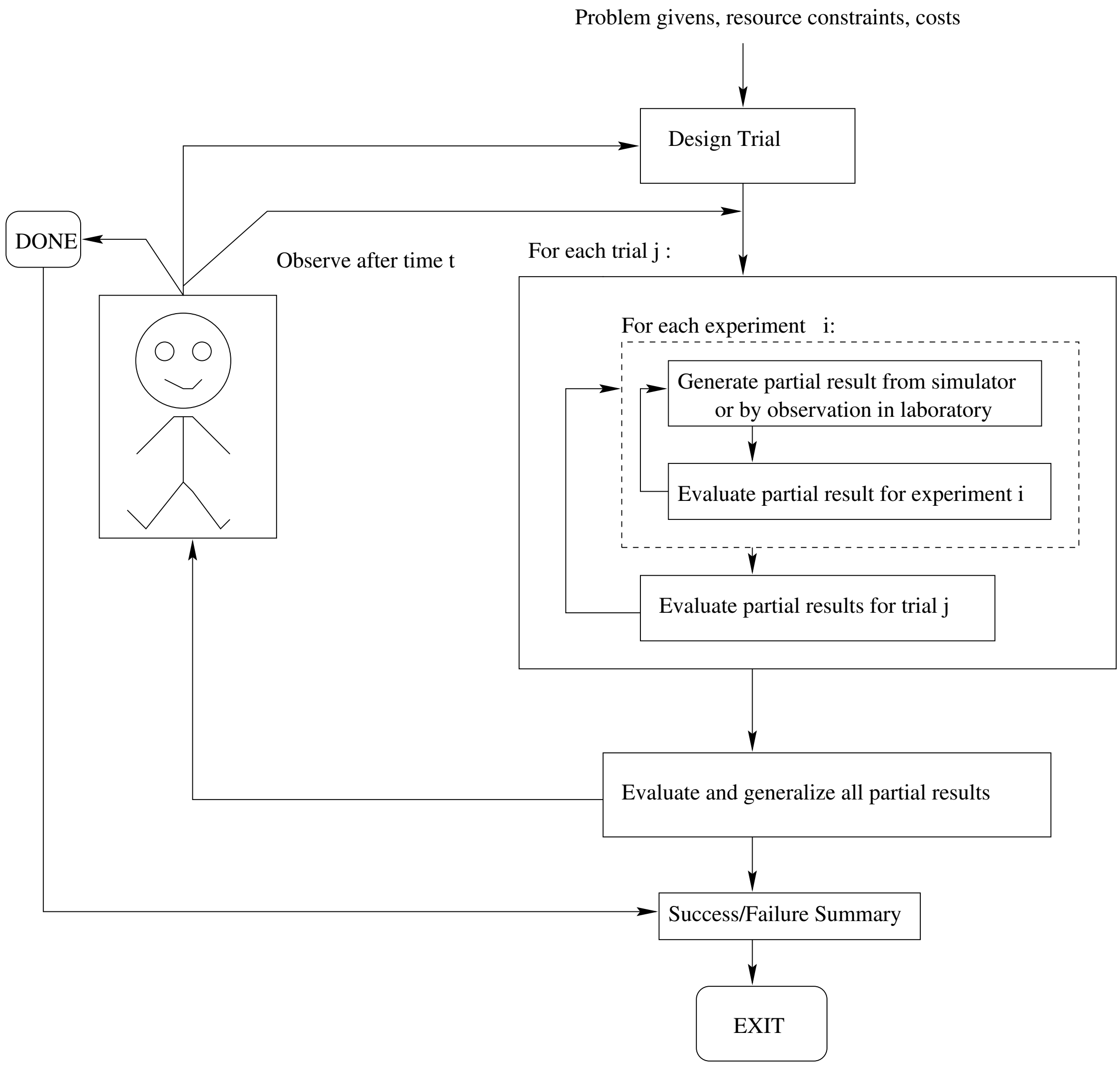


Figure 1.2: Framework for planning parallel experiments via multilevel evaluation of partial results

times before designing a new trial. In the future, a computer could perform the learning and actions that a human agent currently performs within this PEP framework.

1.2.1 Theses

The central thesis is that the framework shown in Figure 1.2 is sufficient for planning experiments in parallel in scientific domains in which

1. evaluation of partial results is imprecise,
2. effects of actions change over time,
3. resources are limited, and
4. tradeoffs are necessary in order to successfully search a very large parameter space.

Secondary theses are the following:

1. In order to be able to perform reflective analyses of the different parallel experimentation scenarios, it is necessary to maintain and manage information related to experiments. This dissertation shows the necessity for and usefulness of different levels of summaries of partial results.
2. The framework includes an agent, currently human, that acts as the heuristic control for determining which portion of the search space to expand next, and with what granularity. We assume that the human agent in the loop will be able to learn global strategies that achieve the goal, based on several runs of the PEP prototype, on different types of hypothetical protein response behaviors or scenarios. The user can therefore learn global strategies for one or more classes of (hypothetical) proteins by actually trying them out within this framework, and seeing the costs and benefits of performing them using simulated experiments.
3. In order to generate different scenarios it is necessary to generate and evaluate different models that act as the local evaluation function. A model for generating response behavior of a hypothetical protein in different physico-chemical environments is designed and developed. This model is then modified to generate different types of response behaviors.

I will now restate the main thesis as two claims, one weak and the other strong.

Weak Claim

A Parallel Experiment Planning (PEP) system is developed that provides a computational representation and set of tools to manage information about parallel experiments (or trials).

Strong Claim

A Parallel Experiment Planning system is developed that can provide intelligent assistance for decision-making by suggesting likely places in a search space for new trials and portions of the space that may be closed because they are unlikely to yield results.

1.3 Significance

From a computer science perspective, the framework is an abstraction of a method to perform parallel experiment planning, which has been extracted from observations of experiment planning in a real-world domain with a very large parameter space and few heuristics that can guide the search. The framework provides mechanisms for learning more heuristics that can then be used for computing prior probabilities of success of different policies for deciding what to do next depending on the evaluations of partial outcomes. This learning is currently enabled by human trials - a learning by doing approach.

This work may also be viewed as including reinforcement learning wherein there exists an agent that tries to learn a policy that can achieve its goals by manipulating a complex environment and receiving feedback from it. Much of prior work in reinforcement learning in artificial intelligence uses only a real-valued feedback or reward in sequential probes of the environment ([65, 66, 6, 41]). Within the framework developed here, an agent receives more than one type of feedback, which includes a set of symbolic rules, and receives feedback from multiple probes of (copies of) the same environment. Symbolic feedback amounts to a description of the boundary between observed classes of partial results that can be utilized to prune the vast search space, and thus is more useful than point values.

Reinforcement learning in complex decision-making domains such as macromolecular crystallization, requires computational representations of two types of objects: the environment and the agent. The agent senses its environment through percepts and manipulates it through effectors. The environment is a model that provides a relatively good approximation of the real-world. Here we can immediately see that the big advantage of such computational representations is the ability to be able to design different agent functions as well different environment functions to be able to understand both the environment better as well as the types of agents (or policies) that work well, in given situations.

I have constructed a model of a complex real-world environment and identified the types of decisions that a human-agent would have to perform in order to reach the goal. Sufficiency of the PEP framework for parallel experiment planning is shown by computational implementation of a prototype that simulates the parallel experiment planning environment. The prototype PEP system has human decision-making agents who can manipulate the environment to achieve their goals. The design and development of an agent simulator, or a computational agent function would be the scope of future work.

Even though there is an optimal answer in theory, which is the equivalent of being able to pick one experiment that is the solution to a given problem (within resource constraints), in reality, all that one desires is to be able to reach the goal without violating the constraints. Thus we can observe that there are several answers to the problem of getting to the goal without violating the constraints, and each answer varies depending on the agent function. It has been noticed that some people are better at obtaining crystals than others. Thus, it would seem that we would be interested in learning how to represent these different agent types, and which of their strategies are more successful and why.

From a domain perspective, this research provides an approximate model of the macromolecular crystallization process, and illustrates its use in the design of hypothetical proteins with different response behaviors under different physico-chemical environments over time. By incorporating such a model within the framework for parallel experiment planning, a novice crystallographer can be made aware of several subtleties that arise in the design of experiments in crystallization. By incorporating cost information and resource constraints into the overall framework, it is possible to actually evaluate the effect of tradeoffs at each decision step of a crystallization project.

The model of macromolecular crystal nucleation and growth developed in this dissertation is a simplified model. In order to evaluate it correctly, we will need to enter solubility information from either the literature or actual experimentation. This would be part of future work from a domain standpoint. In this dissertation, I have shown that it is feasible to produce some flavor of different hypothetical protein response behavior using the model developed in Chapter 5.

1.4 Dissertation Overview

Chapter 2 provides an overview of the problem domain and the necessary background for discussing the computational aspects of the framework. Chapter 3 provides an annotated example of the capabilities and features of the prototypical implementation of the PEP environment where the agent is human, and manipulates the environment as a tool. The inputs to and outputs from the framework implementation are shown verbatim, except they are compressed in the form of tables to facilitate reading. Chapter 4 provides information on the data structures and methods used to describe and implement the various components of the framework. In order to test the framework as well as provide intelligent assistance for decision-making, it was necessary to construct an approximate physical-chemical model that can generate different response behaviors of hypothetical proteins over time. This elaborate model is described in Chapter 5. Chapter 6 presents arguments and the main experiments performed to demonstrate the central thesis. Chapter 7 discusses conclusions and future work in both domains of computer science and macromolecular crystallization. The domain knowledge, such as the solubility tables used in the model for generating protein response behavior that resembles the real enzyme lysozyme, is depicted in Appendix C.

Chapter 2
Background

2.1 Overview of Macromolecular Crystallization

Crystallization is an essential first step in macromolecular 3-D structure determination by X-ray crystallography. This is the only method capable of revealing high resolution structures for most proteins, protein-DNA complexes, viruses etc. The high resolution structural information is critical for modern molecular biological methods which rely on knowledge of the geometrical interrelationships of the various components that comprise the overall structure. (Multidimensional NMR can also determine macromolecular structures, but it is limited to molecules whose molecular weight is under 20,000; most proteins are larger than that).

The rate limiting step in X-ray structure determination is the crystallization itself. It takes anywhere between a few weeks to several years[1] to obtain macromolecular crystals that yield good diffraction patterns. The theory of forces that promote and maintain crystal growth is preliminary, and crystallographers systematically search a large parameter space of experimental settings to grow good crystals. The process is described by [43] as follows:

> This [macromolecular crystallization] is done by conducting a long series of crystallization trials, evaluating the results, and using information obtained to improve matters in successive rounds of trials. Because the range of variables is so broad, intelligence and intuition in designing and evaluating individual and collective trails becomes essential.

A set of about twenty-two parameters (e.g, temperature, pH - see Appendix A for complete list) that determine success in crystal formation have been empirically identified. Crystallization attempts begin with the design of an initial screening experiment that coarsely samples the parameter space with a small number of parallel probes (typically between 50 and 100). The initial experiments usually incorporate only general information about crystallization combined with some specific details regarding the individual molecule under consideration (stability data, solubility data,

[1] In fact, the question of how long to wait for crystals or precipitate to appear in a cell is not easy to answer. Anecdotes abound on experimenters finding crystals in plates abandoned for years in the laboratory.

isoelectric point, etc). One approach begins with a very coarse, uniform grid that spans the variables of interest [18]. Another begins with an incomplete factorial design that randomly samples the variables such that all pairwise combinations are tested [13]. The results from the initial screens are used to design finer sampling strategies for subsequent rounds of crystallization trials.

Rational parameter sampling in the iterated experimental protocol is made difficult by the need to run experiments over several months as well as by the non-linear behavior of macromolecules with respect to the different parameters. The length of time between iterations makes it a challenge for humans to remember contextual information. Crystallographers presently rely on paper logs of experiments for this purpose; however the types of access they support are very limited. In order to facilitate the capture of experimental information electronically, an effort has been made to build an electronic laboratory notebook[32, 35]. Even though the software XtalGrow [58] is currently being used by about 10 laboratories in the United States, we are still faced with the problem of insufficient data for purposes of analysis. Also, the software is still evolving.

The main source for data about successful crystallizations is the Biological Macromolecule Crystallization Database (BMCD) [27]. This database captures information about successful experiments only. Several attempts have been made to exploit the available data in order to design initial crystallization trials for unknown macromolecules. Samudzi et al. [62] performed a cluster analysis on version 1.0 of the BMCD and suggested a set of screening conditions specific to a major class of macromolecules. In Gopalakrishnan et al. [31], preliminary attempts were made to recreate the clusters obtained in Samudzi et al. using two kinds of methods – statistical analysis (same as Samudzi's) and COBWEB [45] (a machine learning and discovery program). The results from the clustering analysis were then used as input to the RL [15] inductive rule-learning program (see section 2.5.2), resulting in verification and expansion of Samudzi's results. Hennessy et al. [34] augmented the BMCD with a hierarchical classification of the macromolecules contained therein, as well as data on the additives used with them, and performed a statistical analysis that has led to a Bayesian technique for postulating the degree of success of a set of experimental conditions for a new macromolecule belonging to some known class.

2.1.1 An Experiment in Macromolecular Crystallization

The basic crystallization experiment is to slowly reduce the solubility of a sample solution of the macromolecule by one of several established methods[20] (see Chapter 5 for details). The solubility is determined by all the environmental parameters, one of which is usually the concentration of a "precipitating agent", such as polyethylene glycol (PEG), a commonly used precipitant. The "crystallization method", such as *vapor diffusion*, slowly raises the concentration of the precipitating agent (and almost everything else). If all the conditions are favorable, a point is reached where a crystal nucleates and grows.

The basic experiment is repeated with different parameters until the experimenter succeeds, abandons the effort entirely, or decides to work on crystallizing a mutant or variation of the original macromolecule. Typically, many experiments (between 50 and 100) are started simultaneously and allowed to run for several weeks to several months. During this time, the experimenter attends to other projects. Then, the results are evaluated and a new series begun (to run concurrently with the older ones). Thus, large volumes of data accumulate over long periods of time. A slow rate of change is very often essential for crystal growth, thus there may be no other way to speed the process than to find good environmental parameters quickly.

An experiment can be described as containing values for three sets of variables, *givens*, *controllables* and *observables*. *Givens* represent the known information about a protein such as its name, molecular weight, and isoelectric point. *Controllables* represent the values for the control parameters such as the concentrations of protein and precipitant, pH, and temperature. The term *observables* is used to denote the vector of partial results that are observed at different times. The givens determine the observable effects of the choice of controllables. As the observables change over time, they are referred to as partial results at any particular time of observation. This classification of the variables as givens, controllables and observables helps us understand the manner in which the different types of variables influence one another.

Figure 2.1 gives a pictorial description of how experiments and trials are done by hand. Each experiment is shown as a circle within a tray that can be used to set up a maximum of 24 experiments. Several trays are set up as the initial trial or set of parallel experiments. Each experiment includes some measures for control variables such as protein, precipitant and salt concentrations. Each well or experiment is examined under a microscope to yield observables over time. Depending on partial results observed, new trials are started and examined, until a good quality crystal is obtained.

2.1.2 Motivation for the Framework

The above description of how experiments are actually designed and performed in parallel forms the basis for the abstracted PEP framework. Because there are insufficient data regarding both successful and unsuccessful experiments (and the degree to which they are successful/unsuccessful), we needed a framework within which we could study the feasibility and requirements for capturing and analyzing experimental data. One commonly adopted method for such a study is modeling and simulation. The research described in this dissertation is a direct outcome of interaction with domain experts, particularly John. M. Rosenberg, and includes the construction of an approximate

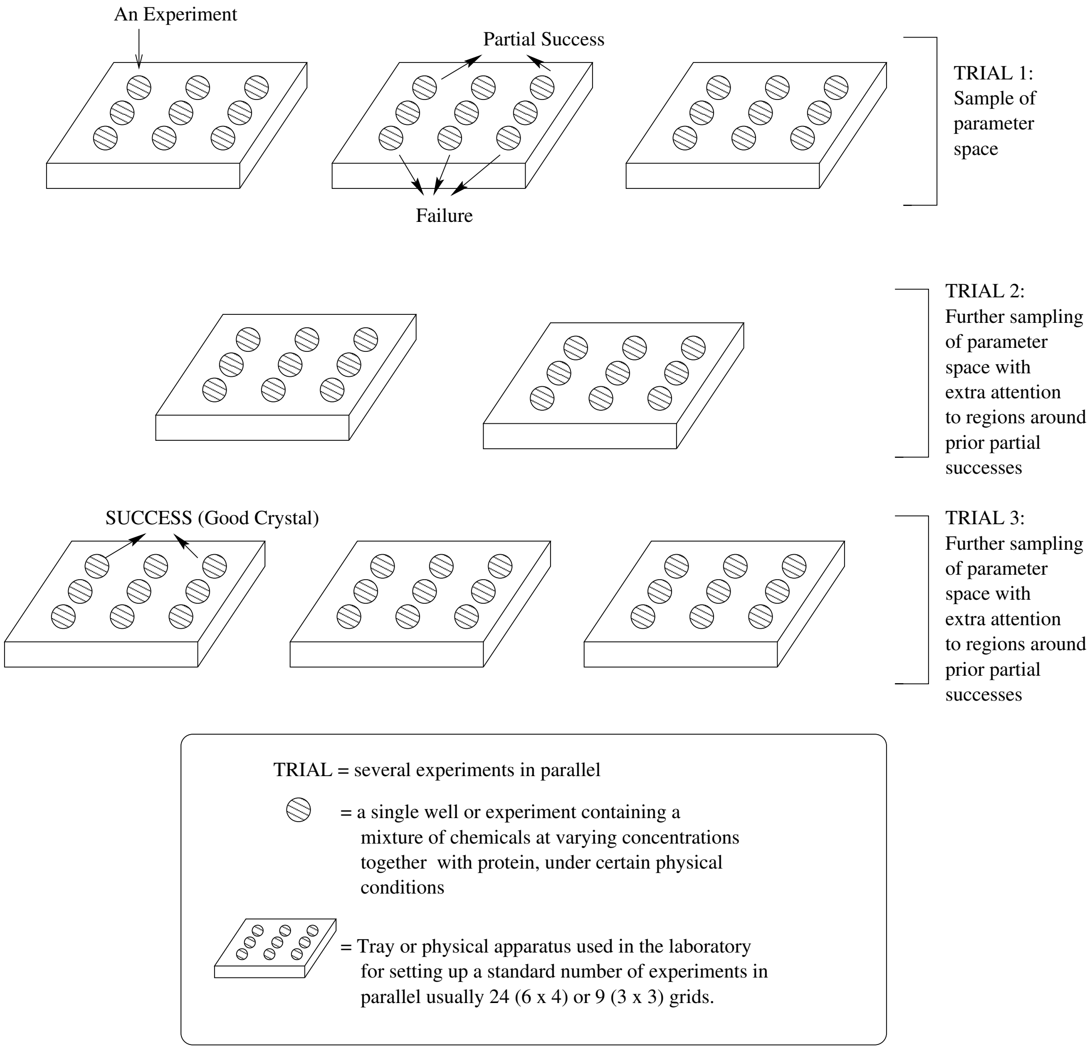


Figure 2.1: Parallel experimentation in macromolecular crystallization. After each trial is set up: success = good crystal, partial success = precipitate or unusable crystal, and failure = clear or no change in solution

physico-chemical model of crystal nucleation and growth. Most of the validation for the performance of the model was subjective, with the satisfaction of the domain experts as the major goal.

Figure 1.1 in Chapter 1 depicts the ideas that form the basic motivation for this research. When a crystallographer is faced with an unknown protein with only a few given information, such as its molecular weight and its isoelectric point or pI[2], the goal is to obtain at least one crystal[3] that is good for X-ray diffraction as soon as possible and without running out of protein material. A good quality crystal is usually one that has a resolution limit of diffraction (difflim) less than 3 Å. In special cases, there exists only a limited amount of protein (such as when a patient is dead, and is the only source of a particular protein). Also, protein isolation and purification is an expensive process. The major factors that limit the number of experiments that could be performed in the laboratory therefore are the amount of protein and the tedious nature of experimentation. Thus, any mechanism or method that would be able to help in cutting down the the number of experiments that we would try before getting at least one hit (good crystal), would be most helpful to crystallographers. The research undertaken as part of this work attempts to lead us toward such methodology.

The method shown in Figure 1.1 attempts to use preliminary solubility data available for an unknown protein that needs to be crystallized, as input for a model that could be used to simulate experimental outcomes over time. The space of experiments defined within the simulator model consists of a reduced set of factors that could possibly influence the outcomes. If the assumptions that are made in the model hold in the real-world, we would then be able to predict the experiments within this reduced space that are more likely to yield a good quality crystal. These identified experimental conditions could then be further manipulated in the laboratory, if necessary, by varying parameters that are outside the scope of the simulator model. Thereby, using existing theory and a model based on such theory to predict the likelihoods of success for experiments along some dimensions of the search space, we could reduce the total number of experiments that need to be performed in the laboratory before a single good crystal of any given protein is obtained.

We describe the development and workings of a predictive model of approximate protein behavior in commonly-used physico-chemical environments. That is, using this model we can predict the outcome of one or more experiments that contained a particular set of values for the physical and chemical parameters that influence crystal growth. What would be the type of input parameters for such a model, given that the output would be some quantitative indicator of observable result and its quality? One logical answer would be to use preliminary information that is already known regarding the solubility of the unknown protein in some particular reagents. The model would then

[2] At its isoelectric point, a macromolecule carries an equal number of positive and negative charges, and is therefore electrostatically neutral or balanced. It had been found that this is the best state for mutually attractive interactions. Solubility is found to be at a pronounced minimum at their pI values.

[3] commonly depicted as Xtal. Crystalline phase is also expressed as Xtalline phase for purposes of this dissertation.

need to make some assumptions about how the different parameters react in solution, and under what situations a macromolecular crystal is likely to nucleate and grow. Some of these assumptions could be well grounded in reality based on the experience obtained over years of laboratory work and observation of these phenomena, as well as from well-known theories ([44, 20, 59]) that exist in the literature.

It should be noted that some preliminary information comes along with a new protein that needs to be crystallized – such as its isoelectric point or pI and macromolecular weight – and also maybe some response information based on some preliminary screen designs. It is foreseeable, that we may be able to use such response behavior to reverse engineer or hypothesize the likely information that should go into defining the solubility table information needed by the simulator.

Given that it is possible to construct such a simulator based on a predictive model of macromolecular crystallization, it is then possible to try out any number of experiments in virtual space to understand the characteristics of the boundary that separate the successful experimental conditions from the unsuccessful ones. Identification of boundary is an important aspect of being able to come up with some internal model of how the given protein seems to be responding under different experimental controls. This gives some insight into possible rates of change of various hidden variables that influence crystal nucleation and growth such as the saturation level of protein in solution. A generalization program (see section 2.5) is used within the framework to provide a human-understandable description of the boundary based on all the experiments and partial results observed so far. By using the framework with the simulator and generalization program to learn such protein-specific strategies to try out in the laboratory, we hope that we will be able to reduce the number of actual experiments tried before obtaining a good quality crystal. Using this method we would be able to save time, effort and material involved in the crystallization of an unknown protein.

The model for simulating hypothetical macromolecule response behavior presumes that it is possible to elicit solubility information of solid phases of a protein in individual chemical reagents (for example, concentration of PEG 8000) using very little protein (less than a milligram on a needle tip is sufficient). The type of solubility information that can be obtained by such direct experimentation in the laboratory is incomplete in the sense that only *turbidity*[4] can be studied. Thus, we know that some kind of precipitation occurs for a given new protein at some concentration of the precipitating agent (for example), but we do not know the characteristics of the solid phase (such as whether it is an amorphous solid mass or crystalline). Nevertheless, it would be possible to infer and construct such solubility table information that could then be used as input to the simulator of the new protein response behavior.

[4] Turbidity is a measure of how much light passes through any solution. Thus, if turbidity is 1, it means no light passes through. Turbidity is 0 for a clear solution, as all light passes through.

We have constructed a fairly complex model of the crystallization process, and have developed a framework that allows different strategies, with varying numbers of parallel or simultaneous experiments, to be tried and evaluated. The model uses solubility graphs that have been developed with the help of our domain expert, Prof. John M. Rosenberg. The solubility graphs that are used in our experiments are modeled fairly closely to those available in the literature and from experimentation at the laboratory. For the purposes of this dissertation, we rely solely on these constructed graphs. The information from the graphs are translated into compact descriptions of mathematical functions with tables for parameters. These tables are called solubility tables and are used as the main domain knowledge that is input to the simulator.

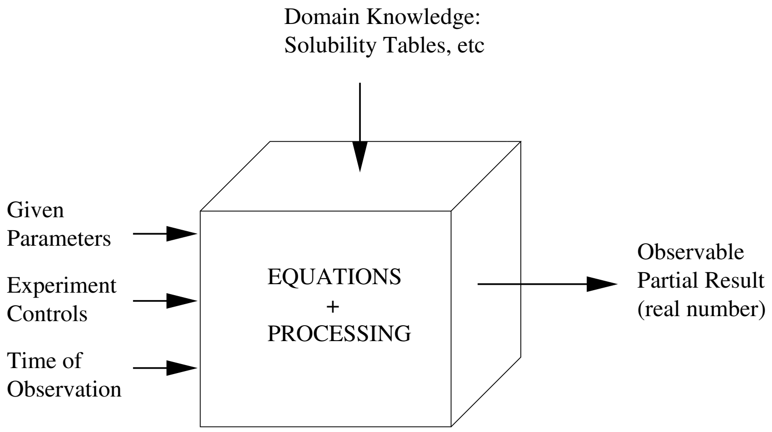


Figure 2.2: Overview of simulator model

Figure 2.2 presents an overview of our model used to simulate experimental observables for any hypothetical protein described by the set of solubility tables, equations and processing as developed in Chapter 5. The inputs to the simulator include domain knowledge in the form of solubility tables that represent mathematical functions that describe the interactions between solubility of particular solid phases of the protein and individual chemical reagents such as the concentration of a particular salt or precipitating agent. Other inputs to the simulator include a description of protein itself in terms of its given parameters, such as its isoelectric point and saturation conditions for nucleation and growth of its different solid phases. Appendices B and C describe simulator inputs in more detail. The output from the simulator is a real number that is representative of the partial result observed at a particular time. This number is typically used to represent one of three broad classes of observables: clear, crystal or precipitate. The model that consists of equations and processing used to convert the inputs into one output real number is described in detail in Chapter 5.

2.2 Overview of Simulation and Modeling

The development of models for performing an analysis to determine the costs and benefits of actions under different scenarios has been undertaken in the past. Experimental design and simulation go hand in hand. The authors of SIMAN [50] give a thorough introduction to simulation and modeling which forms the basis for the analysis of complex systems. In the context of the PEP framework, we have identified the need for collection of information pertinent to experimentation that is performed in parallel.

Computer simulation is a widely-adopted technique for understanding complex phenomena or behavior of real-world problems. It is a "learning by doing" approach where actual material resources need not be used, but can help further the understanding of interacting components within a model built to reflect the real-world situations. The first step toward a computer simulation is the building of a model that approximates the problem domain. This usually entails the involvement of several disciplines, particularly those of computer science and the problem domain. Current day simulations are highly sophisticated in their levels of interactions with the user and types of animations and graphics that can be portrayed.

The focus of this research is on the elicitation and construction of a model that approximates the process of macromolecular crystal growth. The choice of implementation language was ANSI-C, since it is portable and our aim was to determine the feasibility of eliciting computational behavior that resembled that of some real-world protein.

2.3 Heuristic Search

In artificial intelligence (AI), search is the general term used to describe the process by which a solution is arrived at for a given problem. A problem is formulated based on the goal, and consists of four parts: the initial state[5]; a set of operators from which one or more operators can be applied to get to the next state from the current state; a goal test function that is used to decide whether the search should terminate; and a path cost function. The space of all possible states of the problem is referred to as the state space. A solution to a problem is either a path through the state space from the initial state to a goal state or a detailed description of the goal state. There are several search strategies such as depth-first search and breadth-first search that are general search strategies that perform complete search and hence are guaranteed to find a solution if one exists. AI search strategies are generally heuristic in nature.

Since most real-world problems and problems that are considered in AI are NP-hard (exponential time/space solutions exist, no polynomial solutions) [26], heuristics are used to prune search

[5] State refers to some description of the world.

spaces and hence reduce search cost. The best-first search technique uses heuristics to order the search in such a manner as to take the action that leads to the best possible next state according to the criteria used to describe "best", which are usually based on an estimate of the cost of the path from a state to the closest goal state. The function that is used to order the nodes in the search is called the evaluation function, and denotes a measure of estimated distance to the goal. A node is a data structure that represents a state in the state space. The search proceeds by visiting nodes along some sequence. Nodes have other information that correspond to their position in the search tree, the path cost of the path from the initial state to that node, the parent node and the operator applied to generate this node. For a more detailed description of heuristic search in AI, see Pearl's textbook called *Heuristics* [49].

Heuristic search algorithms such as simulated annealing and tabu search are popular methods for solving combinatorial optimization problems. These methods fall into the category of iterative improvement algorithms and are used in problem domains where the path by which the solution is reached is irrelevant. The state description itself contains all the information needed for a solution. Hill-climbing or gradient descent algorithms are one kind of iterative improvement algorithms, where the search starts from an initial state and keeps moving in the direction of increasing value or decreasing cost. The main problem with hill-climbing (gradient descent) algorithms is that they could get stuck in local maxima (minima). Simulated annealing[37] is another type of iterative improvement algorithm that provides a way to escape local maxima. Instead of picking the best move, it picks a random move. If the move actually improves the situation, it is always executed. Otherwise, the algorithm makes the move with some probability less than 1. The probability decreases exponentially with the "badness" of the move, that is the amount by which the evaluation is worsened. The concept used in tabu search methods is to penalize moves which take the solution, in the next iteration, to points in the solution space previously visited. A recent book by Glover and Laguna [28] describes these techniques in detail. Recently, there has been a lot of interest and research in parallel heuristic search methods that are parallel versions of these algorithms (see Azencott's book on parallelization techniques for simulated annealing [4]).

The PEP framework is a heuristic best-first search technique where the human agent acts as the evaluation function. The human agent is provided with intelligent assistance from the simulator and generalization program, where the simulator acts as an approximate evaluation function for a reduced set of parameters for search, and the generalization program identifies likely and unlikely places for obtaining a good crystal based on simulated experiments.

Experiment design is usually viewed as a systematic search procedure. In the real-world, we have constraints on resources, and by experience, we are able to elicit paths or plans that could get us to the goal. Implicit in this experience is the knowledge about how to analyze partial results and how to navigate further toward the solution. This knowledge constitutes a complex evaluation function that is constantly changing as new pieces of information is added based on new experiences. Paths therefore translate to protocols that one could use in the laboratory for future experimentation. Within the PEP framework, the human agent could try different paths from the initial state to the goal state, and learn cost-effective strategies by observing the associated costs and effects.

2.4 Agents and Reinforcement Learning

An agent is basically anything that can *perceive* its environment through sensors and *act* upon its environment through effectors. A human agent has eyes, ears and other organs for sensors, and hands, legs, mouth, and other body parts for effectors. A robotic agent substitutes cameras and infrared image finders for sensors and various motors for the effectors. A software agent has encoded bit strings as its percepts and actions. Figure 2.3[6] depicts how an agent interacts with its environment through sensors and effectors. The term "percept sequence" is used to denote everything that the agent has seen so far, that is its complete perceptual history. Ideally, an intelligent agent should perform whatever action is needed in a particular problem-solving situation on the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

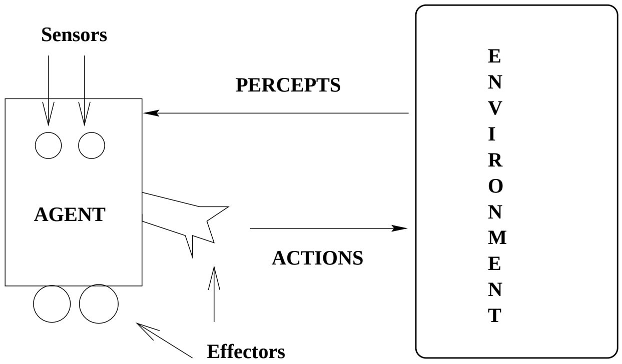


Figure 2.3: Agents interact with environments through sensors and effectors

In reinforcement learning, an agent explores its environment and learns from success/failure. In games such as tennis, immediate feedback or rewards are available, such as number of points won. The task of reinforcement learning is to use rewards to learn a successful agent function. An agent can be a passive learner or active learner. A passive learner simply watches the world go by, and tries to learn the utility of being in various states, an active learner must also act on the learned information, and can use its problem generator to suggest explorations of unknown portions of the environment.

[6] The figure has been adapted from Russell and Norvig's textbook for Artificial Intelligence [61].

In complex decision making environments such as in the real-world, decision problems are usually concerned with complex actions undertaken sequentially. Such sequential decision problems include uncertainty and sensing. These complex problems generalize the classic problems of AI search and planning [61]. The solution to such problems is described by a policy, that is, a set of situation-action rules for each state, and is usually arrived at by calculating utilities[7] for each state. A policy is therefore a complete mapping from states to actions. Reinforcement learning could be used to learn an optimal (or good) policy or to evaluate a fixed policy.

Two basic designs for agents that perform reinforcement learning have been described in the literature[61] that reflect two differences in the way utilities are calculated by the agents. One type of agent learns a utility function on states (or state histories) and uses it to select actions that maximize the expected utility of their outcomes. The other type of agent learns a function from actions to values that gives the expected utility[8] of taking a given action in a given state. The latter type of agent that learns action-value functions performs a kind of learning called Q-learning. Such learning agents can be simpler in design than utility learners, since they do not require a model of the environment in order to make decisions. These learners need to only know the legal moves, and can directly compare the values of the different actions without having to consider their outcomes. An agent that learns utility functions must have a model of the environment in order to make decisions, because it must know the states to which its actions will lead.

This section is brief, but simply aims to describe the research area of reinforcement learning, which would be what the agent that is within this framework of PEP would be doing. Given that this agent is a human-being, he/she is able to see the effects of actions based on either a simulated model or real-world experimentation, and is constantly learning to revise their internal model of the nature of the protein-specific experimentation.

2.5 Overview of Machine Learning Algorithms

Generalization is necessary within the PEP framework for two important reasons:

1. the need to distinguish the boundary between successful and unsuccessful experiments with respect to the values of the control variables, and

[7]An agent's preference between world states are captured by a utility function, which assigns a single number to express desirability of a state. Utilities are combined with outcome probabilities for actions to give an expected utility for each action.

[8]A nondeterministic action A will have possible outcome states $Result_i(A)$, where i ranges over the different outcomes. Prior to the execution of A, the agent assigns probability $P(Result_i(A)|Do(A), E)$ to each outcome, where E summarizes the agent's available evidence about the world, and $Do(A)$ is the proposition that action A is executed in the current state. Then we can calculate the **expected utility** of the action given the evidence, $EU(A|E)$, using the formula:
$EU(A|E) = \sum_i P(Result_i(A)|E, Do(A))U(Result_i(A))$.

2. the difficulty for a human to eyeball several hundreds of experiments and observations in order to identify this boundary.

In this domain, since several experiments are performed in parallel and observation effects change over time, it is necessary to maintain a human-understandable compact representation of the evolving boundary between the classes of observables in terms of controllable values. A class of generalization algorithms called symbolic inductive learning algorithms are able to take as input data sets of feature[9] vectors and output human-understandable models expressed as a disjunction of rules that contain a conjunction of features on the left-hand side and a classification on the right-hand side.

Machine learning from examples, or inductive learning, has received considerable attention since the 1950's, with several approaches now in the growing tool kit of knowledge-based programs that have been demonstrated to work[12]. In this section, I will describe two methods for inducing disjunctive expressions consisting of conjunctive features, to classify training instances into one of several discrete target classes. The methods perform almost equivalently, even though the measures used for classification vary.

2.5.1 Decision Tree Learning

One of the most widely adopted methods for inductive learning from pre-classified training examples is decision tree learning. This method is robust to noisy data and is capable of learning disjunctive rules/expressions that approximate discrete-valued functions. The learned function is represented by a decision tree that can also be represented as a set of if-then rules. Decision trees classify instances or examples by assigning them to a leaf node that provides the class (or classification) in a tree by traversing down from the root node. Each node in the tree specifies a test of some attribute of the instance (example), and each branch descending from that node corresponds to one of the possible values for this attribute[46]. Classification of an instance then proceeds as follows:

1. Start at the root node, and test the attribute specified by the node.

2. Move down the tree branch that corresponds to the value of that attribute in the instance that is being classified.

3. Until leaf node reached and instance is classified,
   Repeat steps 1 and 2 for subtree rooted at the new node.

A simple example of the decision tree for the concept *GoToSchool* is shown in Figure 2.4. This decision tree represents the type of classification trees that are typically learned. This decision

[9] A feature is described as an (attribute,value) pair. Example of features are (Shape, triangle) and (Shape, rectangle), where the attribute *Shape* can take on values such as triangle, rectangle and so forth.

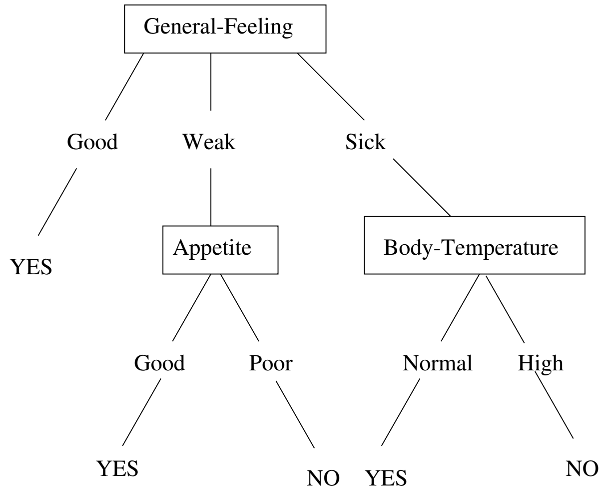


Figure 2.4: A decision tree for the concept *GoToSchool*

.

tree classifies children based on their current symptoms as to whether they are alright to go to school. Decision trees in general, represent a disjunction of conjunctions of constraints on attribute values of instances. For example, the decision tree in Figure 2.4, corresponds to the following expression:
(General-Feeling = Good) $\vee$
(General-Feeling = Weak $\wedge$ Appetite = Good) $\vee$
(General-Feeling = Sick $\wedge$ Body-Temperature=Normal)
This example demonstrates concept learning, where the target class has Boolean value (YES or NO). The earliest decision tree learning algorithm is ID3 [56] that employs a top-down, greedy search through the space of possible hypotheses (decision trees). ID3's successor C4.5 [57] is one of the most popular decision tree learning programs used for various classification tasks in the real world. The central choice in the ID3 algorithm is selecting which attribute to test at each node in the tree. ID3 uses a statistical property, called *information gain*, that measures how well a given attribute separates the training examples according to their target class. This measure is used by ID3 to select among candidate attributes at each node while growing the decision tree. The information theory that lies behind the information gain criterion can be stated as follows: The information conveyed by a message depends on its probability and can be measured in bits minus the logarithm to base 2 of that probability. So, for example, if there are eight equally probable messages, the information conveyed by any one of them is $-log_2(1/8)$ or 3 bits.

If we imagine picking one example at random from a set S of examples and announcing that it belongs to some class C_j. This message has probability[57]:

$$\frac{freq(C_j,S)}{|S|}$$

and so the information it conveys is

$$-log_2\left(\frac{freq(C_j,S)}{|S|}\right)\text{ bits.}$$

To find the expected information from such a message pertaining to class membership, we can sum over the classes in proportion to their frequencies in S, giving

$$info(S) = -\sum_{j=1}^{k}\frac{freq(C_j,S)}{|S|}\times log_2\left(\frac{freq(C_j,S)}{|S|}\right) bits. \tag{2.1}$$

When applied to a set T of training examples, înfo(T) measures the average amount of information needed to identify the class of a case in T. This quantity is also known as the *entropy* of a set. Now, suppose we consider a similar measurement after T has been partitioned in accordance with n outcomes of a test X. The expected information requirement can be found as a weighted sum over the subsets, as

$$info_X(T) = \sum_{i=1}^{n}\frac{|T_i|}{T}\times info(T_i). \tag{2.2}$$

The quantity

$$gain(X) = info(T) - info_X(T) \tag{2.3}$$

measures the information that is gained by partitioning T in accordance with the test X. The information gain criterion in the ID3 algorithm, then, selects a test to maximize this information gain. Figure 2.5 presents the decision tree learning algorithm. Let us consider the training examples given in Table 2.1, and calculate the information gain measures at each decision node while growing the tree.

General-Feeling	**Appetite**	**Body-Temperature**	**GoToSchool**
Good	Good	Normal	YES
Good	Poor	Normal	YES
Good	Good	High	YES
Weak	Poor	Normal	NO
Sick	Poor	High	NO
Weak	Good	Normal	YES
Sick	Poor	Normal	YES

Table 2.1: A small set of training examples from which decision tree in Figure 2.4 was grown

Let us now look at these 7 training examples, 3 attributes (GF, A, BT) and 1 boolean target class attribute (GoToSchool), and see how the information gain measure was used to build the tree shown in Figure 2.4. At the first step, the Root node is constructed, and information gain in equation 2.3 applied to each of the 3 attributes that could possibly be tested at that node. The

```
function ID3(Examples, Target_attribute, Attributes)
 Inputs: Examples - the set of training  examples
         Target_attribute - the attribute whose value is to be predicted by the tree
         Attributes - a list of attributes that may be tested in order to classify an example
 Output: a decision tree rooted at Root, that correctly classifies the given Examples

  (1) Create Root node for tree
  (2) If all Examples have the same classification, return the single-node tree Root with that classification
  (3) If Attributes is empty, the return single-node tree Root, with classification same as
                                  the majority value for target attribute among Examples
  (4) Else
     (a) A  <—  the attribute (with highest information gain) that best classifies Examples
     (b) Decision Attribute Label for Root   <— A
     (c) For each possible value v_i of best attribute A do,
          (i) Add a new tree branch below Root, corresponding to the test A = v_i
          (ii) Examples(v_i)  <—  { subset of Examples with A = v_i }
          (iii) If Examples(v_i) is empty
                then Add a leaf node to this branch with classification
                        = majority value for target attribute among Examples
                else Add the subtree produced by following call, to this branch
                        ID3(Examples(v_i),Target_attribute, Attributes - {A})
```

Figure 2.5: A decision tree learning algorithm

gain for each attribute is obtained as shown below:

From equation 2.1 we can calculate the entropy of the training set and each of its subsets.

$info(T) = -\frac{5}{7} \times log_2 \frac{5}{7} - \frac{2}{7} \times log_2 \frac{2}{7} = 0.859$

$info(T_{GF=Good}) = -\frac{3}{3} \times log_2 \frac{3}{3} - 0 = 0$

$info(T_{GF=Weak}) = -\frac{1}{2} \times log_2 \frac{1}{2} - \frac{1}{2} \times log_2 \frac{1}{2} = 1.0$

$info(T_{GF=Sick}) = -\frac{1}{2} \times log_2 \frac{1}{2} - \frac{1}{2} \times log_2 \frac{1}{2} = 1.0$

From equation 2.2,

$info_{GF}(T) = \frac{3}{7} \times info(T_{GF=Good}) + \frac{2}{7} \times info(T_{GF=Weak}) + \frac{2}{7} \times info(T_{GF=Sick}) = 0.57$

And from equation 2.3, we obtain the gain for attribute GF as:

$gain(GF) = 0.859 - 0.57 = 0.289$

Similarly,

$info(T_{A=Good}) = -\frac{3}{3} \times log_2 \frac{3}{3} - 0 = 0$

$info(T_{A=Poor}) = -\frac{2}{4} \times log_2 \frac{2}{4} - \frac{2}{4} \times log_2 \frac{2}{4} = 1.0$

$info_A(T) = \frac{3}{7} \times info(T_{A=Good}) + \frac{4}{7} \times info(T_{A=Poor}) = 0.57$

$gain(A) = 0.859 - 0.57 = 0.289$

And,

$info(T_{BT=Normal}) = -\frac{4}{5} \times log_2 \frac{4}{5} - \frac{1}{5} \times log_2 \frac{1}{5} = 0.722$

$info(T_{BT=High}) = -\frac{1}{2} \times log_2\frac{1}{2} - \frac{1}{2} \times log_2\frac{1}{2} = 1.0$

$info_{BT}(T) = \frac{5}{7} \times info(T_{BT=Normal}) + \frac{2}{7} \times info(T_{BT=High}) = 0.8$

$gain(BT) = 0.859 - 0.8 = 0.059$

Thus, from the above calculations for gain, we see that the attribute called Body-Temperature (BT) has maximum entropy and hence, lowest gain. The attributes General-Feeling(GF) and Appetite(A) have same amount of information gain – generally, the attribute listed first in the order of attributes is picked in case of a tie[10]. Now, let us proceed to build the rest of the tree with decision attribute being tested at the Root node as General-Feeling. GF has three values and thus three branches are created. The value GF=Good completely classifies an example as being a positive predictor for target attribute *GoToSchool*. Thus, a leaf node is created and labeled YES. Now, consider the branch where GF=Weak. The algorithm again calculates the gain for each of the remaining attributes given only the training examples that have GF=Weak.

$info(T_{GF=Weak}) = -\frac{1}{2} \times log_2\frac{1}{2} - \frac{1}{2} \times log_2\frac{1}{2} = 1.0$

$info(T_{GF=Weak\&A=Good}) = -\frac{1}{1} \times log_2\frac{1}{1} - 0 = 0$

$info(T_{GF=Weak\&A=Poor}) = -0 - \frac{1}{1} \times log_2\frac{1}{1} = 0$

$info_A(T_{GF=Weak}) = 0 + 0 = 0$

$gain(A_{GF=Weak}) = 1.0 - 0.0 = 1.0$

Whereas,

$info(T_{GF=Weak\&BT=Normal}) = -\frac{1}{2} \times log_2\frac{1}{2} - \frac{1}{2} \times log_2\frac{1}{2} = 2.0$

$info(T_{GF=Weak\&BT=High}) = 0$

$info_{BT}(T_{GF=Weak}) = 2.0$

$gain(BT_{GF=Weak}) = 1.0 - 2.0 = -1.0$

Therefore, clearly, the subtree below the branch GF = Weak will contain a test with attribute Appetite (A), leading to the attachment of two leaf nodes with differing class labels(YES and NO), for each of the two values for the attribute A. The same process applies to the subtree along the branch from the root node, with test outcome for General-Feeling as Sick. It is to be noted that the attribute whose values best split the training examples with respect to the target class is always chosen using this measure.

2.5.2 Rule Induction

RL[15] (Rule-Learner) is a knowledge-based inductive rule learning program, that induces one or more **IF**-*condition*-**THEN**-*class* rules from specific examples of classes. The main strength of RL is its flexibility. Given a learning problem, many different problem models and assumptions can be tested. The flexibility is partly achieved through the use of a domain model, called the Partial

[10] Yes! The order in which attributes are listed does make a difference!

Domain Model (PDM), which can guide RL's search separately from the guidance implicit in the statistics of training examples. The PDM contains definitions of attributes to be used in representing examples and rules, a list of classes, assumptions and constraints on rules being sought, and domain knowledge relevant to a particular problem. Constraints and domain knowledge usually take the form of preference criteria characterizing desirable properties of rules to be learned. Thus, induction in RL is guided not only by syntactic similarity and dissimilarity of features of examples, but also by constraints and prior domain knowledge in the PDM. Given a learning problem, *i.e.*, the names of one or more target classes, a set of their examples, and a PDM of the problem, RL searches for rules by examining a large but limited number of combinations of features (RL performs a beam search wherein the beam width is user-definable and dictates the extent of the search, viz.coarse or detailed. The rules on the beam are sorted according to rule coverages and confidence measure of rule). RL uses a "generate-then-test" paradigm for searching the hypothesis space. The RL algorithm thus performs a top-down specialization to learn the maximally general set of rules that best classify the training examples. The beam search algorithm and specialization techniques employed by RL is summarized in Figure 2.6. The plausibility of a rule is determined by its performance (how accurately it classifies examples) and its concordance with assumptions, constraints, and domain knowledge. The result of rule search is a disjunction of **IF**-*condition*-**THEN**-*class* rules, where condition is a conjunction ("AND") of features. Each rule has associated with it a statistic which indicates the likelihood of accuracy of the prediction. This certainty measure is calculated based on the positive and negative coverages of each individual rule (percentage of positive examples/negative examples covered by the rule with respect to the total number of positive/negative examples present in the training examples). Such **IF-THEN** rules are very easy to understand and comprehensibility facilitates the easy verification of rules by experts.

For example, in some earlier studies [35, 31], the data from the Biological Macromolecular Crystallization Database (BMCD) were partitioned based on the diffraction limit of a crystal - e.g., difflim < 3 and difflim > 3Å. RL was given a set of BMCD data that is partitioned based on the diffraction limit, and it induced one or more rules that described correlations between experimental parameter values and the diffraction of crystals obtained with those conditions. RL uses a heuristic rule search which can utilize prior domain knowledge, such as facts or heuristics used by scientists, and can examine a much larger number of identification criteria than can be examined with manual analysis.

For RL, an example is represented as a vector of attribute-value pairs, each of which describes a feature of the example. For example, the representation of a feature subset of an entry or experiment in the BMCD is shown in Table 2.2.

RL: Top-down specialization algorithm

(1) Create initial rules: one for each class

LHS of rule = NIL

RHS of rule = class

(2) Repeat while there exist too-general rules on the beam

and stopping conditions are not met

(stopping conditions: redundant search, maximum number of conjuncts in a rule)

Consider only the best n too-general rules where n=beam-width

For each too-general rule that covers an example not previously covered,

(a) perform all one-step specializations on that rule

(b) compute positive and negative coverages of rule

(c) if rule is too-specific (covers very few positive examples),

then prune entire subtree rooted at that rule

else if rule is too-general (covers too many negative examples),

then add the rule to next generation of too-general rules

else save the rule as a good-rule

if learning maximally-specific rules,

then add rule to next generation of too-general rules

(3) Keep maximally general/specific of good-rules

Function for specializing a rule (one-step specializations)

if adding a conjunct to the rule will exceed the Maximum Conjuncts limit specified by user,

then do not specialize the rule

else formulate new rules by

(1) adding one new attribute to LHS (new conjunct) and attaching the most general value in its value hierarchy

(2) specializing an existing attribute by replacing the conjunct in the rule with a more specific value for the attribute from its value hierarchy

Figure 2.6: RL beam search algorithm

Representation	((macmol "Alcohol-dehydrogenase") (maccon 5) (crmethod Bulk-Dialysis) (srctis liver) (srcgsp equus-caballus) (pH 8.4) (temp 4.0) (buffer Tris) (spacgp C222-1) (difflim 2.4))
Meaning	An entry in the BMCD that contains the successful crystallization conditions of the macromolecule Alcohol-dehydrogenase obtained from the liver of a horse, used the bulk-dialysis method of crystallization. The favorable experimental conditions were a macromolecular concentration of 5 mg/ml, pH of 8.4, the buffer Tris, and temperature of 4^oC. The crystal that resulted from this experiment diffracted well with a difflim of 2.4Å, and belonged to the space-group C222-1.

Table 2.2: Attribute-value pair representation of a subset of features in an experiment in the BMCD database.

The result of rule search is a disjunction of **IF**-*condition*-**THEN**-*class* rules, where condition is a conjunction ("AND") of features. For example, the following rule uses two features to predict crystal diffraction quality:

IF-(buffer cacodylate) and (molwgt > 39630)-**THEN**-(DIFLIM-UNDER -3)

which is interpreted as *if the buffer cacodylate is used to crystallize macromolecules that have a molecular weight over 39,630 kilo Daltons; then you can obtain well-diffracting crystals.* Each rule has associated with it a statistically computed number, which indicates the likelihood of accuracy of the prediction. Equation 2.4 provides the calculation for certainty factor or weight that is associated with each rule in the beam.

$$cf = \frac{p}{p+n} \tag{2.4}$$

where cf = frequency-based certainty factor, p = number of positive examples covered by the rule or the number of true positive predictions over the training examples, n = number of negative examples covered by the rule or the number of false positive predictions.

Some additional features of RL (which can also be found separately in other methods) are: It can

- exploit redundancy in the attributes (rather than insisting that they be independent).
- integrate numeric and symbolic attributes. Semantic relationships among symbolic attributes can be exploited. Meaningful intervals of continuous numeric attributes are learned automatically.
- tolerate erroneous and missing data. The training data may be presented to the program incrementally, that is they do not all need to be in the database at once.

RL beam search - set of rules on the beam

(Shows portion of the hypothesis space)

{NIL => YES, NIL=>NO}

↓

{(GF=Good)=>YES, (GF=Weak)=>YES, (GF=Sick)=>YES, (GF=Good)=>NO,(GF=Weak)=>NO, (GF=Sick)=>NO}

tp=3, tn=0, cf=1.0 tp=1,tn=1,cf=0.5 tp=1,tn=1,cf=0.5 tp=0,tn=0,cf=0 tp=1,tn=1,cf=0.5 tp=1,tn=1,cf=0.5

↓

{(Appetite=Good)=> YES, (Appetite=Poor)=>YES, (GF=Weak) & (Appetite=Good)=>YES , ...}

tp=3,tn=0,cf=1.0 tp=2,tn=tn=2,cf=0.5 tp=1, tn=0, cf=1.0.....

↓

Finally, RL reports that it found 5 good rules, whose disjunction covers all the training examples:

(GF=Good)=> YES, (Appetite=Good)=>YES, (GF=Sick)&(BT=Normal)=>YES,
(GF=Weak)&(Appetite=Poor)=>NO,(GF=Sick)&(BT=High)=>NO

Figure 2.7: Beam search by Rule-Learner(RL) for concept *GoToSchool*

- use prior knowledge about the relative costs of false positive and false negative predictions to learn more appropriate rules. RL learns rules that make their predictions with different degrees of strength.

- attribute and value hierarchies. Hierarchies of attributes and their values can be represented and used during specialization.

RL is a descendant of the Meta-DENDRAL system [22], which specialized in finding rules of mass spectrometry in chemistry. However, unlike the Meta-DENDRAL, RL is a general purpose learning program that can be applied to many problems in different domains. RL has been applied successfully in several real world problems, including learning predictive rules for carcinogenic activity in chemicals [2], learning filtering criteria for high energy physics experiments [53], and learning diagnostic rules for telephone line problems [54].

Figure 2.7 shows the manner in which RL searches the hypothesis space for the GoToSchool concept. The outcome of the learning program is also shown. The rule $Appetite{=}Good \rightarrow YES$ is more general than the rule that has 2 conjuncts, viz. $GF{=}Weak$ & $Appetite{=}Good \rightarrow YES$ and covers the example covered by the more special rule. Therefore, only the more general rule has been retained. Please note that when rules are actually built from the decision tree obtained from running the program C4.5 on the set of training examples, the rules are later generalized by removing conjuncts as necessary and evaluating the performance of the smaller rule within the rule-set. Every path from the root of a decision tree to the leaf gives one initial rule. The left-hand side of the rule contains all the conditions established by the path, and the right-hand side specifies the class of the leaf. Each such rule is simplified by removing conditions that do not seem helpful in discriminating the nominated class from other classes, using a pessimistic estimate of the accuracy of the rule. For each class in turn, all the simplified rules for that class are sifted to remove rules that do not contribute to the accuracy of the set of rules as a whole. Thus, the outcome of the two symbolic induction programs are typically similar and hence, comparable.

2.5.3 Are these equivalent?

RL is more flexible of the two induction programs, due to its design. It can be run in both a batch mode, as well as an incremental mode, wherein prior rules learned can be used to prune the search involved, by eliminating the need to cover training examples already covered by the prior rules. RL is also able to utilize value-hierarchies that reflect the partial domain model (PDM) during specialization of rules. This has been found to be very useful in converging to a good rule-set that classifies training examples with less computational time.

The C4.5 program typically executes faster because there are commercially available versions of the software program. We therefore use C4.5 for a majority of the experiments performed with the PEP prototype system.

2.6 Active Experimentation using Machine Learning

Buchanan[11] provides a philosophical discussion of the role of experimentation in artificial intelligence in which he identifies two important dimensions of the phenomena studied by empirical science. These relate to manipulation and intervention, where manipulation refers to the nature of production of the phenomena themselves, such as those that are naturally occurring vs. those that are reproducible in artifacts; and intervention refers to the manner in which the phenomena are studied, such as by passive observation (in nature or in the laboratory) or by active experimentation (usually in the laboratory). The discussion in this paper [11] pertains to illustrating the view-point that AI is an empirical science and by building a computational device that embodies a plausible mechanism designed to solve a problem that requires some intelligence, we can run this device under many conditions, and using such an experimental apparatus, we can try to understand the principles that govern intelligent action.

The use of machine learning programs for learning the definition of boundaries between target classes has presented an interesting research problem, namely that of how to select the training instances that directly influence the learning achieved by the program. Buchanan et. al. [10] discuss the problem of intelligent selection of examples for learning systems, and show via some experiments performed in the context of the rule-learning system RL described above, that intelligent selection is better than random selection of training examples. They also conclude that incremental learning will converge with fewer examples than batch learning, since the conjectures in the evolving theory about where the boundaries lie between classes can enable effective choice of the next set of training examples.

The PEP framework includes a generalization program that essentially provides a representation of the evolving theory of the boundary between the different classes of observable results. The false positive and false negative predictions of test or training examples based on this evolving theory constitute the "holes" in the theory and point us to the areas in the search space that would require further experimentation.

Chapter 3

Annotated Example

In this chapter, I present an example of the types of inputs to and the outputs obtained from the program implemented to demonstrate sufficiency of the PEP framework. The model used for simulation is representative of a hypothetical protein that crystallizes rapidly under a great many initial conditions (e.g. lysozyme). The program is meant to be interactive and is based on the framework shown in Figure 4.1 in the following chapter.

The various constants or givens for a particular protein including its pI and phase information is saved in a file labeled "givens". Table 3.1 shows an annotated example of the givens used in this example. Definitions of the given and control parameters are provided in Appendix B.

Parameter	Value
Protein Name	Alpha Globulase
Protein pI (Isoelectric Point)	7.7
Number of phases	2
Phase I Name	Amorphous
Nucleation saturation for Phase I	1.3
Growth saturation for Phase I	1.1
Maximum rate of growth (Phase I)	0.06
Number of heterogeneous nucleii (Phase I)	3
B-factor (Phase I)	25.0
Phase II Name	Crystalline
Nucleation saturation for Phase II	2.0
Growth saturation for Phase II	1.6
Maximum rate of growth (Phase II)	0.05
Number of heterogeneous nucleii (Phase II)	2
B-factor (Phase II)	12.0
Number of salts	1

Table 3.1: Example of *givens* for a hypothetical protein

Using the information in the givens, and the controls generated[1] for each experiment as shown in the following example, we can run and observe the partial results for each experiment over different times. Table 3.2 shows the set of inputs that comprise controllable parameter values for experiments in this example.

	Value
Globals	
Protein Givens	Table 3.1
Initial amount of Protein	100 mg
Drudge Tolerance Factor	1.0 (high tolerance for pain)
For each new trial	
Number of experiments	8
Type of search	Grid - factorial search
Number of salts to vary	2
Names of the salts	Sodium Chloride, Ammonium Sulfate
Number of salt concentrations to vary	2
Amounts of the concentrations	0.1 , 1.0 M
Number of precipitants to vary	1
Names of the precipitant	Mama PEG
Number of precipitant concentrations to vary	1
Amounts of the concentration	20.0 %w/v
Number of pH values to vary	1
Value of pH	8.0
Number of protein concentrations to vary	2
Amounts of the concentration	1.0, 5.0 mg/ml
Time until next observation	1 day (0.14 weeks)

Table 3.2: Annotated example of inputs to the framework

The program generates the factorial combination of experiments based on the control variable values as specified above. Please note that in the example, only the salts and their concentrations are varied together with protein concentration. The precipitant is fixed as Mama PEG at a single concentration and pH value. Thus, in the initial trial, the experiments are performed at low and high salt concentrations of two commonly used salts, Sodium Chloride and Ammonium Sulfate. The program asks the user to specify the time at which she will observe the partial results and then generates summaries of results. The user is presented with summaries of all partial results as well results of individual experiments and trials. The user is then asked to make a decision as to whether

[1] The control parameters for each experiment are generated from a set of discrete values for each control variable specified by the user. The user also indicates the type of search or manner in which the values are distributed across the number of experiments performed in parallel. In the prototype PEP system, we use factorial or grid search, that is all possible combinations of values of the specified control variables are generated, and each combination represents a set of the control parameters for an experiment.

Statistic Name	Statistic Value
Experiment Number	2
Experiment Controls	Sodium Chloride, 0.1, Mama PEG, 20.0, 8.0, 5.0
Trial start time Current time	0.00 0.14
Amount of protein consumed Direct material costs of controls	0.05 mg 1.012 units
Experiment Observables Time of Observation Difflim Result Class	 0.14 weeks (1 day) 1.67 Å 1
Best Results statistics Best difflim Time of Observation	 1.67 Å 0.14 weeks (1 day)
Experiment utility	3.0

Table 3.3: Example of an experiment summary

she would like to close any of the trials or experiments for observation. The idea of this facility is to be able to calculate the savings in cost in terms of time and effort involved. The user can reopen any closed trial or experiment for observation again if necessary. The user can also decide not to begin a new trial but instead to just observe the results of past trials. Each experiment is observed after a user-specified time (e.g., a day) and the results are reported, along with an indication of how much protein was consumed by the experiment. Total cost is specified as a summation of three types of costs - direct material costs, intangible costs (based on how tiring it is for a particular agent to set up a large number of experiments in the lab) and observation costs (i.e., cost for observing the results of experiments over time). Set up costs include direct material costs and intangible costs. The set up costs for a trial indicate the costs for setting up or performing a certain number of experiments in parallel in the laboratory. Each time an experiment is observed, there is a cost associated with it. Thus, the total costs incurred at any time for a crystallization project is the sum of set up costs for each trial and costs for observing partial results for each open experiment. Utility of each experiment is also calculated as a function of type and quality of partial result and time taken to obtain it (see section 4.3.3). Table 3.3 presents a sample experiment summary.

After each experiment in a trial is observed at the time of observation, a trial summary is generated - a sample is shown in Table 3.4

Statistic Name	**Statistic Value**
Trial number	1
Start Time	0.00
Number of experiments	8
Amount of protein consumed	0.24 mg
Percentage of initial protein consumed	0.24%
Direct material costs	8.10 units
Intangible costs	0.8 units
Set up costs	8.90 units (Direct + Intangible Costs)
Percentages of observables in each class	
Class 1	50.00%
Class 2	0.00%
Class 3	50.00%
Experiments with class 1 results	2, 4, 6, 8
Experiment 2 Controls	Sodium Chloride, 0.1, Mama PEG, 20.0, 8.0, 5.0
Experiment 4 Controls	Sodium Chloride, 1.0, Mama PEG, 20.0, 8.0, 5.0
Experiment 6 Controls	Ammonium Sulfate, 0.1, Mama PEG, 20.0, 8.0, 5.0
Experiment 8 Controls	Ammonium Sulfate, 1.0, Mama PEG, 20.0, 8.0, 5.0

Table 3.4: Example of a trial summary

The trial summary depicts a list of the experiments showing the mixtures of control variable values that result in hits or good quality crystals (labeled as class 1 results). If this list was short, we could eyeball the summary and make appropriate inferences about the most influential variables for crystal growth. In the example shown, we can see that the salts and their concentration do not make much impact, that is either salt could be used at any of the two concentrations. Protein concentration, however, does make a difference. We will see below how the generalization program can help identify and represent this boundary between observed classes of results in a compact, human-understandable form. The need for such a generalization program arises because several experiments (hundreds or more) typically run concurrently, and it is very difficult to summarize and define the evolving boundary that relates control values to observed effects without intelligent computational assistance.

As we note above, in addition to the results of each individual trial, this entire process is ongoing and concurrent, with new trials that can be started at any time after observing a set of partial results. As we evaluate the global summary, we can begin to see the patterns that distinguish conditions that yield good crystals from other classes of results. Here, we have grouped the results as follows: Class 1 refers to crystals of good quality, Class 2 to precipitates or crystals of poor quality and Class 3 to no change or clear result. A sample global summary is shown in Table 3.5.

Statistic Name	**Statistic Value**
Observation Time	0.14 weeks (1 day)
Number of trials so far	1
Number of experiments so far	8
Number of closed trials	0
Number of additional experiments closed	0
Number of hits (good crystals)	4
Total direct costs	8.10 units
Total intangible costs	0.8 units
Observation costs	1.36 units
Total costs	10.26 units
Material consumed	0.24 mg

Table 3.5: Example of a global information summary

The program then states the following and prompts the user to make the following decisions.

```
Results of decision tree analysis is in file SEEME.
Would you like to close any trials at this time?n
Would you like to close any other experiments at this time?n
Would you like to open any closed trials at this time?n
Would you like to open any closed experiments at this time?n
Do you want to start a new trial (y/n): n
Do you want to perform an observation?: n
```

The above decisions reflect those that need to be taken by the experiment designer while using this framework to try different strategies for arriving at suitable conditions for crystallizing a new protein. The ability to close experiments and trials for observation is useful, since this results in a reduction in the cost of observation of experimental results that change over time. Since we might be able to ascertain via partial result examination and evaluation that it would not be fruitful to pursue certain experiments, this mechanism is necessary to include within the PEP prototype.

The framework is open-ended on purpose, so that more tools for analysis can be added as necessary. For instance, these could be tools that could help in the visualization of this multi-dimensional space, with a clear marking of the boundary between the classes. Some preliminary work has already been done in Hennessy, et al. [58]. In this dissertation, we focus on providing a sufficient set of statistics, data structures from which several important statistics can be computed, and tools such as the decision-tree analysis program C4.5 [57] that can provide a representation of important features that distinguish the broad classes of observed partial results. Such a generalization program can learn the features that discriminate classes of examples given a set of training examples represented as feature or (attribute,value) pair vectors. This is useful in presenting to the experiment designer an idea of which control variables are most effective in distinguishing the results along the various classes that can be seen in the set of partial results obtained from experimentation. The C4.5 machine learning program provides one representation of the evolving boundary in a multi-dimensional space. Surely, this is not the only representation, but for the purposes of experiment design, this could be quite useful, as we will see in Chapter 6.

The input to the generalization program is a set of examples. Each example represents the values of experimental control variables together with the time of observation of partial result and the outcome or observed class of result at that time. Thus, the number of training examples presented to C4.5 is equivalent to the total number of observations performed up to the time at which the analysis is performed. The data accumulates as more observations are performed (either on the same set of open experiments, or another set which includes some new experiments as well).

Below is a summary of the results from the decision-tree program in file SEEME.

```
Rule 2:
        protconc > 1 ->  class 1 (CRYSTAL)


Rule 1:
        protconc <= 1 ->  class 3 (CLEAR)
```

The rules indicate that protein concentration was the most influential variable in providing the distinction between classes 3 (clear) and 1 (good crystal). Thus, using these rules, the experiment designer can carry out future trials, wherein protein concentrations less than or equal to 1 mg/ml are not used.

We can see even from this simple example, how using this inductive analysis, the designer can infer the most influential parameters and their values that are responsible for marking the boundary between the visible classes. Interestingly, this analysis can be used to identify the focus of future experimentation, by identifying the examples that are being classified incorrectly. Buchanan et al. [10] have performed initial studies that show how intelligent choice of training examples affect learning.

The final summary of the path is shown in Table 3.6. The final summary is used to indicate to the designer the time at which the first hit was observed, and the number of experiments performed by then, as well as the protein consumed by the experiments. This summary can help the designer understand the value of the strategy from the perspective of time and cost to obtain a single hit. The final summary is additional feedback given to the user of the PEP system. The main goal for the user is to obtain a hit given some protein to be crystallized. The additional feedback is

Statistic Name	**Statistic Value**
Initial Protein Amount	100.0 mg
Drudge Tolerance Level	1.0 (High Tolerance)
Time first hit observed	0.14 weeks (1 day)
Protein consumed	0.24 mg
Number of experiments	8
Total costs	10.26 units

Table 3.6: A final summary of time and cost to obtain a single hit (crystal)

simply a way to quantify the costs involved by employing different strategies for searching the space of possible experiments.

Chapter 4

Implementation of the PEP framework

The framework for designing parallel experiments as concurrent trials is shown in Figures 4.1 and 4.2. The framework has been developed to facilitate the setting up of parallel experiments as a defined region of the vast search space that is being searched at some level of granularity. Parallel experiment planning is therefore viewed as heuristic search with the human experiment designer as the heuristic control element. The control exercised by the human user is both appropriate and necessary in this domain.

4.1 Key Ideas

The main ideas that are represented in the framework in Figure 4.1 can be stated as follows:

1. Economic variables are important in strategic decision making, and hence must be included as part of every data gathering and data analysis project.

2. The nature of the real-world is such that multiple actions are taken in parallel, and the effects of some critical actions/decisions are visible only in the long-term, the evaluation is not precise as it is often based on visual information, and resources such as time and money are constrained. So, we need a model that can be built on such assumptions, and can enable us to operate effectively under such conditions. Interestingly enough, the domain of experiment design in crystallization actually meets all the afore-mentioned real-world challenges, along with some possibly unique to its own.

3. In order for parallel experiments to be designed, it is necessary to design variables/data structures that can manage the information in an aggregate manner. Figure 4.3 depicts an overview of the PEP information management system that keeps track of experiments performed, resources consumed and costs incurred. The system maintains information that pertain to experiment, trial and global information summaries. Experiment summaries include information about the best partial result observed so far for each experiment, and the time of observation

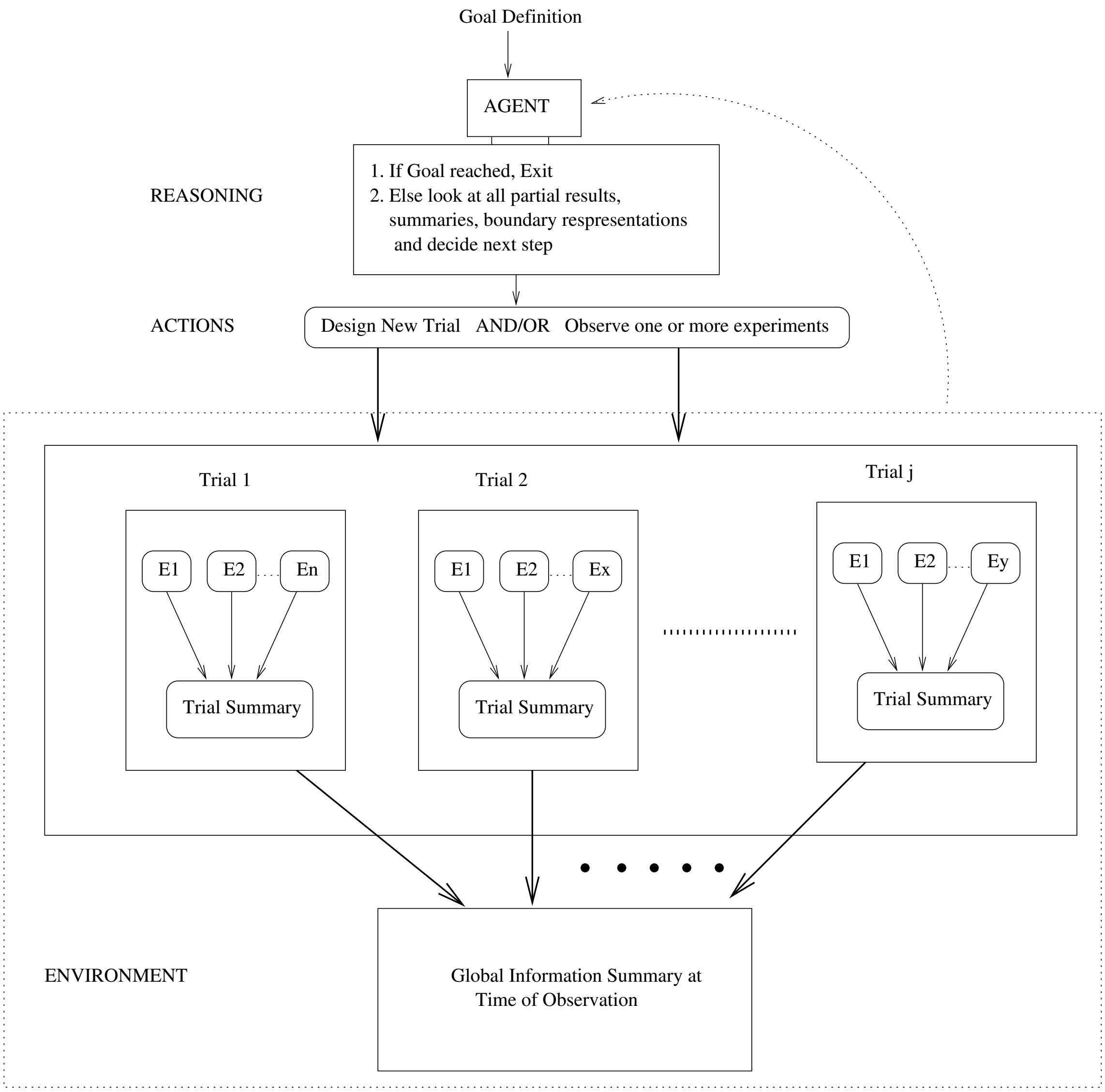


Figure 4.1: Framework for designing parallel experiments as concurrent trials

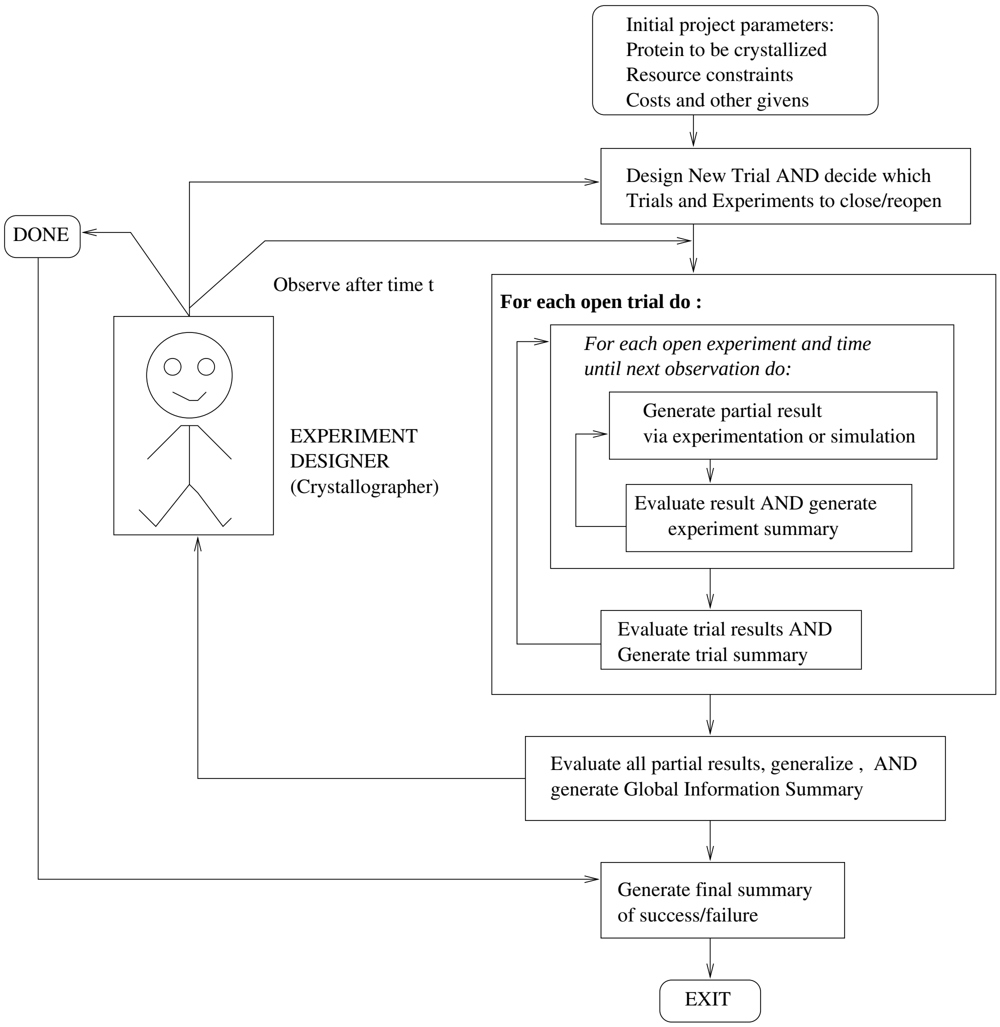


Figure 4.2: Details of one trial shown within the PEP system implementation

of the best result. This record is important as it is indicative of the rates of change of control parameters for that experiment, as well as its chances of yielding a good quality crystal. Trial summaries include percentages of observable results that belong to each class (clear, crystal or precipitate). This is helpful in identification and classification of the sub-area of search defined by the set of experiments that comprise the trial as promising or not. Global information summaries are useful in tracking the state of the search over time.

4. In order to provide intelligent assistance for decision-making, it is necessary to implement first the PEP information management system that can maintain the information about experiments in electronic form. Intelligent assistance can then be provided by building partial or approximate models, and using them to simulate experimental data for input to a generalization program. The generalization program can help identify the boundary characteristics that separate the successful and unsuccessful experiments given the model. We could thus simplify a complex problem with a large parameter space for search, by building a model to predict the results for a reduced set of dimensions, whose interactions are probably well-understood in theory. The PEP framework, which includes the simulator model for hypothetical macromolecules (see Chapter 5) and a generalization program (such as C4.5 or RL), comprises the intelligent assistance for decision-making in this domain. By using simulation, we can reduce the tedious nature of experimentation in the laboratory, and conserve resources.

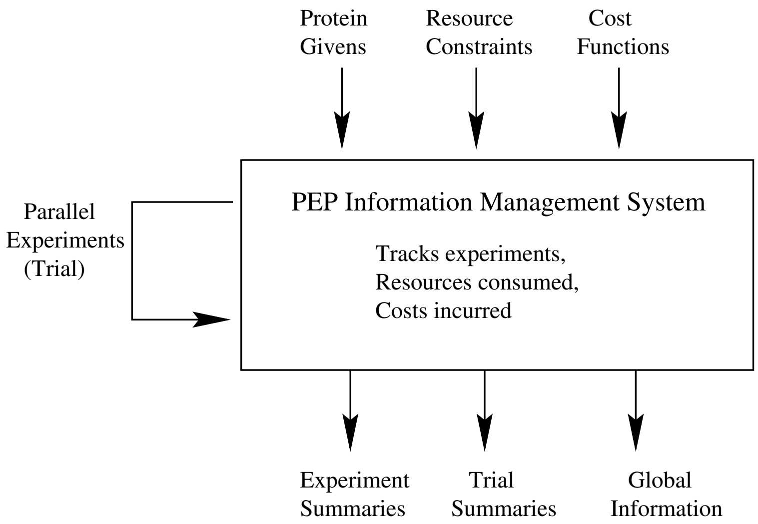


Figure 4.3: PEP information management system overview

Figure 4.4 depicts an overview of the search involved in the PEP framework. An initial population of experiments represents the first trial. These experiments are observed and evaluated, and based on evaluation of these partial results, a decision-making agent applies operators to generate a new trial and close (or reopen) one or more experiments for observation. The initial set of experiments are still running concurrently. The simulator model developed in Chapter 5 returns an observation for any experiment after a specified period of time (a real number specified in weeks). The generalization program represents intelligent evaluation of experiments and their results over time, and provides a representation for boundary characteristics between observed classes of results in terms of rules containing a conjunction of control values on the left-hand side and the observable class (such as crystal) on the right-hand side.

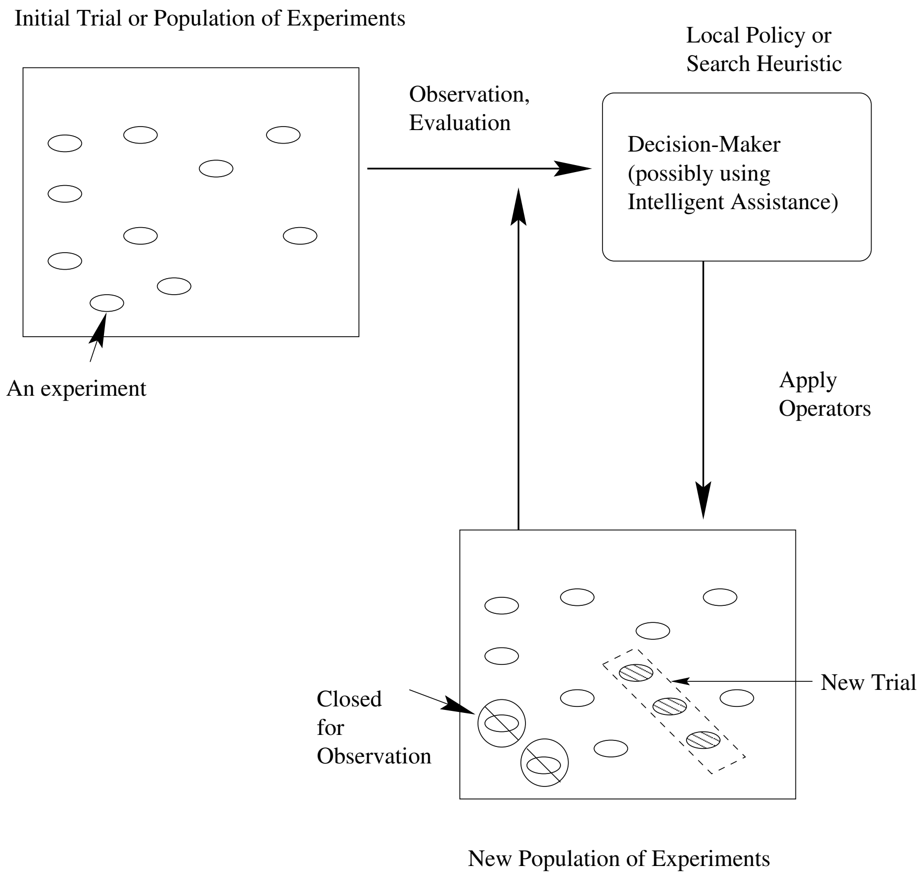


Figure 4.4: The PEP search

4.2 Representations and Data Structures

4.2.1 Data Structures

The variables that define a project can be classified as follows:

1. Givens: These are parameters whose values remain unchanged for an entire project. Examples are macromolecule name, its isoelectric point or pI, the various solid phases that could potentially emerge, the saturation levels required for nucleation and growth of these different phases, and the solubility tables for the protein with respect to salt, precipitant and pH. Also

unchanged are the cost functions associated with obtaining and purifying the protein and effort involved in setting up experiments.

2. Controllables: These are the parameters that are changed from one experiment to another in the course of a project. For the simulator, these include macromolecular concentration, salt, salt concentration, precipitant, precipitant concentration and pH.

3. Observables: These are the outcome variables associated with the goal of growing good crystals. They are stored as vectors containing information about the result type and quality over time. Class 1 results refer to crystals of good quality, with the resolution limit of diffraction less than or equal to 3 Å. Class 2 results refer to precipitates or crystals of bad quality, with the final outcome number between 3 and 100. Class 3 results refer to no change in solution or a clear observation, with final outcome number from simulation equal to 1000.

4. Experiment summary: The summary of each single experiment refers to the controllables used in the experiment, and the vector of observables over time. Also computed and recorded as part of the summary is the best result obtained for that experiment (i.e., best difflim quality) and the time at which that result was obtained. This is important information for a crystallographer, as it is often noticed that a very good quality crystal would appear at a particular time, and if it was not taken out for X-ray diffraction analysis, would eventually deteriorate in quality as time passes.

5. Trial summary: The summary of a single trial is a record of a parallel action and its observable effects over time. The parallel action refers to the act of designing a new set of experiments in parallel or observing some number of experimental results in parallel. Design of a new trial includes defining a discrete set of values for each control parameter, and an algorithm for assigning values for each set of control variables for a specified number of experiments. For example, such an algorithm could be based on a factorial allocation of control parameter values, where the total number of experiments performed in parallel would be equal to the factorial combination of the numbers of discrete values defined for each control variable. The number of different values chosen for each single control parameter reflects the granularity of the search. The observable effects of a trial are summarized by storing percentages of observables belonging to each of a set of classes, based on the observable quality of the result of each experiment in the trial.

6. Global Information Summary: The overall global information is the set of global variables that provide a summary of the state based on the total number of experiments and trials performed, costs incurred and resources consumed. This summary also includes the number of hits or good crystals obtained. Thus, by looking at the information provided in the global information

summary, the user can decide whether to stop, proceed with a new trial or perform more observations. The global information summary also includes the results of a generalization program (an information-gain based decision tree analysis program[57] called C4.5) that can provide intelligent assistance by identifying and presenting in human-understandable form the characteristics of the boundary between observable classes of partial results. The decision-tree analysis also yields useful insights as to the most influential parameters in an experiment with respect to the observable class. Such intelligent analysis can help the user in the the choice of parallel actions, such as deciding which experiments to introduce in a new trial, and which experiments to close for observation from older trials.

4.2.2 Representing Cost

Cost has been modeled as a combination of direct costs for material and time, and intangibles, such as the tiring nature of the work of setting up experiments in crystallization. To model intangibles, I have used a factor called the *drudge tolerance factor* as an input that affects the cost of intangibles. This parameter γ is a number that ranges from 0.1 to 1.0, where 0.1 represents an experimenter with low tolerance to drudge work and 1.0 represents a robot that has very high tolerance levels. I model the effect of this parameter as follows:
Let us assume that there is some constant intangible cost $c_{i,1}$ associated with the setting up of one experiment[1] and that the tedium of experiment setup rises exponentially with increasing numbers of experiments. We can express the intangible costs arising from setting up N number of experiments as:

$$c_{i,N} = c_{i,1} \times N^{1/\gamma} \tag{4.1}$$

This equation represents one way in which we can express the intangible costs that occur with long hours of strain associated with setting up experiments in the laboratory. We could easily incorporate variations to this cost calculation, by assuming that the γ factor does not come into effect until a certain threshold value of N is reached.

Direct costs for materials have been modeled as scaled actuals, based on the consumer booklets from the chemical reagents suppliers [17, 16].

4.3 Descriptions of Main Parts

The main components of the framework that supports parallel experiment planning in this domain are discussed below.

[1] We could always change this assumption by modeling a variable intangible cost depending on the particular composing factors of an experiment.

4.3.1 Design Trial

This is the first step or task that needs to be undertaken by an agent. Initially, only resource constraints (and maybe time constraints) are specified. The agent or experiment designer has knowledge of all the givens (including costs of reagents) for the protein that needs to be crystallized. The task of the designer is then to decide how many experiments to perform in parallel, and what the actual values of the controllable parameters are for each experiment.

If this same task needs to be repeated in light of partial outcomes that have been observed and evaluated, the type of decision-making involved changes slightly. The designer now needs to make a choice among several possible actions, basically involving whether to begin a new trial or not. These include:

1. Do not start a new trial, but observe *some* or *all* existing experiments after a certain period of time, when new evaluation can take place.

2. Close *none*, *some* or *all* of the existing experiments for observation and start a new trial, which requires further decision-making as to how many experiments to setup and what controllable parameter values to use for each.

The decision-making process is thus complex and involves tradeoffs along several dimensions, two of them being cost and time. If the decision made is to design a new trial, then the user must enter information regarding the number of experiments to perform in parallel, the number of values for each control parameter that she would like to vary across the experiments, and specify the values. In this prototype implementation of the PEP framework, we include only one type of search or manner in which the specified sets of values for each control parameter are spread over the set of parallel experiments. This is essentially based on a factorial grid where each possible combination of values that could be generated from the specified sets of values for control parameters is used to represent a single experiment. Thus, the grid or factorial search generates a set of parallel experiments that comprise a single trial.

4.3.1.1 Grid Search

There are several types of initial search methods that could be included as part of the framework in the future. These represent different types of initial screen designs from prior research. A well known incomplete factorial approach of Carter and Carter[13] assumed that all points in the parameter space of crystallization conditions are equally probable, which has been proven to be an incorrect assumption[34]. Jancarik and Kim [36] employed a semi-automated sparse matrix sampling of published crystallization conditions that has led to the design of commercially available crystallization kits that contain several initial experiments that have proven to be fairly successful.

```
Grid_search()
Inputs: #experiments, #variables, #values for each variable and value names
Output: experiment array with factorial combination of variable values

expt_counter = 0;   var_counter = index of last variable in list;
while(expt_counter < numexpts)
        for each value k of variable[var_counter]
              expt[expt_counter][var_counter] = k;  increment expt_counter;
        step-size = number of values of variable[var_counter];
         while(variable_counter > 0)
                 decrement variable_counter;
                  fill_position(var_counter, step_size);
                  step_size = step_size  x  number of values of variable[var_counter]


 fill_position: fills the appropriate value for the variable into array of experiments

 Inputs: index j of variable, step-size of increment

 expt_index = 0;
 while (expt_index < numexpts)
           for each value k of variable j
                   for #times = step-size
                         expt[expt_index][j] = k;
                          if (expt_index < numexpts) then increment expt_index;
                          else break;
```

Figure 4.5: Grid screen design: factorial search pseudo-code

Grid screens [67] are factorial designs. For the purposes of demonstrating the main claims of this dissertation, I have implemented a simple factorial search, which is equivalent to the grid screen designs. This is one popular method for systematic screening and optimization of crystallization conditions. The procedure is shown in Figure 4.5.

4.3.2 Generate Partial Result

Generation of partial results refers to the task of observing and reporting (electronically) the partial result observed for each experiment. An assumption is made herein that since the process is slow, the exact time at which an experiment is observed could be slightly different than the time noted, since a whole bunch of experiments are typically observed sequentially by a single agent. (If there were multiple agents, then true parallel effects can be achieved.) If we want to get very sophisticated in terms of modeling time, then we could increment current time after each observation, and after each setup. For our purposes, it is sufficient to assume that the same time-lag for setting up each subsequent experiment, balances the time difference in observation of the experiments in a sequential manner. So, we can assume that each trial starts at a particular time, and all experiments within that trial start at that same time, that is simultaneously.

4.3.2.1 In the Laboratory

Typically, experiments are checked on a daily basis, though sometimes, they can be left unobserved for a week or so, depending on what is known a priori about both the protein being crystallized, as well as the experimental conditions themselves. The types of partial results that are observed can be categorized broadly into one of three types - clear (i.e., no change), crystal, or precipitate. Precipitates and crystals have further categorizations.

4.3.2.2 Via Simulation

We have chosen the parameters of the simulation model such as to produce quantitative output describing partial result quality that resembles the actual resolution limits of diffraction used for describing crystal quality in the laboratory. Precipitates are described as large numbers. Good crystals have difflims less than 3Å.

This model is a major component for providing intelligent assistance and hence has been described in detail in the following chapter. The flowchart containing the description of the model developed for this purpose is shown in Figures 5.8 - 5.11. In this dissertation, the term simulator or experiment simulator is used to refer to the implementation of the model developed and described in Chapter 5. The model contains definitions and descriptions that describe the overall process of protein crystallization and by varying the control parameters that are input to the simulator, we can obtain partial results of any experiment (within the search space of definable experiments for the simulator) at any given time of observation.

4.3.3 Evaluate Partial Result and Generate Experiment Summary

The partial result of an experiment is usually observed visually, under a microscope. The result is a description of the observable(s), i.e., whether a crystal or precipitate is seen, and whether it looks like a needle, plate, or is amorphous, and so on. For the purposes of our initial evaluation, we classify the simulated partial result broadly into three classes: clear, precipitate and crystal. It should be pointed out at this time, that the model that describes protein response behavior in different physico-chemical settings is flexible and sophisticated enough to be able to describe and produce different types of solid phases of a protein, each with varying description in terms of shape and size. For purposes of our analysis, we will use just the simple classification, and produce simulations modeling mainly two types of solid phases, i.e., crystalline and amorphous. The amorphous solid phase is labeled as *precipitate*. The crystalline solid phase is labeled as *crystal*. The liquid phase is labeled as *clear*.

1. Update the number of observations performed for experiment E
2. If first observation, compute direct costs (trial T, experiment E)
3. Generate summary of observables at current time for E:
   Time _of_Observation ← current_time - start_time(T)
   Diffraction Limit ← Partial Result
   Result Class ← Assign Class
4. Update material consumed (protein mainly)
5. Update observation costs
6. Update best result statistics (best diflim, time obtained)
7. Calculate experiment utility

Figure 4.6: Evaluate result and generate experiment summary

Name	Description
Observation Number	The number of times partial result seen so far
Controllables	Pointer to description of control parameter values
Observables	Pointer to array of observable data indexed by observation number
Resources Consumed	Pointer to data structure describing resource quantities
Summary	Pointer to summary describing time, best observation, cost, utility of experiment

Table 4.1: Data structures for the items in the experiment summary

The procedure used to evaluate the result and generate a summary of an experiment over time is shown in Figure 4.6. Table 4.1 depicts the data structures that comprise a summary of an experiment.

The utility of an experiment is calculated as follows:
If the experiment is being observed for the first time, we simply assign a number to utility that is one greater than the difference between the largest result class (i.e. 3) and the observed result class. Otherwise, we add to the current utility of the experiment, a quantity proportional to the difference between the result class of current and previous observation and subtract a quantity proportional to current time. The last quantity is a penalty for delay in obtaining a good quality result.

Thus, the first time experiment e is observed, the following equation is applied:

$$U_{e,t} = 4.0 - R_{e,t} \tag{4.2}$$

where $U_{e,t}$ is the utility of experiment e at time t of observation, and $R_{e,t}$ is the result class of experiment e at time t. During subsequent observations, the following equation is used to calculate utility of experiment e:

$$U_{e,t} = U_{e,t-1} + R_{e,t-1} - R_{e,t} - (k \times t) \tag{4.3}$$

where k is a small constant such as 0.1, and t is current time.

4.3.4 Generate Trial Summary

Figure 4.7 below shows the simple procedure used to evaluate trial results and Tables 4.2 and 4.3 show the data structures used to describe a summary of a trial and a trial definition.

1. Calculate the percentage of observables belonging to each class
2. Calculate the amount of material consumed and percentage of initially available protein consumed
3. Compute total costs as sum of direct, intangible and observation costs
4. Assess utility if desired

Figure 4.7: Evaluate trial results and generate trial summary

Name	Description
Time	Current time of observation
Number of results belonging to class1	Integer
Number of results belonging to class2	Integer
Number of results belonging to class3	Integer
Best trial statistics	Pointer to array containing best trial results so far
Observation cost	Cumulative cost for observing experiments in trial so far
Utility	Trial utility as sum of individual utilities of experiment (optional)

Table 4.2: Data structures for the items in a trial summary

4.3.5 Generate Global Information Summary

The data structures that describe the global summary are shown in Table 4.4. This in essence constitutes a summary of the global state in terms of parallel actions taken and their observed effects. The global information over time summarizes the feedback from the complex environment from multiple probes over time. Part of the global information includes the output from the C4.5 program that takes as input the description of all experiments so far and outputs a model description

Name	Description
Start Time	Time at which trial began
Number of experiments in trial	A real number
Search Type	Type of search (e.g., grid, partial factorial, etc.)
Variables	Pointer to array containing the set of control parameter values for each controllable variable
First experiment in this trial	Experiment number
Intangible Cost	Cost of intangibles for this trial
Setup Cost	Cost for setting up experiments
Resources Consumed	Pointer to resource quantities
Trial Summary	Pointer to table containing trial summary at each observation (Table 4.2)

Table 4.3: Data structures for items in a trial

Name	Description
Observation Time	Current time
Number of trials so far	Integer
Number of closed trials	Integer
Number of experiments so far	Integer
Number of closed experiments	Integer
Resources consumed	Pointer to resource quantities
Total direct costs	Direct costs consist of setup costs and material costs
Total intangible costs	Cost of intangibles so far
Total observation costs	Cost of observing all open experiments so far
Total costs	Sum of all three costs above
Utility	If desired (currently unused - will probably be a function of number of hits and time)
Number of hits	Total number of results of class 1
Best results	Pointer to array containing best results so far (difflim, observation time, experiment and trial numbers)
Boundary definition	Pointer to file containing C4.5 results

Table 4.4: Data structures for items in a global summary

or theory consisting of a disjunction of conjunctive rules that is consistent with the data seen so far. The features that represent any experiment and its partial result are encoded as a string of comma separated values for each of its control parameters, as well as time of observation of partial result, and finally the class of the partial result (Class 1 means good quality crystal, class 2 refers to poor quality crystal of precipitate and class 3 means no observable change was found).

4.3.6 Agent Architecture

One of the key aspects of this PEP framework is that it includes an agent who is the experiment designer in the loop. At the present time, the agent is a human. Below I explain the characteristics of an agent that resides within the loop in terms of the types of inputs (percepts) that will be received and the types of actions that the agent will perform through its effectors (such as hands that can enter information into a computer from a keyboard, or set up new experiments in the lab).

Agent Function

The inputs to the human agent involve the percepts from the environment -

1. Raw observation labels for each experiment at time of observation,

2. Summaries for each experiment, trial, and global state, which contain costs as a sum of setup and observation costs for each experiment, trial and global state information, as well as the amount of resources consumed, and

3. A representation of the evolving boundary in the form of a disjunction of conjunctive rules that describe a model of overall protein response behavior over different times of observation over all experiments performed so far until current observation time.

Assume that the agent has knowledge of its perceptual history and uses some policy[2] for deciding its next action. We also assume that the human agent is able to ascertain whether the goal has been achieved based on the feedback from the environment, that is the percepts.

The outputs or actions that the human agent within the PEP framework performs upon its simulated environment are:

1. Exit (that is, say no to all possible actions) OR

2. Re-open or close experiments or trials for next observation (optional) AND

3. Design a new trial AND/OR observe experiments after some time period.

4.4 Summary of PEP Framework Implementation

Several types of evaluations need to be performed with respect to partial results in order to serve different overall goals for the PEP framework. The data structures identified and represented as shown in Tables 4.1 - 4.4 enable the PEP environment to keep a record of experiment, trial and global state summaries.

The types of goals for which this framework was designed are:

G1: Identify and label experiments as belonging to particular classes (1, 2 or 3) at time of observation based on partial results.

G2: If crystals are found, identify and report the experimental conditions that seem favorable.

[2] See Figure 5.21 for a commonly used policy that I elicited from discussions with several crystallographers.

G3: Identify the boundary between the classes along the most influential variables. The most influential variables themselves will need to be identified.

G4: Calculate the costs involved for setup and observations of each experiment and the various trials, and use them for deciding what types of tradeoffs to make at each decision-step.

G5: Keep track of resources that are being consumed, particularly that of protein. This will directly influence strategy.

G6: Store summaries of effects of parallel actions in a data structure (such as the global summary), for analysis of strategies.

Goal 1 (G1) is achieved based on the final quality outcome that is obtained from one run of the simulator. Usually class 1 or good crystals are outcomes of 3.0 or less. Class 2 refers to either bad crystals or precipitates that are outcomes between 3 and 100. Class 3 refers to an outcome that indicates a clear result - 1000.

Goal 3 (G3) reflects the identification of the boundary between the classes of observed outcomes. This goal is achieved using inductive machine learning programs such as C4.5 and RL, that provide a human-understandable description of the boundary based on all the experiments and outcome labels seen so far. Identification of boundary is an important aspect of being able to come up with some internal model of how the given protein seems to be responding under different experimental controls. This gives some insight into possible rates of change of various hidden variables that influence crystal nucleation and growth such as the saturation level. Human agents do not necessarily think about the process in terms of the kind of influence some input controls might have on hidden variables. They focus on the kinds of boundary separations they observe, and based on their knowledge of the process based on experience, they tend to make reasonable hypotheses that influence their search toward more profitable areas.

The methods used for achieving G1 and G3 provide the intelligent assistance for designing crystallization experiments. The remaining goals are met by the PEP information management system as described in this chapter.

Chapter 5

Simulation of Hypothetical Macromolecules

In order to utilize existing theories about the process of macromolecular crystallization for practical assistance in experiment design, it was necessary to develop, implement and evaluate a simulation model. With this goal in hand, we developed approximate[1] physical models underlying the simulation in two stages. In the first stage of development, we focused our efforts on building an over-simplified model that had the right flavor of simulating the process of vapor diffusion, which is the driving process, but did not have sufficient complexity to model naturally occurring phenomena such as nucleation and crystal growth. Once we were satisfied that our simple model had been implemented correctly and was behaving as expected by the domain experts, we proceeded to the second stage of developing physical models to mirror nucleation and growth of macromolecules. The complexity of the model developed in stage 2 provided a more accurate representation of crystallization behavior of real-world macromolecules. Though some choices made in the model can be justified by theory, many others though reasonable, cannot be strongly justified based on physical theory.

5.1 Overview of the Crystallization Process

Crystallization from solution, like the formation of ice from water, represents a phase change, and constitutes a discontinuity in the behavior of a system. It occurs because an energy barrier has been surmounted and the phase transition leads to a more favorable state of the system with respect to energy.

As with any science, even though theory and practical experimentation are expected to go hand-in-hand, this is not always the case in the domain of macromolecular crystallization. The goal of experimentation in this field is to grow a good quality X-ray diffractible crystal of a protein or DNA

[1] The same terms imply different meanings to the two communities of computer science and physical chemistry. To the computer science community, what we have developed in this work can be described as a physical model. To the physical chemists, we would like to state that this model is not rigorous from a physical chemistry point of view, but rather represents a method for eliciting computational behavior that has a flavor of the real process of crystallization.

in solution. As very little is actually known about how the different experimental variables interact in solution, the experiment design is based more on trial-and-error and reliance on past experience in setting up and performing experiments, rather than theory. Many scientists believe that the theory, though weak, can and should be used to guide experimentation, especially as crystallization from solution is a sequence of events that occur more or less consecutively but are rarely completely unconnected [9].

Even though a complete theory of the crystallization of macromolecules is unknown at the present, basic theories of phase changes are well understood. In broad brush terms, phase transition behavior of macromolecules is similar to other molecules, as is evidenced in Rosenberger et al. [59]. The problem with macromolecules is that there are many solubility parameters[2] that are idiosyncratic such as vibration shock that is applied to an experimental tray during transport. We are trying to define a subset of the solubility parameters that capitalize on those areas where theory is strong: this way we can model different types of proteins on the basis of sound theory and use variations in solubility parameter definitions to bring about the idiosyncrasies in individual protein behaviors being simulated.

A crystallization set-up consists of an initial set of chemicals mixed together in varying concentrations at a particular temperature and pH, along with protein. The rest of what happens is basically a continuously evolving process, where the supersaturation of the solution increases or decreases, triggering events such as nucleation and growth of crystals. The level of supersaturation affects the rate at which crystals grow and directly impacts the quality of crystal obtained from experimentation. A widely adopted method for experiments in crystallization is the vapor diffusion hanging drop method. The principle and practice of the method are described below.

5.1.1 Principle of Vapor Diffusion

A droplet containing the biological macromolecule, buffer, precipitating agent and any additives in solution, is equilibrated against a reservoir containing a solution of precipitating agent at a higher concentration than the droplet. Equilibration proceeds by diffusion of water or organic solvent until vapor pressure in the droplet equals the one in the reservoir. If equilibration occurs by water exchange from the drop to the reservoir, it leads to a droplet volume change. Consequently, the concentrations of all the constituents in the drop will change [20].

Figure 5.1 provides the schematic representation of the hanging drop vapor diffusion method as well the physical rectangular plate/tray apparatus used to perform experiments in the laboratory. The tray is usually a 4x6 grid plastic box with 24 cells each containing reservoir solutions. The drop hangs from the top of a cover slip, and contains the biological macromolecule solution. It must

[2] Solubility parameters refer to the experimental parameters that affect the solubility of the macromolecule. For a complete list of parameters see Appendix A.

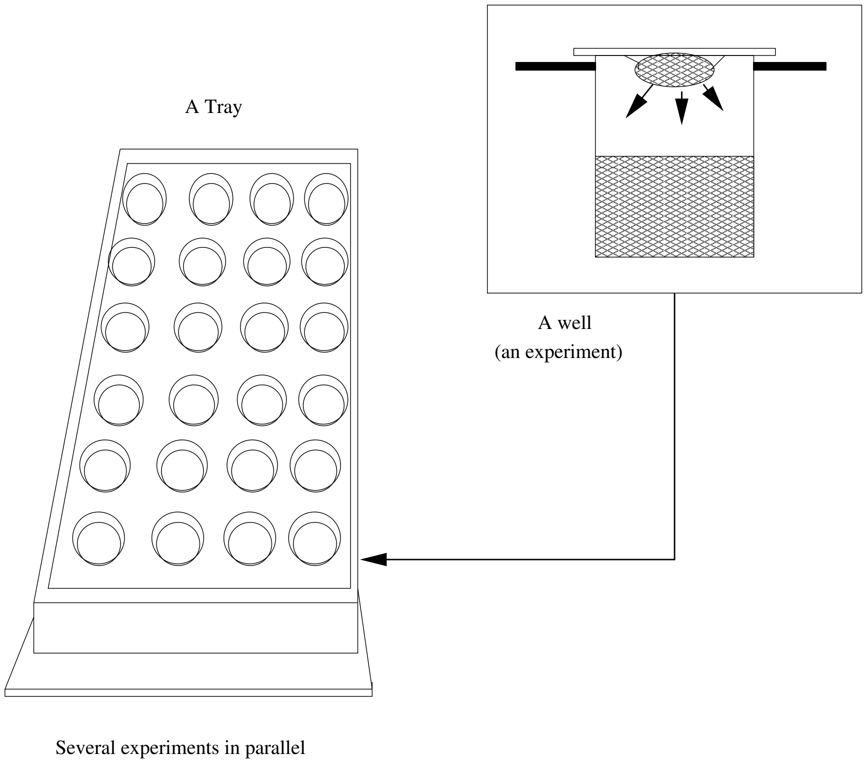


Figure 5.1: Hanging drop vapor diffusion method

be noted that there always exists a small leak from the tray as there is a seal, which cannot be guaranteed to be perfect. So, the vapor pressure in the cell is also equilibrating with the ambient pressure in addition to the vapor pressure in the drop equilibrating with that of the reservoir. Figure 5.1 is redrawn from McPherson[44]. It shows a standard configuration for hanging drop protein crystallization experiments. A plastic plate provides 24 wells for reservoirs of about 0.5 to 0.75 ml. Each well is an experiment and can be covered by a coverslip with a drop of protein solution hanging from its underside. The drop of 2-20μl equilibrates with the reservoir solution over time through the vapor phase, causing precipitant and protein concentrations to increase in the drop, and nucleation or the induction of crystallization to occur. Each well or experiment is observed under a microscope periodically to observe effects.

5.2 Description of our Initial Model for Simulating the Vapor Diffusion Process

Our model accepts as input values for protein given parameters, such as its name and isoelectric point, and simulation parameters such as the phase information for the protein being simulated. It also accepts solubility table definitions that describe the solubility of this protein in individual reagents and conditions such as pH. A set of control variable values are also input that specify the initial values of various concentrations of chemicals and the initial pH. We have modeled solubility of a hypothetical protein in 4 commonly used salts and 3 commonly used precipitants[3]. Another set of inputs include the values for the driving equation that describes the rate of change of water volume in the hanging drop (see equation 5.1). The time for which the simulation is to be run is also specified.

The model then simulates the changes (at different rates) in concentration of the various chemicals and protein within the drop, and updates these values at discrete intervals[4]. The main hidden variables in the model are the solubility and saturation of each phase of protein. The saturation of a particular phase of protein is directly proportional to protein concentration in solution and inversely proportional to the solubility of protein at that time. If at any time, the saturation of a particular solid phase exceeds the input saturation required for nucleation (nuc_sat), a solid mass forms with size equal to 0.01. If saturation of that phase is greater than that required for growth, the solid mass grows in proportion to its size. Otherwise if the solution is undersaturated (that is, saturation is below 1), then the solid mass shrinks.

We had to make several simplifying assumptions in our model of the process of vapor diffusion described in Section 5.1. In the first stage of model development, we tried to mimic mainly the water vapor exponential decay (see equation 5.1), with two driving components, namely the rate at which things were changing towards equilibrium between the reservoir solution and the drop containing the macromolecule (the equilibration rate) and the rate at which some vapor was leaking from the seal of the physical apparatus (the leak rate). We used discrete time units (which could be varied - we experimented with hours and five-minute intervals). At each time unit, the volume of the sample would be calculated using the water exponential decay equation. Then, the solubility of each solid phase of protein would be calculated as a function of 3 solubilities namely, solubility of that phase with respect to salt concentration, precipitant concentration, and the ionic concentration. In the first stage of development, we actually used graphs containing straight-line approximations of functions to describe solubility information of solid phases of a hypothetical protein. These solubility graphs were later described by smooth mathematical functions and constitute the major domain knowledge for driving the simulation.

$$V_t = (((V_{init} - V_{equilibration}) \times e^{-\frac{t}{\tau_1}}) + V_{equilibration} - V_{leak}) \times e^{-\frac{t}{\tau_2}}) + V_{leak} \tag{5.1}$$

Equation 5.1 models the exponential decay of water volume in the hanging drop with two exponentials τ_1 and τ_2, where τ_1 is the fast relaxation time (or equilibration rate, usually 1 week) and τ_2 is the slow relaxation time (or leak rate, usually 3 weeks).

[3] We use names such as Baby PEG, Mama PEG and Grandpa PEG to depict PEGs of different molecular weights. Baby is the lower molecular weight PEG. PEG is a commonly used precipitant and is the acronym for Poly Ethylene Glycol.

[4] Our initial model updated concentrations hourly. An hour was found to be very long, and later after experimentation with model output, this interval was changed to every five minutes of simulated time.

The solubility tables appear to be a good way to represent the behavior of a particular hypothetical protein with respect to different experimental control variables. They have the potential of being a very flexible representation for describing different functional behaviors of a protein in different precipitants and salts. In our model, we generally use the Error and Gaussian distribution functions[5] [52] to represent the behaviors of solubility of a particular solid phase of protein in different salts and precipitants as a function of their various concentrations. The solubility table parameters represent the values such as mid-points and standard deviations, required by the individual distribution functions. Sample solubility graphs are shown in Figures 5.2 and 5.3 below.

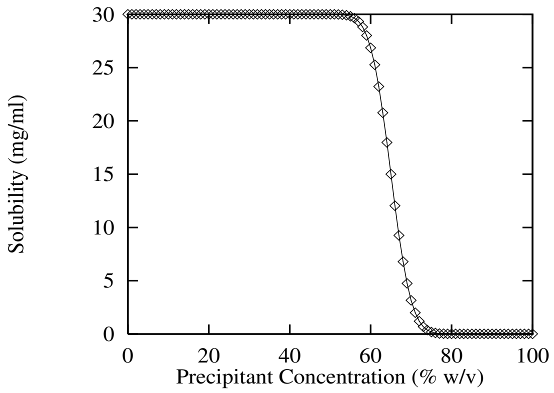


Figure 5.2: Solubility of the crystalline phase of protein with respect to **Baby PEG** concentration

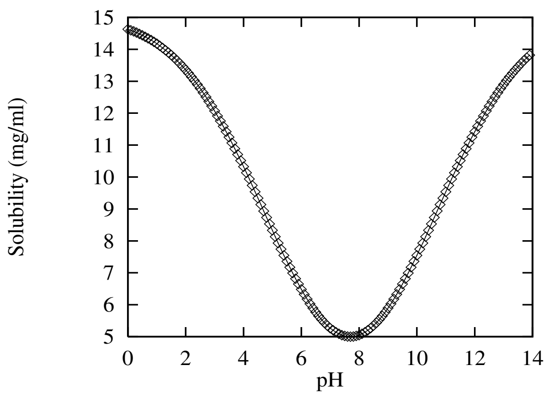


Figure 5.3: Solubility of the amorphous phase of protein with respect to **the pH** of the solution

The sample solubility graphs depicted in Figures 5.2 and 5.3 are represented as mathematical functions with parameters described as solubility tables. For example, the solubility graph depicted in Figure 5.2 is represented as the complement of the error function[52]. We represent the corresponding parameters as values for the midpoint of the x axis, the start and end values for the y axis and the slope of the error function (negative for complement of the function). Thus, the solubility table parameters corresponding to the graph in Figure 5.2 is represented as:

[5] All the mathematical function codes were obtained from Numerical Recipes in C.

```
PEG_table[2].phase[1] = 'C';
PEG_table[2].xy_values[1].x_midpt = 65;
PEG_table[2].xy_values[1].y_start = 30.0;
PEG_table[2].xy_values[1].y_end = 0.01;
PEG_table[2].xy_values[1].slope = -3.0;
```

Similarly, the inverse Gaussian function shown in Figure 5.3 is represented as a set of parameters for the Gaussian function, namely the mid point value for x axis, the highest and lowest y values and the sigma or standard deviation that specifies the slope at the mid-point of x axis.

```
PH_table.phase[0]  = 'A';
PH_table.xy_values[0].x_mid = initial_state->protein.prot_PI;
PH_table.xy_values[0].y_level = 15.0;
PH_table.xy_values[0].y_min = 5.0;
PH_table.xy_values[0].x_sigma = 3.0;
```

For purposes of our modeling, we have incorporated into the solubility tables for a hypothetical protein called Alpha Globulase, numbers that were obtained from solubility graphs provided by our domain expert (see Appendix C). The difficultly in obtaining actual solubility table information arises due to the fact that such information will have to be gleaned from actual experimentation in the laboratory by a physical chemist. Solubility graphs can be extrapolated from points that depict solubility of the solid phases of a given protein in each precipitant or salt at different concentrations by performing several simple experiments in the laboratory with very small amounts of protein material. Thus, in theory, it is possible to obtain such solubility information that can be used as input for our model.

The interaction between the different solubility functions is specified by a simple equation (5.2) that multiplies the solubilities of protein in each of precipitant, salt, and buffer (pH). This is a simplifying assumption that the overall solubility can be represented by multiplying the solubility of protein in each of the individual independent variables. The overall solubility of protein in a solution is highly non-linear due to the interactions among the individual independent variables. Our simple representation of overall solubility as multiple of individual solubilities works well for our simulation purposes (in terms of expected output behavior for given inputs). Future research might well involve variations to this equation for describing interactions in this highly multidimensional parameter space.

$$Solubility(phase) = \sqrt[3]{Solubility(phase)_{[PEG]} * Solubility(phase)_{[Salt]} * Solubility(phase)_{[Ionic]}} \tag{5.2}$$

where the concentrations of precipitating agent (i.e. [PEG[6]]), salt and ions affect the solubility of a particular phase of protein as given by the solubility table information and domain knowledge of relative impact of the different reagents for each type. The concentration of ions in solution depends on the salt concentration and buffer (that is, the pH). Their relationship in defined in Figure C.17 in Appendix C.

At each time unit of simulation, the solubility of each solid phase of protein is calculated (see equation 5.2 as a function of its solubility in solution containing precipitating agent, salt and buffer (pH).The concentrations (indicated as $[]_{time}$) of these chemical reagents change as a percentage (according to the ratio of initial volume of sample to current volume of sample at time t) of their initial concentration (see the equations for more details). Once solubility of a solid phase is calculated, the saturation of that particular solid phase of protein in the sample is calculated as the protein concentration in solution divided by solubility of this solid phase at this time (see equation 5.3). Protein concentration is updated as given in equation 5.4. When the saturation of protein in the sample reaches the saturation required for nucleation to happen, a nucleus would appear with solid mass approximately 1 % of existing protein concentration in the solution. The flowchart shown in Figure 5.5 outlines what happens next. If the saturation level is below 1, then the solid mass starts shrinking. If saturation is above 1 (supersaturated),and the saturation level is greater than required for growth, the nucleus begins to grow and accumulate mass in the solid phase at the expense of the liquid phase. The output of stage I model is simply the development of solid mass, that is, the transfer of protein from liquid to the solid phase.

Equation 5.3 represents the calculation of saturation of a particular solid phase of the given protein at time t. Saturation is directly proportional to the concentration of protein in solution at that time, and varies inversely with the solubility of that phase at time t. Solubility is calculated as per equation 5.2, where we multiply the individual solubilities at different concentrations of precipitant and salt, and ionic strength, and take the cube root. We found that this assumption for how solubilities influence one another to produce the desired effects in the outputs from the simulator. Equations 5.4 - 5.6 depict the change in concentrations of the protein, salt and precipitant at time t as a function of the initial concentration and ratio change in water volume of the drop.

$$Saturation(phase)_t = \frac{[Protein]_t}{Solubility(phase)_t} \tag{5.3}$$

$$[Protein]_t = \frac{V_0}{V_t} \times \left([Protein]_0 - \sum_{i=1}^{\#nucleii} size(nucleus_i) \right) \tag{5.4}$$

[6]Poly Ethylene Glycol - a commonly used precipitating agent, available in several molecular weights - e.g., 200 -20000. PEG concentration is measured in units of % w/v that is, percentage weight per volume.

$$[Salt]_t = [Salt]_0 \times \frac{V_0}{V_t} \tag{5.5}$$

$$[Precipitant]_t = [Precipitant]_0 \times \frac{V_0}{V_t} \tag{5.6}$$

Figures 5.4 and 5.5 depict the first model that was developed and tested in stage I. In the first stage of development of the simulation model, the focus was on the driving process of vapor diffusion, and how to represent it. In the next stage, we included actual events such as nucleation and growth of crystals. Nucleation is a stochastic event, and so we included some element of randomness in our model. The flowchart in Figure 5.6 shows the change in the growth loop for model developed in Stage II. The main difference is in the nucleation process, where we account for two different types of nucleation that could possibly happen simultaneously or separately. The two types of nucleation that occur in nature are referred to as heterogeneous and homogeneous nucleation. I will defer further description of the stage II nucleation model, until I have described the nature of solids and the nucleation process in macromolecular crystallization.

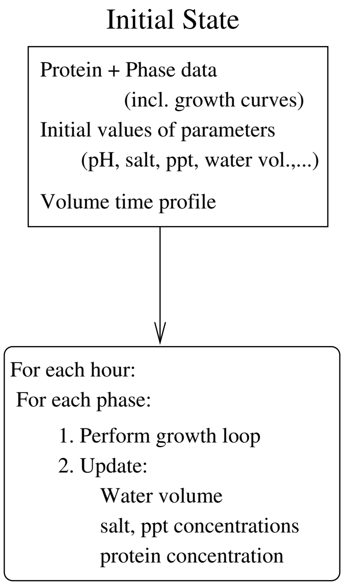


Figure 5.4: Initial (Stage I) model of vapor diffusion

5.3 Description of Solid States

There are two generally observable solid states for macromolecules: amorphous and crystalline. The distinguishing feature between them is the long-range order of molecules in the solid matter. Amorphous solids have short-range order that is aperiodic, while crystalline solids have

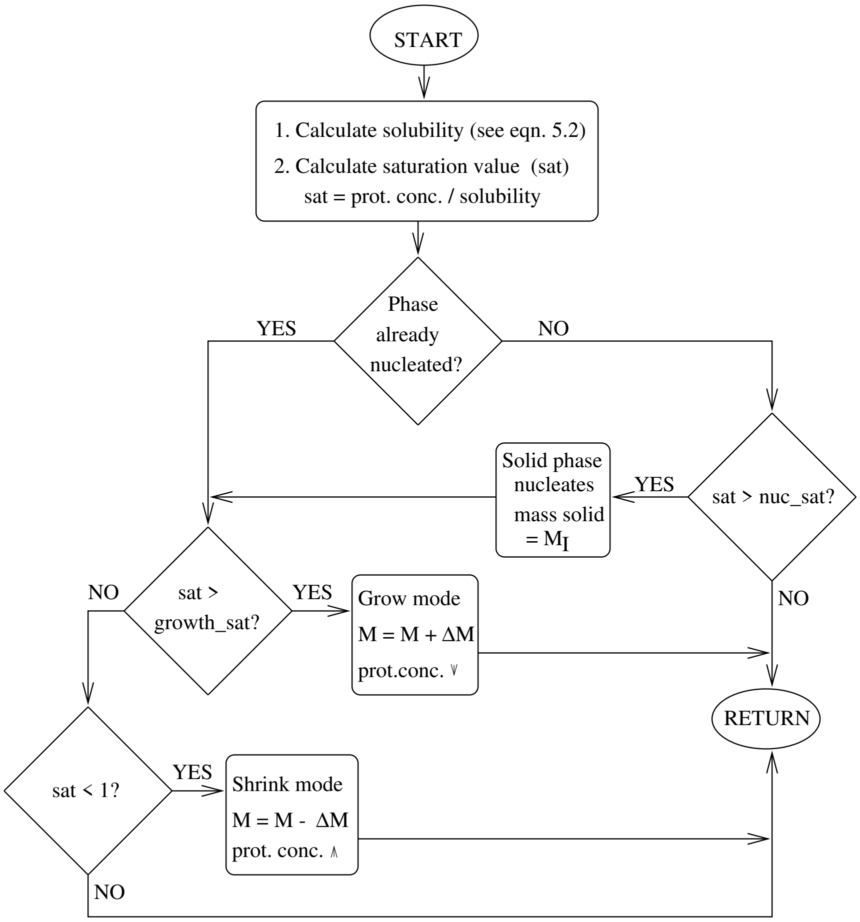
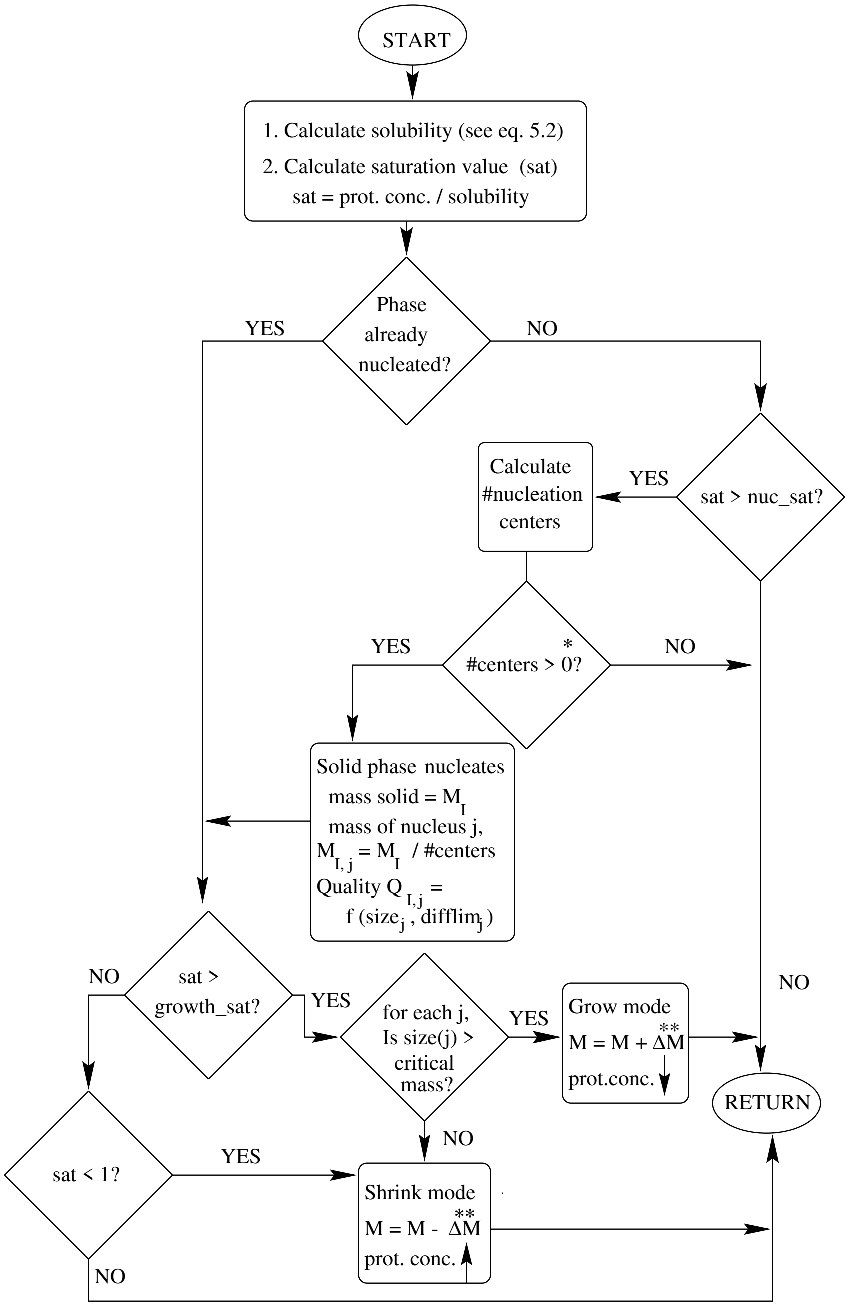


Figure 5.5: The procedure (Growth Loop) for growth of solid mass in Stage I model



* Homogeneous nucleation occurs with probability p over distribution d.
Higher the supersaturation => more #nucleation centers (i.e. we get a shower of microcrystals).

** ΔM = f(growth_rate)
growth_rate = f(saturation)

Figure 5.6: The inclusion of stochastic nucleation into Stage II model

long-range order that is periodic. For purposes of X-ray diffraction, periodicity is key since the diffraction pattern of highly periodic arrangements result in spots on the diffraction screen, making it easier for analysis(usually Fourier transforms). If one were to diffract an amorphous precipitate, one would get a highly stringy mess on the diffraction screen, as opposed to a pattern of spots, making it difficult to perform any kind of analysis to determine structure (Fourier transform is no good in this case).

An important question when discussing solid states of macromolecules is how they differ from other chemicals, both in terms of actual states of existence in the solid phase, as well as in difficulty of obtaining the solid states themselves. Although conventional solution-grown organic and inorganic crystals may appear superficially similar to macromolecular crystals grown from solution, there are many important differences in physical and mechanical properties, conditions for nucleation and growth, and the structures of the internal network of molecules that form crystal lattices [44]. The differences in properties, in turn, affect the way in which crystals are grown and used in the laboratory.

Some of these differences are evident simply by inspection. Conventional crystals can often be grown with dimensions of several centimeters, with no apparent limit on their final size. Macromolecular crystals, on the other hand, rarely exceed a millimeter on an edge and are generally much smaller. Macromolecular crystals could be grown to dimensions of centimeters, but it is very much harder to grow them to that length, due in part to the difficulties in obtaining and purifying such large amounts of protein. Equally significant from a structural analysis point of view is that protein crystals seem to reach a more or less terminal size and seldom exceed it. Why this happens still remains a puzzle. Protein crystals also tend to be extremely fragile.

A major distinguishing feature of macromolecular crystals is their high solvent content (ranges from 30% to 90% by volume). Most of this solvent is contained in channels and interstices that pass among the molecules in the lattice, and often there are large actual channels of solvent in the unit cells that approach the dimensions of the macromolecules themselves. We can readily diffuse small molecules through these channels into macromolecular crystals. This feature makes macromolecular crystals stand out as unique systems for conducting biochemical experiments, making them invaluable tools in enzymology and rational drug design. The active sites of enzymes, or other ligand[7] -binding sites, are frequently accessible to small molecules diffusing through the crystal lattice. Macromolecules in the crystal bind their ligands just as they would in solution, except in those rare cases where lattice interactions interfere.

We can think of the macromolecules in a crystal as islands, isolated from one another in a sea of water, but all interconnected by a network of delicate contacts (macromolecular contact

[7] A small organic molecule that binds to a macromolecule. Example of ligands are co-factors or metal ions such as germanium that bind to protein.

molecules are bound weakly because of the constant bombardment by solvent molecules due to the kinetics). Because contacts and ties among macromolecules in crystals are so tenuous, alteration of only one or few through chemical or physical change can dramatically affect the packing of molecules.

Amorphous phase is the kinetically more favored solid state, since aggregation of molecules occurs almost spontaneously under favorable supersaturation conditions, and it is very easy to form globs of precipitate. It is however, much harder to obtain a periodic arrangement of molecules that is favorable to X-ray diffraction analysis.

With respect to our simulation model, we have tried to incorporate the following thermodynamic (a and b) and kinetic features (c): (a) In general, there exists a range of conditions in which the crystalline phase is more stable than the amorphous phase, (b) These conditions are near the boundary[8] between clear (i.e., no observable result or change in liquid phase) and amorphous, and (c) If thermodynamics favor the crystalline phase, the kinetics could favor the amorphous phase. The kinetics are based on the rates of change of the various parameters used in experimentation.

It is much harder to obtain the solid states of macromolecules as opposed to simple inorganic or organic molecules, because of the vast number of parameters that can affect nucleation and growth, the efforts involved in terms of time and money to obtain sufficient quantities of protein and purify it, and the fact that a protein can denature easily. The next subsection describes the phenomenon of nucleation.

5.4 Nucleation and Growth of Macromolecular Crystals

Nucleation is a very complex phenomenon and even physical chemists do not understand every detail of the manner in which this event occurs for macromolecules. Some simple liquid-solid transitions such as that of water to ice is understood, but even in that case, the process of liquid condensation from the vapor phase is the phenomenon that is most well-understood. A leap of faith is made with respect to the phase transition from liquid to the solid phase. The reason for this level of understanding is because liquids do not possess an inherent shape, while solids do. So, while a sample of matter in its liquid and vapor states can be described in terms of its volume, pressure, temperature and amount of substance present in it, description of a sample in the solid state requires a study of its shape and symmetry, reflecting the arrangement of molecules in a structured fashion or lattice.

Any point on a phase boundary of the phase diagram of a substance represents a state of dynamic equilibrium between the two neighboring phases [3]. A state of dynamic equilibrium is one in which the forward process is taking place at the same rate as the reverse process. In the case of

[8] The biggest factor in crystal growth is time. Thus, even if a thermodynamically stable state is favored far from the boundary, it would take a much longer time to obtain such a crystal than one that is stable kinetically, due to the slow rate of change that drives the process of crystal growth.

liquid-vapor phase boundary, any point on this boundary represents a state of dynamic equilibrium in which vaporization and condensation continue at matching rates. Molecules are leaving the surface of the liquid at a particular rate, and molecules already in the vapor phase are returning to the liquid at the same rate so that there is no net change in the number of molecules in the vapor (and hence no net change in its pressure). Similarly, a point on the solid-liquid curve, represents a state in which molecules are continuously breaking away from the surface of the solid and contributing to the liquid, but they are doing so at a rate that exactly matches the rate at which molecules already in the liquid are settling on to the surface of the solid and contributing to the solid phase.

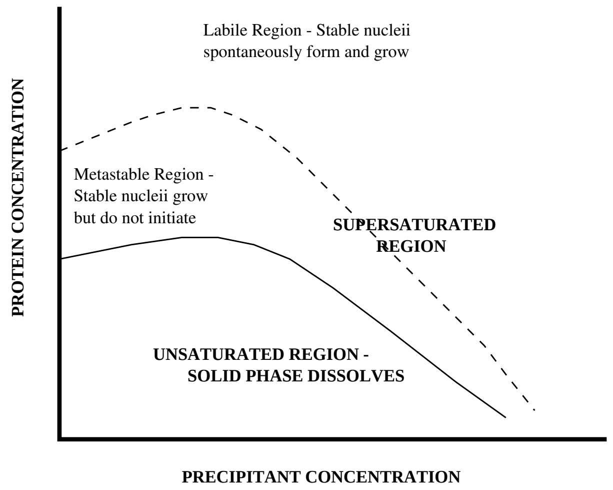


Figure 5.7: Two important dimensions of phase diagram for protein crystallization (from [21])

Figure 5.7 [21] represents a traditional phase diagram that reflects the most common technique for driving a macromolecule from solution, called *salting out.* This is similar to the evaporation of sea water to produce salt. The heavy solid line represents the equilibration line corresponding to a saturated solution. At concentrations of solute below the solubility limit for values along the horizontal axis, the system is said to be unsaturated. Above that limit, the region is called supersaturation. As we can see from the figure, the supersaturation required for nucleii to form is greater than that required for growth. Because several physical and chemical parameters may influence solubility, the phase diagram for a protein or nucleic acid is typically multi-dimensional. Our model takes as input this complex phase diagram and attempts to provide the response of that protein under varying experimental conditions. Each solubility table represents a portion of the real phase diagram of the protein. Each solubility graph has one variable such as precipitant concentration on the horizontal axis and solubility (basically protein concentration in mg/ml) values on the vertical axis.

For nucleation to happen, it is necessary for a sufficient number – the critical mass – of molecules that collide to aggregate together. Nucleation occurs when one or more aggregates exceeds the critical mass. The constant collisions of molecules result in a population of flickering aggregates of molecules within a solution, with a size distribution. The degree of saturation affects the mean of the size distribution and its variance. The molecules are continuously banging against one another with tremendous force, and the bombardment causes dislodgement of molecules from an aggregate. Below saturation, that is, when a solution is undersaturated, dislodgement or disaggregation is always favored. When the solution is above saturation or is supersaturated, above a critical size, aggregation is favored. Thus, a nucleus can form and grow without bound. The critical size, or the mean of the size distribution of aggregates in solution, becomes lower as saturation level increases. Hence, we observe the effect of spontaneous nucleation happening at a particular saturation level, as the critical mass has decreased to a point where aggregation of molecules is favored.

In homogeneous nucleation, we see a random collision of molecules forming aggregates. It is observed that in homogeneous nucleation, we often get a shower of micro-crystals. The process of nucleation is highly non-linear as can be inferred from the observed effects. Suddenly, there will appear a shower of micro-crystals and their number is proportional to the percentage of protein in the solution. This effect can be explained by using the following analogy. If we think of critical mass as a high-jump bar that is being lowered constantly as the level of saturation increases, and the flickering aggregates as many children hopping to various levels of the bar, then at some point in time, the bar will be low enough that a certain percentage of children could easily hop over. This analogy can be used to explain the occurrence of a sudden shower of microcrystals, and is reflected in our model. In heterogeneous nucleation, there is a substrate that binds molecules. Due to the existence of a substrate, heterogeneous nucleation is found to occur more often in real experimentation, as the aggregates formed due to binding tend to exceed critical mass earlier than those formed as a result of random collisions alone.

We have modeled nucleation in the simulator to reflect this process, by imposing a critical mass for any nucleus. This critical mass for any phase decreases as the saturation increases, thereby simulating the effect just described (see equation 5.7).

$$critical\_mass(phase) \propto \frac{1}{saturation(phase)} \tag{5.7}$$

Rosenberger et al. [59] have performed extensive studies with the macromolecule lysozyme[9] from hen egg-white that is easy to obtain, abundantly available and can be crystallized easily. Our modeling of nucleation and growth effects reflect many of the observations from their studies. For example, I quote directly from Rosenberger et al. [59]:

> The higher the supersaturation, the smaller the critical nucleus and, thus, the higher the *probability* for nucleation[14].

[9] Basically, Lysozyme is an enzyme that is used to break down cell walls of bacteria. The bacteria therefore burst. Lysozyme is part of the immune system in living organisms. Lysozyme is also produced by the tear-duct to fight against external organisms on the outer-eye. A hen-egg has plenty of lysozyme as its defense mechanism.

We have based our model of nucleation on a phenomenological explanation. Both, homogeneous and heterogeneous nucleation are modeled as stochastic processes that happen as a function of chance once the saturation required for nucleation has been attained. One of the input parameters to the simulator modeled in stage II is the number of heterogeneous substrates present in each of the phases that could possibly bind to protein molecules to form heterogeneous nucleii. Usually, this number is small(e.g., 3). In any simulation run, it is therefore possible to obtain either or both homogeneous and heterogeneous nucleii that form and grow in one or more of the solid phases. Homogeneous nucleation is modeled so as to result in the formation of a shower of microcrystals, based on the calculation of probability of occurrence of a homogeneous crystal per unit volume of the sample multiplied by the actual volume at time t of simulation.

In the case of homogeneous nucleation, the probability of finding a nucleus of size equal to the critical mass is dependent on the volume of the sample. The probability density function for calculating the probability of nucleii of particular size forming per unit volume of sample is modeled as a Gaussian (midpt, width), where midpt and width describe the aspects of size of nucleus, which varies directly as a function of saturation. After calculating the critical mass required for nucleating that particular phase, we then calculate the probability of finding a nucleus of size=critical mass by referring to the probability density function (equation 5.8). In this case of homogeneous nucleation (equation 5.9), we multiply the probability per unit volume by the Volume at time t, so as to yield the desired effect (of relating volume of sample to nucleation probability).

$$Prob\_per\_unit\_volume(homonuc) = gauss(critical\_mass(phase), homo.width, homo.mid) \tag{5.8}$$

The probability of homogeneous nucleation happening is modeled by

$$Prob(homonuc) = Prob\_per\_unit\_volume(homonuc) * V_t \tag{5.9}$$

where the probability per unit volume is calculated from a Gaussian probability distribution with variable width and mid-points (that depend on the saturation at any given time)- see equation 5.8.

In the case of heterogeneous nucleation, we start with a user-defined number of fixed heterogeneous nucleii. For each nucleii, we have a different probability density function, described as Gaussian(midpt, width), where midpt and width vary depending on saturation. For each heterogeneous nucleus that exists, we use equation 5.10 to determine the probability for it to nucleate.

$$Prob(heteronuc) = gauss(critical\_mass(phase), hetero.width, hetero.mid) \tag{5.10}$$

Growth rate of a nucleus has been found in practice to increase rapidly with higher supersaturation levels thereby yielding crystals of poor quality or precipitation. The growth-rate is modeled as a direct function of saturation (see equation 5.11). Rosenberger et al. [59] observe that smaller crystals grow slower than larger ones. Therefore, each nucleus grows or shrinks in proportion to its size. Figure 5.11 depicts our modeling of the growth phenomenon.

$$growth\_rate(nucleus) \propto saturation(phase) \tag{5.11}$$

In summary, we could have several nucleii in each solid phase of a protein, and they could be either homogeneous or heterogeneous. Homogeneous nucleation can produce several small nucleii of the same size - these would resemble the shower of microcrystals that are often found in nature and the laboratory.

5.5 The Characterization of Crystal Quality

X-ray diffraction provides data for a number of kinds of quantitative analyses of the relative quality of crystals grown under different conditions. The most important analysis for structure determination has been the resolution limit, or extent, of the diffraction pattern. Reflections in a diffraction pattern are obtained by directing a collimated beam of X-rays onto a crystal. Each intensity corresponds to the reflection of X-rays from a family of planes in the crystal. The limit at which diffraction intensities disappear for a particular crystal defines its resolution, and this in turn determines the precision of the structural model derived from that analysis.

Another feature of the diffraction pattern that tends to reflect the inherent order of a crystal is its mosaicity – that is manifested by the width, or spread, of the intensities. Mosaicity, in a sense, reflects a measure of the various kinds of physical defects in a crystal. The concept of mosaicity is reflected in our model as the B-factor, which is a number that reflects the internal order of the crystal. This internal order is usually best at time of nucleation, and gradually worsens as the crystal grows.

The resolution limit of diffraction (difflim) of a crystal is basically a number that reflects the quality of resolution of an X-ray diffractible crystal. This number is usually between 1.2 and 3.5 Angstroms, and lower numbers imply very good crystal quality (for example, a crystal with difflim 1.2 Angstroms diffracts very well, and is said to be a crystal of excellent quality). The final quality of a grown crystal is indicated by its difflim value and is calculated using equation 5.12. In the simulator model we reflect the internal order within a solid phase as the B-factor, which essentially indicates a number value that is best at the time of nucleation, and steadily increases as growth proceeds. If a crystal grows too rapidly, the B-factor becomes a very large number, leading to a larger final difflim value, indicating poor crystal quality with respect to X-ray diffraction. Growth rate of a crystal is modeled as a function of saturation. If the supersaturation level is very high, the growth rate is also high. The calculation for final difflim is based on two main factors that determine the quality of a grown crystal, namely, its internal order (represented as the B-factor), and its size (or the illuminated volume for X-ray diffraction analysis). The equation for calculating the final crystal quality or difflim is shown below.

$$d = f \times \sqrt{\frac{B}{lnV}} \tag{5.12}$$

where d = resolution limit of diffraction, f = a real constant, B = B-factor (internal order reflector) and V = size of crystal (illuminated volume for X-ray diffraction analysis).

5.6 Flowcharts and Pseudo-code for Simulation Model in Stage II

The flowchart describing the model is shown in Figures 5.8, 5.9, 5.10 and 5.11. The equations used for various calculations within each flowchart are referenced appropriately.

MAIN

1. Obtain initial state.
2. Set up domain knowledge into tables.
3. Initialize structures.
4. For number of hours simulation is to be done:
   Perform the following:
   (a) Calculate Volume at time t.
   (b) For each phase i do:
       PHASE CALCULATIONS (curr-state, i).
   (c) Update current state.
5. Print final outcomes at end of simulation.

PHASE CALCULATIONS (curr-state, i)

1. Calculate Solubility[i]
2. Calculate Saturation[i]
3. If (num-nuc-in-phase[i] > 0)
   Then Perform GROWTH CALCULATIONS (curr-state, i)
   Else Perform NUCLEATION CHECK (curr-state, i)

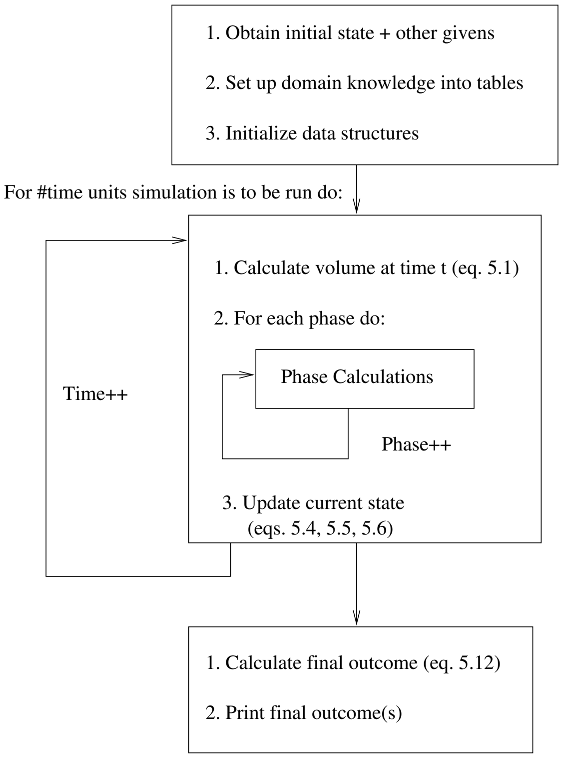


Figure 5.8: Flowchart describing the model of vapor diffusion - Main Flow

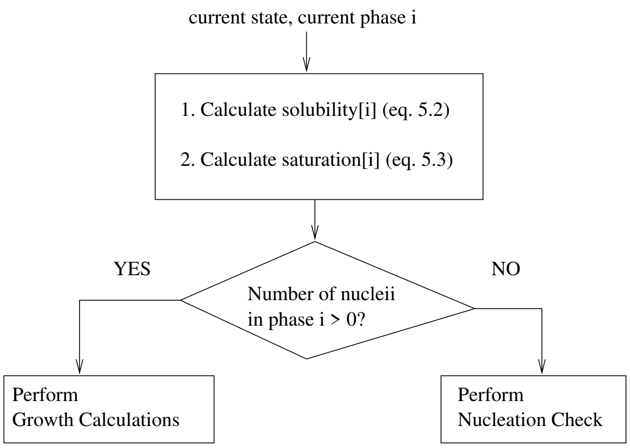


Figure 5.9: Phase Calculations

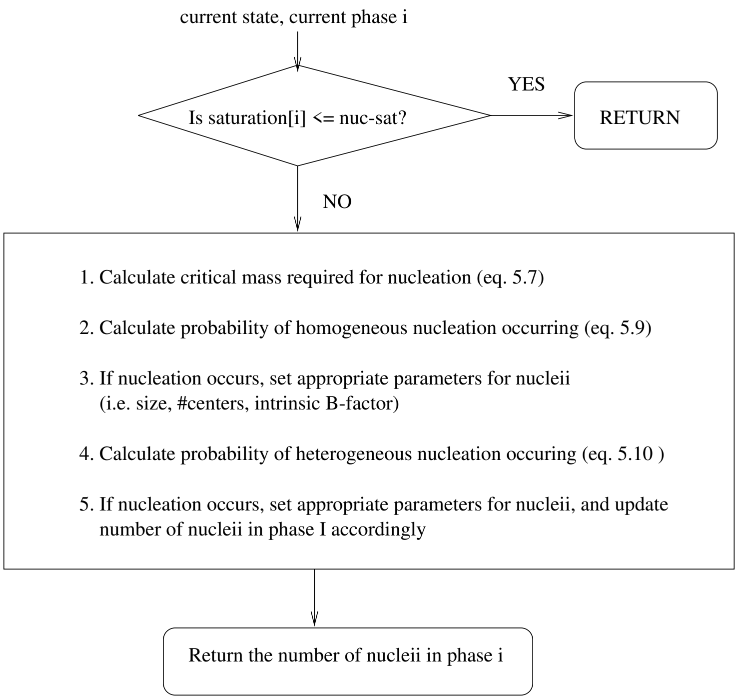


Figure 5.10: Nucleation Check

NUCLEATION CHECK (curr-state, i)

1. If (saturation[i] <= nuc-sat) return;
2. Calculate criticalmass = calc-nuc-mass (i, saturation[i])
3. Get-nuc-values (i, saturation[i]);
4. Calculate probability of homogeneous nucleation (i.e. having nucleii of size=criticalmass)
   (a) Get probability-per-unit-volume= gauss(criticalmass, homowidth, homomid)
   (b) Multiply probability-per-unit-volume by Volume at time t (V-time)
   (c) Homogeneous nucleation then occurs with prob from (b)
   (d) If nucleation occurs, set appropriate parameters for nucleii
5. Calculate probability of heterogeneous nucleation
   (a) We start with a user-assigned fixed # of hetero nucleii
   (b) For each nucleus, we find probability of nucleation from gauss(criticalmass, hetero[nuc].width, hetero[nuc]. midpt);
   (c) If any of them nucleate, update parameters according (i.e. size, nuctype, difflim, and num-nuc-in-phase)

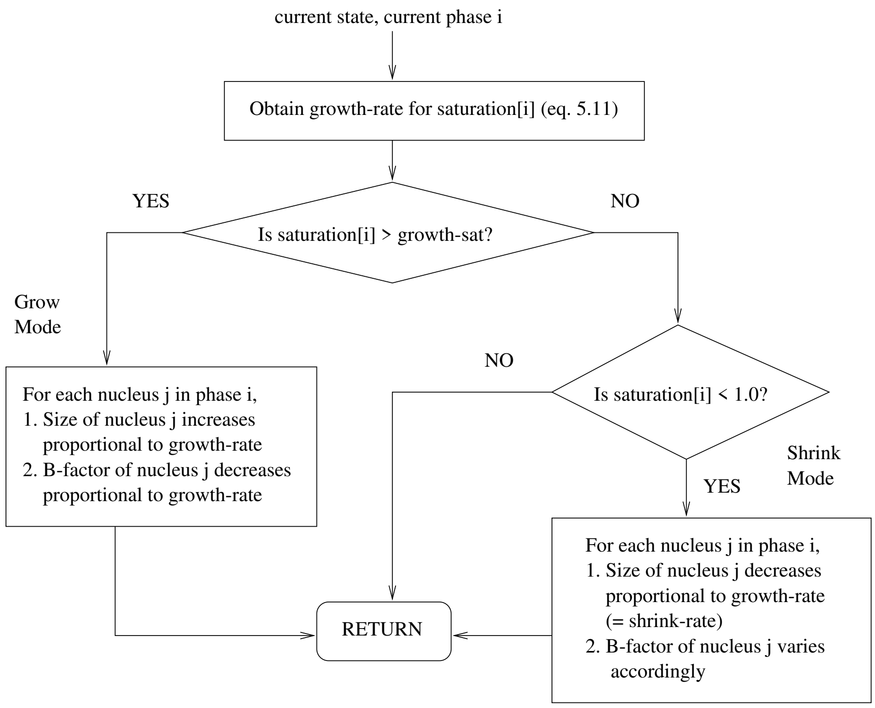


Figure 5.11: Growth Calculations

GROWTH CALCULATIONS (curr-state, i)

1. Get growth-rate(i, saturation[i])

2. if (saturation[i] $>$ growth-sat)

   then in growing mode: for each nucleus in phase i,

   size of nucleus increases proportional to growth-rate;

   else if (saturation[i] $<$ 1.0)

   then in shrinking mode: for each nucleus in phase i,

   size of nucleus decreases proportional to growth-rate;

The following section describes the sensitivity analysis of the parameters that comprise the model for macromolecular crystallization. As the number of inputs to the model is enormous (around the order of hundreds of numbers), we decided that we would try to focus on those parameters that impacted the types of response behaviors that the domain expert expected to see.

5.7 Sensitivity Analysis

This simulator can be used to produce two different types of outputs - a single real number representing the quality of its best nucleus or best result at a particular time of observation, and several graphical outputs that show the changes in the various chemical concentrations and the nucleation and growth of solid mass or nucleii in each solid phase of the protein. The graphical outputs were used for subjective evaluation of the model by domain experts.

In this section, I first illustrate the graphical outputs from each run of the simulator that models the behavior of a single experiment. We have tried to plot several different variables in one graph, in order to depict the process. We first choose one precipitating agent, namely Baby PEG or low molecular weight PEG, and show the graphical output from a simulator run that has no heterogeneous nucleii. All graphs except Figures 5.19 and 5.20 show only the total solid mass that is the sum of several microcrystals that form a homogeneous nucleus. Figures 5.19 and 5.20 show how different nucleii could exist simultaneously in the different solid phases of a protein. The hypothetical protein is termed Alpha Globulase, with solubility table information based on solubility graphs shown in Appendix C.

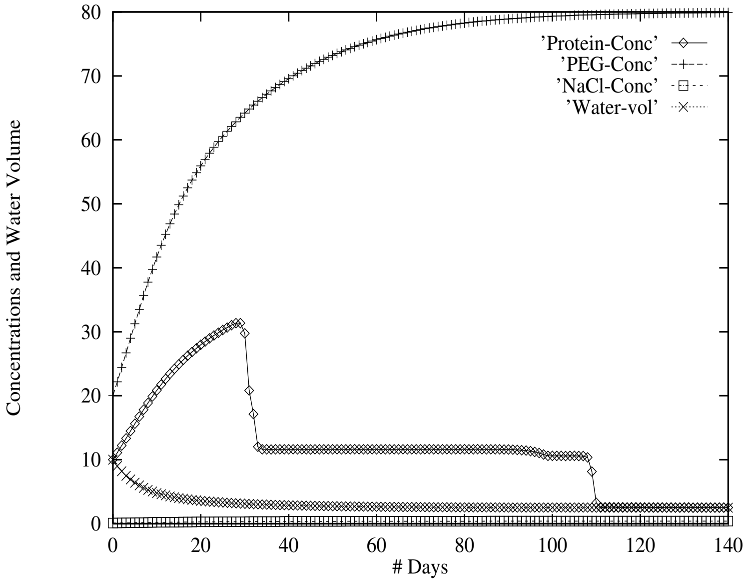


Figure 5.12: Control parameters with 20 %w/v Baby PEG, 10 mg/ml protein, 0.1 M NaCl.

The graph in Figure 5.12 illustrates the change in the control values of four of the parameters that influence the process. The exponential decay of the water volume in the hanging drop is shown. Salt concentration is a very small value and increases with time. Figures 5.13 and 5.14 are graphs that depict the exponential decay of water volume in the drop, and the rise in salt concentration.

There are several factors of the exponential decay that can vary the output from the model. In this example, we have chosen a fairly large size drop, that is 10 μl. Normally, a size between

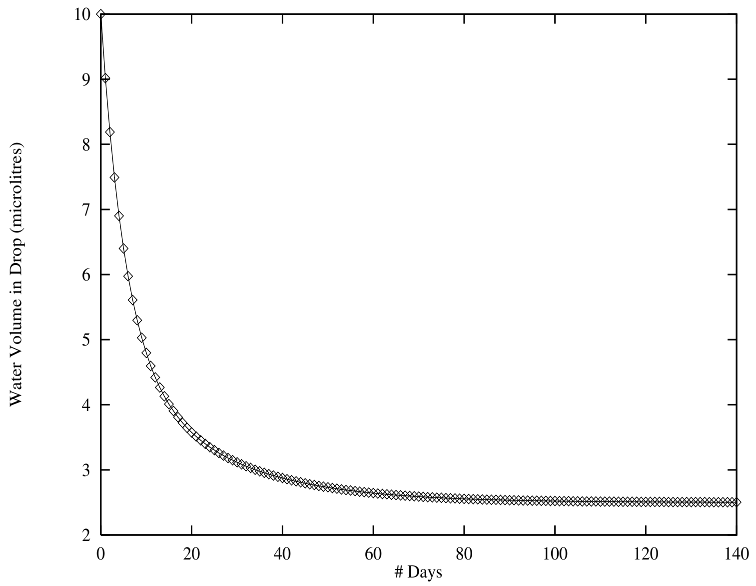


Figure 5.13: Exponential decay of water volume in hanging drop

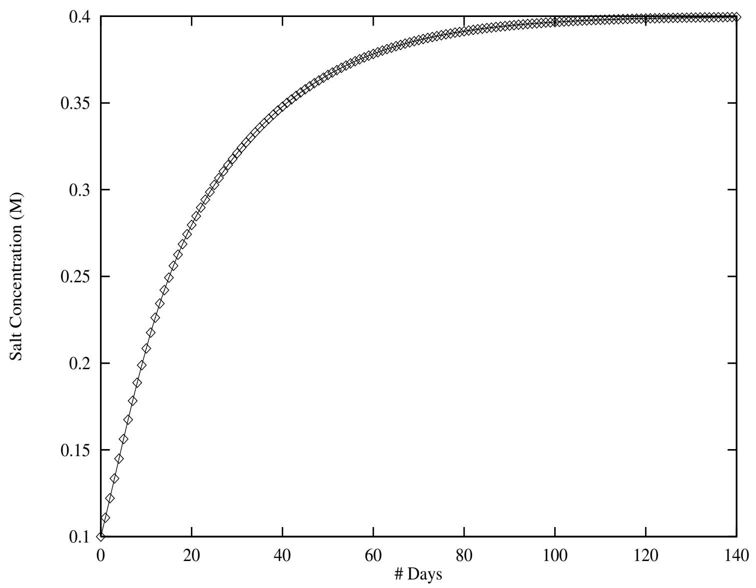


Figure 5.14: Increase in salt (Sodium Chloride) concentration with time

1 and 10 μl is chosen. However, for the purposes of testing our model, this was acceptable. The volume at which equilibration occurs is usually set at half the volume of the original drop. Thus, we defined $V_{equilibration}$ to be 5 μl. The volume of water that leaked from the seal was set to be equal to half $V_{equilibration}$. Thus, V_{leak} is set to 2.5 μl [10]. The rate at which water vapor leaks is represented by the second exponent (τ_2) in equation 5.1 and is normally set to 3 weeks. The rate at which equilibration occurs is represented by the first exponent (τ_1) in equation 5.1 and is set to be 1 week for Alpha Globulase. By varying the two exponents and the volumes in the water decay equation, we can simulate different rates at which the crystallization process takes place.

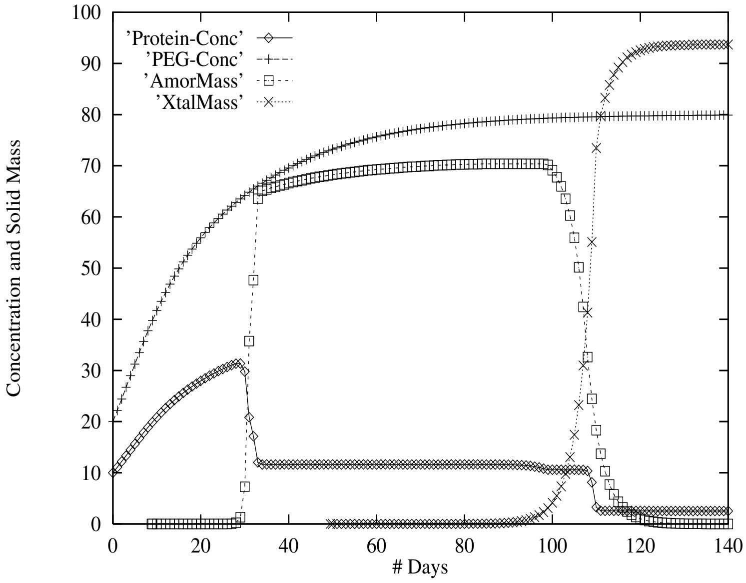


Figure 5.15: Appearance of solid phases in 20 %w/v Baby PEG, 10 mg/ml protein, 0.1 M NaCl

Figure 5.15 depicts the appearance of first the amorphous solid around 10 days after setting up the experiment, and its subsequent growth. The crystalline phase nucleates at approximately 50 days after setup. The crystalline solid phase grows at the expense of the amorphous solid phase. Thus, we can see the amorphous solid mass shrinking as the crystalline solid mass increases in size steadily starting just prior to day 100.

Figure 5.16 depicts the outputs from another experiment initialized with exactly the same parameters in the first experiment, with a small change. The precipitant concentration is increased by 5 % w/v (weight/volume). We can see the difference in two graphs – 5.15 and 5.16 – fairly obviously. The saturation levels required for nucleation and growth of the crystalline phase are achieved earlier in time.

[10] It is interesting to note that we had originally set V_{leak} to be 1.0 μl, but given that we use equation 5.1 to simulate the exponential decay process, we found that this would result in a slight increase in water volume initially, which should not happen. Increasing the V_{leak} parameter appropriately ensured that this would not happen.

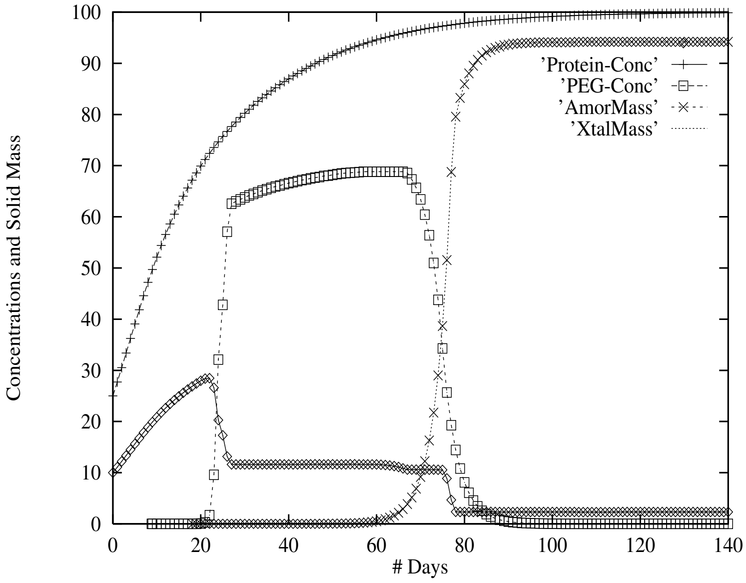


Figure 5.16: Appearance of solid phases in 25 %w/v Baby PEG, 10 mg/ml protein, 0.1 M NaCl

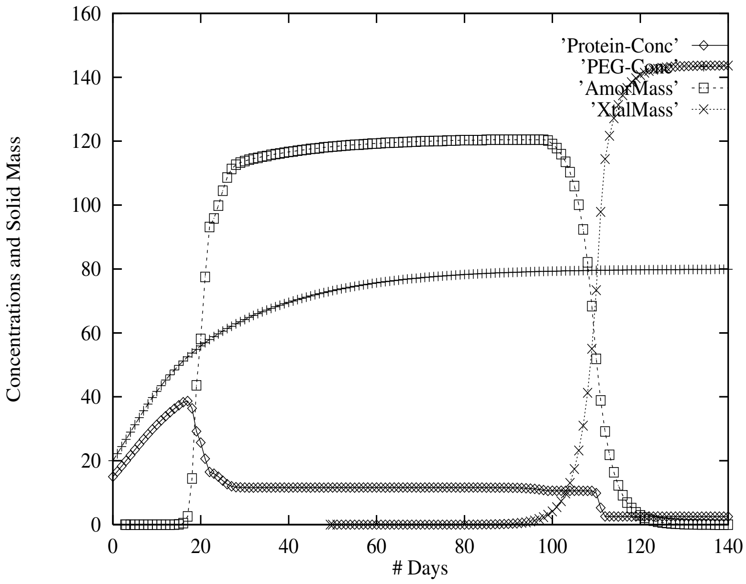


Figure 5.17: Appearance of solid phases in 20 %w/v Baby PEG, 15 mg/ml protein, 0.1 M NaCl

Figure 5.17 represents the output from an experiment with the same values as the first experiment. Instead of increasing the precipitant concentration, if we increased the protein concentration by 5 mg/ml, we find no difference in terms of the time taken to grow the crystalline solid phase. What we see is that the amorphous phase nucleates much earlier, almost within a day or two of experiment setup. We also notice the increase in the overall solid mass obtained in each phase, as there is more protein available.

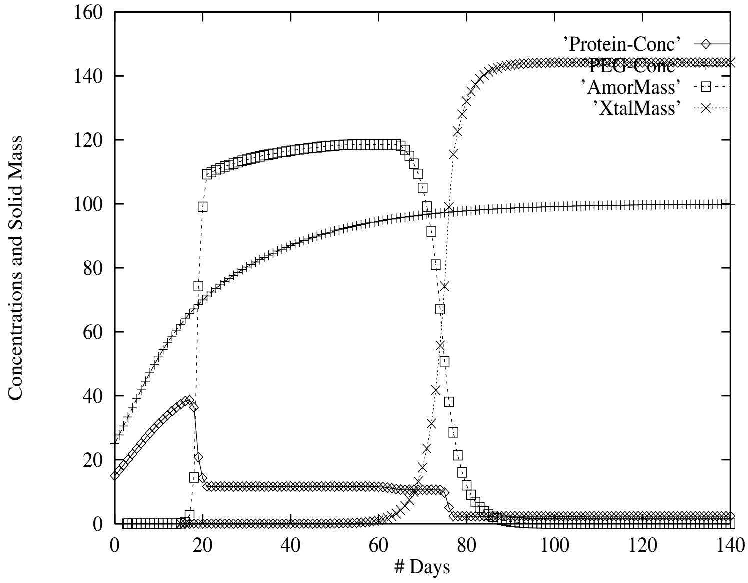


Figure 5.18: Appearance of solid phases in 25 %w/v Baby PEG, 15 mg/ml protein, 0.1 M NaCl

Suppose we were to increase both precipitant as well as protein concentration – Figure 5.18 represents the output from such an experiment, in which precipitant concentration was increased by 5%w/v and protein concentration was increased by 5 mg/ml. Here we can see clearly that both the amorphous phase as well as the crystalline phase nucleates sooner, and the size of the solid mass in each phase is bigger. This result shows the consistency in the behavior of the model.

5.7.1 Effects with Presence of Heterogeneous Nucleii

Now, suppose we indicate to the simulator that a heterogeneous nucleus was present in each phase. Figures 5.19 and 5.20 show the output for each phase. Two types of nucleation occur, both homogeneous and heterogeneous. As we can observe in both figures, the size of the heterogeneous nucleus in each phase is bigger than the size of the homogeneous nucleii. In the amorphous phase, we can see in addition to one heterogeneous nucleus, 5 similar sized homogeneous nucleii (only one is shown in Figure 5.19). Similarly, in Figure 5.20, we can see the appearance of two homogeneous nucleii of the same size (only one shown) and one larger heterogeneous nucleus.

For purposes of simplicity, we choose to depict the outcome from the simulator with just one number that quantitatively represents the qualitative outcome of the best result. An amorphous solid is given a number around 100. The final difflim from the simulator is between 1.2 and 7.0 for a crystal, with difflims above 3.0 representing a bad crystal.

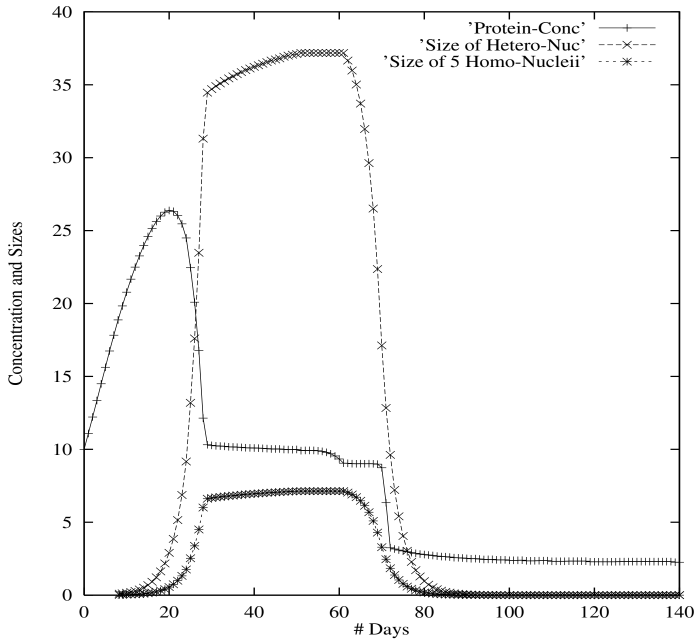


Figure 5.19: Appearance of different types of nucleii in amorphous phase

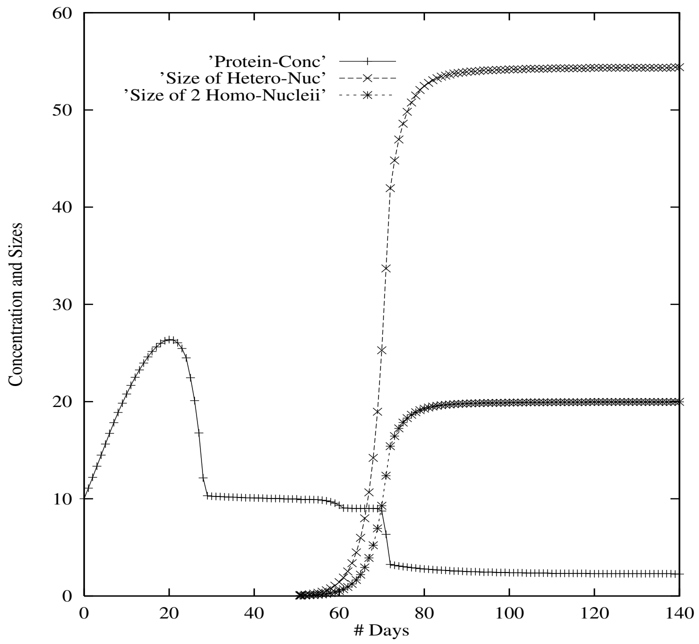


Figure 5.20: Appearance of different types of nucleii in crystalline phase

5.7.2 Effects with Varying the Critical Mass Required for Nucleation

The above experiments were performed with very small critical mass sizes required for nucleation to occur - these ranged from 10^{-6} to 10^{-7} illuminated volume. Now, if we changed the critical mass sizes to range from 10^{-6} to 1.1 depending on the level of saturation, we can see how the output changes as shown in Figure 5.21. At nucleation, the size of the nucleus is equal to the critical mass. Nucleation immediately impacts the concentration of protein in solution. We can see the drop in protein concentration at the same time nucleation occurs and protein starts moving from solution to the solid phase. This is typically the type of phenomenon that is visible at nucleation. The experiments described earlier had very small nucleii. I report only one output graph from the several experiments. This particular output is interesting as no amorphous solid phase nucleated, only the crystalline phase. The concentration of protein used was very low 1.5 mg/ml.

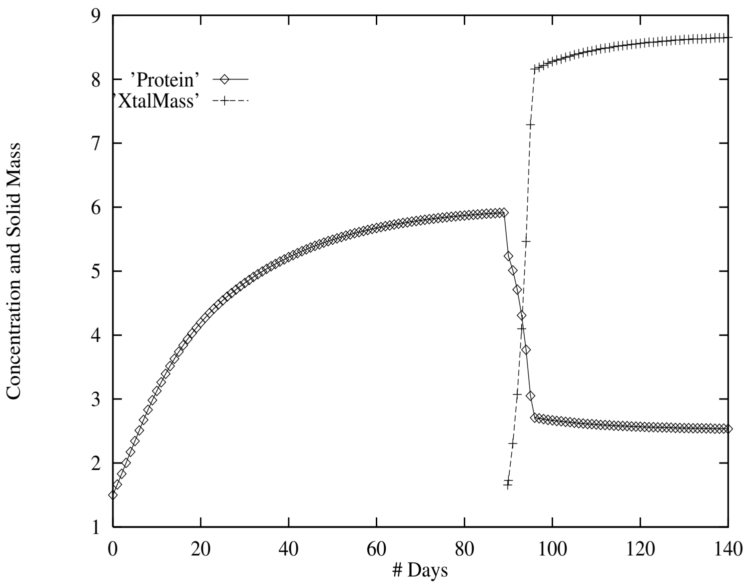


Figure 5.21: Nucleation and corresponding drop in protein concentration

Figure 5.22 and 5.23 show the change in the hidden variables, namely solubility and saturation of the crystalline solid phase of the given protein. We can see that these two parameters are inversely dependent. At nucleation, there is a sharp drop in concentration of protein in solution. This directly affects the saturation (see equation 5.3) and accounts for the drop in saturation seen in Figure 5.23.

I now summarize the sensitivity of the main factors that impact the outputs from the simulator. Sensitivity is simply indicated as *High* or *Low*, and if the level of sensitivity is in between, it is indicated as *Partial*. The impact of each factor is explained in the last column of Table 5.1. Sensitivity refers to the sensitivity of model output with respect to the particular factor. We can see that none of the factors mentioned in Table 5.1 have low sensitivity, since each one affects

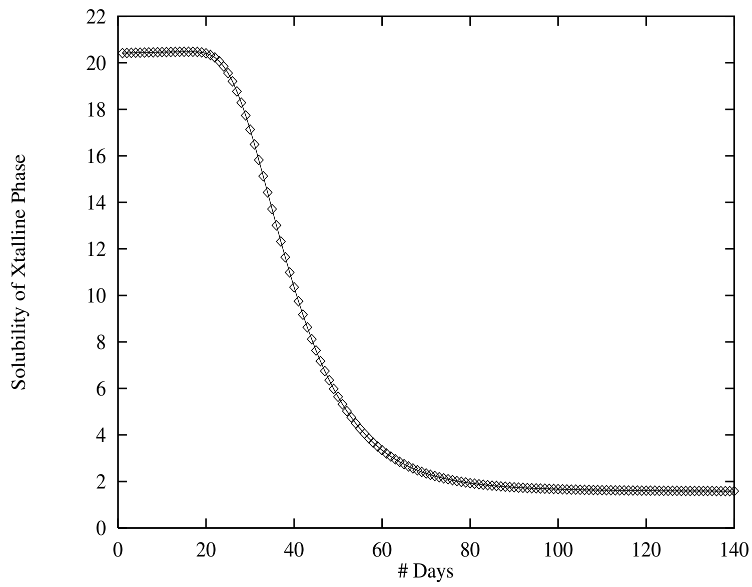


Figure 5.22: Solubility of the solid phases over time

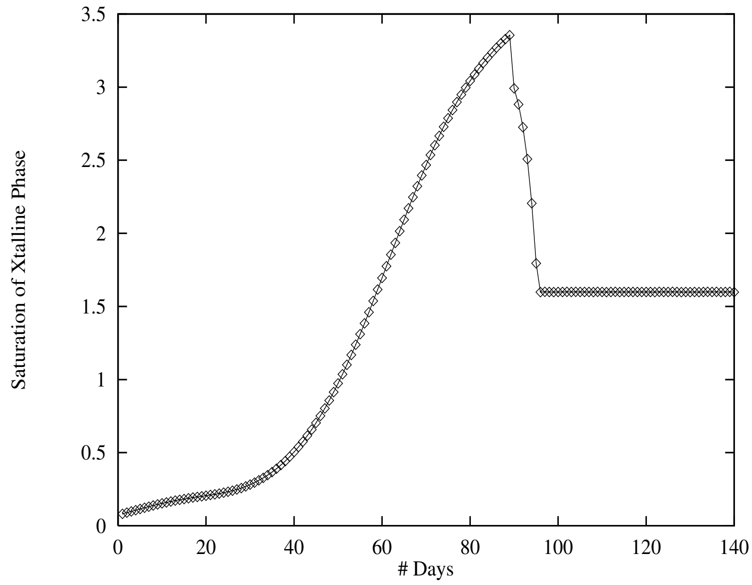


Figure 5.23: Saturation of the solid phases over time

crystallization to some extent. Those factors that seem to yield highly visible effects are labeled as possessing high sensitivity with respect to the model. Those labeled as partial indicate effects that are sensitive only to certain variations.

Factor	**Sensitivity of Model Output**	**Impacts**
PEG Concentration	High	Directly impacts overall solubility of each solid phase
Protein Concentration	High	Directly impacts saturation of each solid phase
Salt Concentration	Partial	Only at low salt concentration the salting-in effect visible solubility of solid phase is very sensitive to pH at low ionic strength
pH	Partial	Static effect only the salting-in effect is based on the isoelectric point (pI) of protein
Growth Rate of nucleus	Partial-High	Directly impacts the Difflim Sensitive to saturation Higher growth rates result in bad resolution limit of diffraction We use low numbers such as 10^{-2} Effect is invisible depending on the definition of the tables for growth rate calculations.
Number of heterogeneous nucleii	High	Number and and size of nucleii in phase
Water Volume in drop	High	Everything basically Also affects the number of homogeneous nucleii that appear as a shower of microcrystals
Critical Mass for nucleating solid phase	High	Affects the probability of nucleation and the size of nucleii at time of nucleation

Table 5.1: Summary of sensitivity analysis of model

The current model that simulates crystallization behavior of hypothetical macromolecules requires as input 16 tables that describe the various dependencies for each phase. Thus, for two phases, we require at least 32 tables describing solubility and other dependencies of given protein with respect to 4 commonly used salts and 3 commonly used precipitants. Apart from the table definitions, the simulator requires 6 parameter definitions for describing the phase information for each solid phase of the protein. Thus, we need 12 parameters to describe 2 solid phases, namely amorphous and crystalline. Additional inputs include control parameters and other givens. Each table requires approximately 4 real numbers for defining the parameters of the corresponding mathematical function. Thus, the total number of inputs to the simulator is approximately 150 real numbers.

Given the complexity of the model, it is possible to notice the effects of critical variables such as protein concentration that influence the overall solubility and saturation of a solid phase of the given protein, while effects of subtle factors that influence crystallization such as growth-rate may not be noticeable except under certain conditions. Thus, it becomes necessary to be able to develop automated methods that can perform the level of sensitive analysis required by a model of this complexity. This would be part of future work.

So far, we have described the response behavior of a hypothetical protein in a single environment over time. In the next section, we will describe overall response behavior of the same protein based on its response across several different environments or experimental conditions.

5.8 Validation of the Predictive Model for Crystal Growth

The model developed in this chapter is evaluated in the following subsections. In the first subsection, the tuning of the model to generate different response behaviors of hypothetical proteins is described. The experiments in this subsection establish face validity of the model. Some anecdotal evidence is provided in the second subsection, where a commonly used policy for evaluating partial results and deciding the next step is identified and represented in the form of a binary decision tree that suggests heuristic strategies that could be translated into actions. This policy is later evaluated and verified in experiments described in Chapter 6. The final subsection discusses evaluation of the simulator and model based on five criteria namely usability, completeness/sufficiency, utility, time complexity and flexibility.

5.8.1 Tuning the Simulator to Generate Different Behaviors of Hypothetical Proteins

We define overall behavior as the response of a protein in several commonly used chemical and physical environments, as described by the observed outcomes over time. Our tuning of the simulator model is then an engineering effort attempting to produce different types of results under different sets of conditions. Toward this end, we tried to categorize different proteins according to the flavor of their behavior. Examples of flavors include proteins that crystallize very easily (e.g., lysozyme), proteins that precipitate easily but have a very small cross-section of conditions where a good quality crystal can be obtained; proteins that do not precipitate easily, but crystallize easily yielding several crystals of bad quality and only a few well-diffracting crystals; proteins that have a very slow growth-rate (i.e., where a crystal is obtained only after 6 months or so), and so on.

In order to generate the different behaviors, it is important to understand how changing any quantity in the model affects the observed output. This could take an enormous amount of time due to the complexity of the model. Hence, we needed a quick and dirty way to generate different hypothetical protein behaviors. To this end, we devised a fixed set of conditions and then manipulated a few of the input and solubility table parameters to achieve our objective. Below, I will describe each tuning experiment with the appropriate changes made to the model in order to achieve the desired flavor of behavior.

5.8.1.1 Experiment I

Objective: To produce a model that simulates the flavor of lysozyme, a protein that crystallizes easily under a vast range of conditions. This protein was termed *Alpha Globulase*.

Method: This was our main objective when we started building the model for simulating macromolecular crystallization. Most of the knowledge engineering effort to build a fairly flexible and robust model of macromolecular crystal nucleation and growth took over a year and a half to achieve. The model is described in detail in previous sections and the domain knowledge used shown in Appendices B and C. The only way to really provide an estimate of the error that exists in the model (if used for prediction), is to actually set up real experiments in the laboratory and observe the results over time, and compare them to the results obtained from the approximate model. This kind of analysis would be extremely costly, both in terms of time and effort, and would only be necessary if we decide to use the model for prediction. In that case, we might also need to incorporate a few more features into the existing model as might be deemed necessary by the domain experts. Hence, we decided to use this model only to provide us with flavors of different macromolecules in terms of their response under different experimental conditions, and use that information to glean an insight into global strategies for experiment design.

5.8.1.2 Experiment II

Objective: To produce a model that simulates the flavor of a a protein termed a *Diabolase* for which only a few conditions in the vast space yield good quality crystals. There are a wider range of conditions wherein crystals form, but are of poor X-ray diffractible quality.

Method: After having created in Experiment I, a model that simulated lysozyme type behavior, we needed to see what type of changes to that model could create other kinds of crystallization behavior. By systematically studying the change of each input variable on the output, I learned that the representation of our model was flexible enough that we could produce different behaviors fairly easily. There were a few main places where changes in the numbers made a difference on observed output. The first was in the representation of the various solid phases of a protein and the conditions, such as the saturation required for nucleation or growth, under which they are likely to surface during experimentation (see Appendix B for definitions of parameters that describe solid phases of protein). The second was in the representation of the solubility table parameters for each independent variable that affects crystallization of that protein. We assume that our other choices for modeling the dependence of the outcome variable on the input and intermediate variables are sufficient for our purposes.

5.8.1.3 Experiment III

Objective: To produce a model that simulates the flavor of a protein with slow growth-rate, that is nothing will be visible for any experiment with some levels of protein concentration for about 6 months of time.

Method: By simply studying the model, it becomes apparent that the key parameters to vary are the exponents, τ_1 and τ_2 of equation 5.1, as they directly affect the rate at which things move toward equilibrium. Hence, we were able to produce response behavior that accurately modeled this desired objective. To give an example, we were able to produce the following type of response for approximately 4000 experiments that represented a factorial search of the parameter space (with medium granularity): For lower levels of protein concentration below 15 mg/ml, there were no observable results (clear result) for over 6 months, while if we were to increase the protein concentration, then we are likely to see a spread of crystals appearing sometime after 5 weeks or so.

5.8.1.4 Experiment IV

Objective: To produce a model that simulates the flavor of a protein with several different types of crystalline phases, each with different morphology.

Method: The model developed is flexible enough to incorporate different numbers of solid phases that can potentially form from solution. All we need to do is to specify the given phase information accordingly. We need to state the number of phases and include solubility table information for each phase. During development of the final model, this was actually tested empirically. For the purposes of this dissertation, we chose to represent and work with only two phases namely, amorphous and crystalline. With addition of each new phase, it is necessary to include domain knowledge in the form of additional solubility tables to describe the functional dependencies.

5.8.2 Anecdotes and Identification of Policy or Heuristics

We have relied on subjective evaluation of the model due to the unavailability of data. One interesting tale regarding the validation of the model comes from my direct experience.

The simulator model and the PEP system had been developed and implemented. I was playing around with the various simulation parameters to learn about their impact on the final result. It was only after the model was constructed that I began asking people in the laboratory how they decided what to do next, based on the partial results obtained from an initial screen. I have summarized what I heard repeatedly in Figure 5.24. I went back to the PEP system and tried the heuristic strategies suggested in the binary decision tree of Figure 5.24. Interestingly enough, the response from the simulator model reflected the fact that the actions suggested by the decision tree indeed lead to a favorable result. This told me two things:

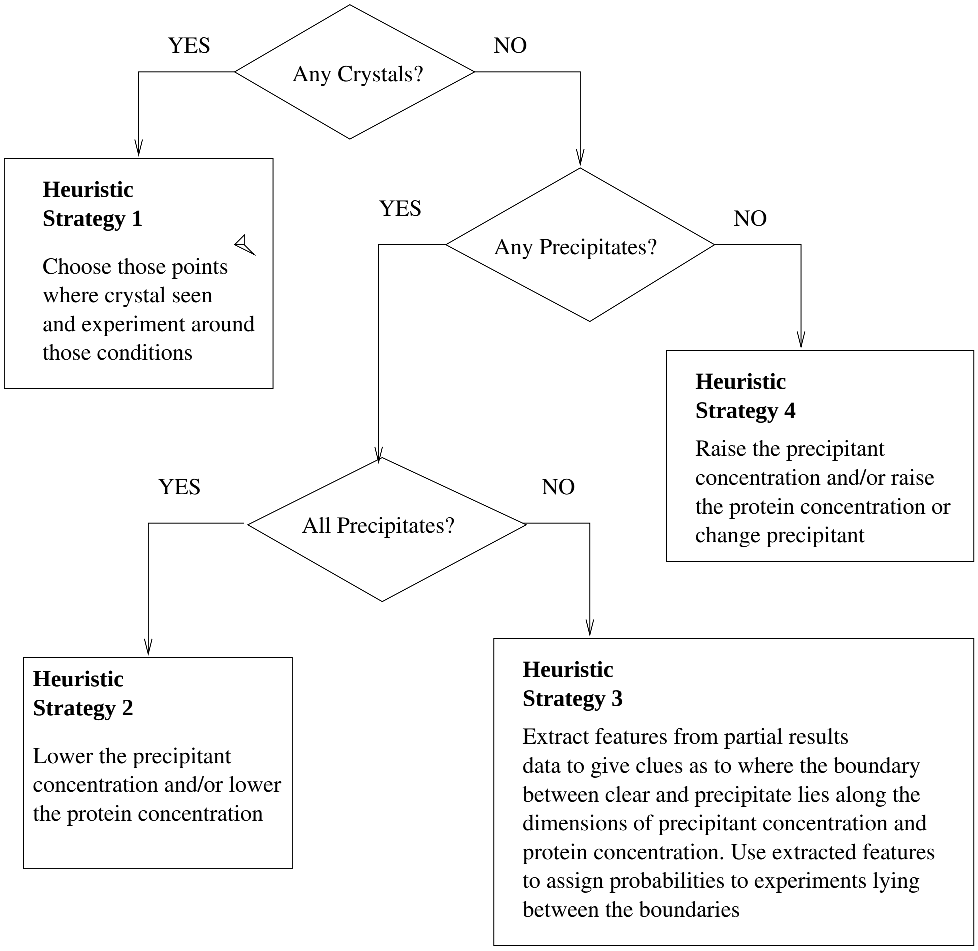
1. since the model was constructed through discussions with someone (John M. Rosenberg) other than the people who actually performed the experiments in the laboratory, the response validated both the model as well as the heuristic policy represented in the decision tree of Figure 5.24. The strategies identified in this figure were based on consultation with several crystallographers in the laboratory (other than John Rosenberg). This was the only commonly used policy in this domain for acting based on partial results evaluation. Heuristic Strategy 3 involves identification of boundary between experimental conditions responsible for precipitation versus those that were clear or unsuccessful so far. The PEP framework includes a generalization program that can present these boundary conditions in the form of human-understandable rules. Using these rules, the experiment designer can choose to experiment with a next set of conditions that are more likely to be successful.

2. since the search space is so vast in this domain, the resources scarce and the work is tedious, people were being consistent with respect to what were their available options in light of partially observable results.

Another anecdote that appears to validate a piece of the model appears in [44]. Almost regarded as a father of crystallization, Alexander McPherson says in his most recent textbook:

> For many years it was conventional wisdom that the optimal pH target for crystallization of a protein should be its isoelectric point, its pI. This was intuitively appealing because as its pI, a macromolecule carries an equal number of positive and negative charges, and is, therefore, electrostatically neutral and balanced...
>
> Eventually, however, an analysis was conducted based on an early version of the NIST/CARB BMC Data Base (Gilliland 1988). The outcome was reported but not published, and it revealed that no correlation between the pI of crystalline proteins and the pH at which they crystallized existed. Unfortunately, the pI appears inclined toward formation of amorphous precipitate, the kinetically favored solid state, over the crystalline state.

The following paragraph is based on discussions with Prof. Rosenberg – What is referred to in the above quote as conventional wisdom refers to an earlier time. Current conventional wisdom has discerned reasons why the pH at which a crystal forms is different from the pI. In the early days, proteins were thought to be spherical objects with ions on the surface of the molecule. Thus all the ionic effects were attributed to the global charge on the surface of the protein. This view has changed dramatically as time has proven the variety of different shapes and structures of proteins.



Points in the parameter space, that is, experiments

Figure 5.24: Binary decision tree for experiment design

Studies have now found that the pH effects on crystallization can be attributed to the localized ions in a protein structure. It does not take very much to suppress the global ionic effect, which is reflected as a protein's pI. Thus, it is that the pH at which a protein actually crystallizes tends to vary from the pI.

I saw this piece of information after we had built our model. I was happy to note that our model was flexible enough to be able to reflect both conventional wisdom as well as the outcome of the NIST analysis. We found that we could elicit the type of response behavior from our models, where the protein did crystallize at pH's other than around its isolectric point or pI. At lower protein concentrations and low salt concentrations, we were able to see that a pH around the protein pI was responsible for early nucleation. On the other hand, at lower protein concentrations and higher salt concentrations, we found that a pH closer to the pI was more likely to yield a result (either precipitate or crystal) sooner, while a pH further away from the pI eventually yielded a good quality crystal, but later in time.

Table 5.2 below shows a summary of the desirable effects from the model. A tick mark ($\surd$) indicates features that exist in our model. The other features are possibly required in the future.

	Effect/Feature	**Exists**
1	Stochastic nucleation	$\surd$
2	Growth	$\surd$
3	Termination of Growth	NO
4	Low salt concentration pH effects	$\surd$
5	Dynamic change of pH and its effects	NO
6	Effects related to temperature	NO
7	Ability for different phases to exist simultaneously	$\surd$
8	Ability to incorporate solubility table information	Needs Work
9	Resolution limit best at nucleation	$\surd$
10	Final crystal quality function of difflim and size	$\surd$
11	Rate of change of water volume in the drop affects the entire process	$\surd$

Table 5.2: Summary of desired effects in model

5.8.3 Evaluation of the Simulator Model

The model developed in this chapter contains representations of domain knowledge that is responsible for describing overall protein response in different physico-chemical environments. It is used to simulate experiments and output observables by taking as input the values for the control parameters of an experiment and the time for which the simulation is to be run. Any experiment can be simulated in this manner, by simply changing the definition of values for the control parameters for an experiment. This model is evaluated using 5 criteria namely, usability, completeness or sufficiency, utility, time complexity and flexibility.

5.8.3.1 Usability

The experiment simulator has three sets of inputs as described below:

1. Protein-specific domain knowledge in the form of solubility or other two variable dependencies expressed as mathematical functions, as well as the number of solid phases and a description of saturation and other conditions under which they could possibly nucleate,

2. A set of givens and controllables for any experiment. The givens represent information about a protein that does not change and is known to the user, such as name and isoelectric point; the controllables refer to the values for the control parameters of any experiment, that is precipitant and its concentration, salts and their concentration, protein concentration and pH. A different experiment can be simulated easily by simply changing the values for the controllables that are input to the simulator.

3. The time for which the simulation is to be run. This is used to simulate the manner in which observations are performed at discrete time intervals by the agent that sets up the experiment.

It is easy to express the domain knowledge inputs to the simulator as one simply needs to fill in numbers for the appropriate tables that describe protein solubility in well-known reagents. Other inputs that describe dependencies among hidden variables in the model are represented in the same manner. It would take a few hours to teach someone who had a fairly good background in macromolecular crystallization the basis for the model that is the types of solubility table parameter values that need to be input from known solubility information.

During the development stages the simulator was used to produce different types of outputs. The kind of outputs used for evaluating the model were graphical and showed the changes in input, hidden and output variables over time. Examples of these are shown in section 5.7. Another type of output used for evaluation was the description of the type and sizes of nucleii that were obtained after running the simulation for a particular period of time. This output showed the time each solid phase nucleates (if it does), the number and types of nuclei (homogeneous or heterogeneous), and the final size of each nucleus. Homogeneous nucleation often occurs as a shower of tiny microcrystals. Thus, the number of nucleii formed is represented as part of the information recorded for homogeneous nucleation. The graphical output from the simulator makes its easy to debug the model and also allows subjective evaluation.

5.8.3.2 Completeness/Sufficiency

The simulator has been evaluated in a subjective fashion to produce effects that were satisfactory to the human expert. For the purposes of demonstrating the sufficiency of the PEP framework, the simulation model developed in this chapter is able to produce the kinds of effects that human crystallographers are used to observing in the laboratory. These include the following:

- The effect of the resolution limit of diffraction that is a reflection of crystal quality slowly increasing, that is degrading, as time goes by is captured in and is evident from the model. This would give novice crystallographers the idea that timely action is required in order to pick up a crystal and obtain its diffraction pattern.
- The effect of precipitating agent and protein concentration as two influential factors that affect nucleation and growth of the solid phases of a macromolecule in solution is clearly achieved.
- The overall response behavior of proteins based on the cross-section of the search space that is typically searched that yields good crystals is used to generate different hypothetical proteins. That is, from the experiments described in section 5.8.1 we could generate hypothetical proteins that respond differently: some can precipitate easily, some can crystallize easily, and some have narrow ranges of control value combinations at which good crystals appear.
- Effects that involve different definitions of solubility of protein in specific reagents are easily produced by changing the numbers in the solubility tables that represent values for the parameters of the underlying mathematical functions.

There are several weaknesses to the current model that could be overcome in future revisions:

- The representation of solubility table information assumes a functional form such as an error function or Gaussian. A more general representation would be to use only points and interpolate the functional form from the points.
- The assumption that overall solubility is calculated as shown in equation 5.2. Although this assumption has produced the effects that the domain expert would like to see, it has not been tested or varied.
- The model is basically an initial attempt to produce the types of effects noticed in the laboratory. It does not include any effects that reflect impact from dynamic changing of pH or temperature. Including such effects might involve more complex representations and processing. The simulator is however, fairly informative even without these effects.
- This model has simplified the macromolecular crystallization process to a large extent. It is heartening to note that such over-simplification can produce many of the effects observed in

real-world experimentation. At the same time, several choices made at the time of development cannot be justified from a rigorous physical chemistry point of view.

5.8.3.3 Utility

The experiment simulator is useful in two different contexts - (1) for tutoring and (2) as an evaluation function within the PEP framework. As we will see in experiments with human subjects described in Chapter 6, the simulator can act as a black box, and by simply trying out different conditions it is possible to learn heuristics that depict some of the effects produced by the model. Thus, it is possible to use the simulator within a tutoring environment, where novice crystallographers could learn aspects of the macromolecular crystallization process without actual experimentation in the laboratory. Some of the learning includes the types of commonly used reagents, the effects of varying the concentrations of the control variables, identification of the more influential variables and the role of time.

The experiment simulator is necessary within the PEP framework developed in Chapter 4 in order to provide intelligent assistance for decision-making. The simulator model approximates the protein response behavior over a search space of reduced dimensionality. If we were to present actual solubility information for a given protein, this approximate model would be useful in predicting its response in the space of commonly used experimental conditions specified within the model.

The experiment simulator acts as the evaluation function that is based on an approximation of the real-world phenomena, and predicts experimental outcomes over time. This evaluation function is imprecise in the real-world, since the observation is performed visually, by examination of a well under the microscope. We model this imprecise nature of observation of partial result, by grouping the quantitative output of the experiment simulator into one of three visual groups. Thus, the simulator is currently used as a crude evaluation function that can produce the types of observed effects that one might see in the laboratory. Using the simulator as the evaluation function, it is then possible for the human agent in the PEP framework to try different strategies that could get him/her to the goal given certain resource constraints. The feasibility of the PEP framework using the simulator is discussed in Chapter 6.

5.8.3.4 Time Complexity

The actual time taken to perform any single run of the experiment simulator is directly proportional to the simulated time of the run. That is, if we want to observe a result in 1 week, a single run would take approximately a second of computational time on a Digital Alpha machine. The domain knowledge consists of around 16 solubility and other table definitions for *each* solid phase of the given protein. Thus, for 2 phases, we need to define 32 tables (i.e., approximately 128 real numbers). Associated with each solubility or other table definition is a mathematical function such as the error function or Gaussian distribution that is described using approximately 4 parameters (see Appendix C).

If a given protein had 2 solid phases, the number of inputs to the simulator for each run (given all tables are pre-defined) is 23. For each additional solid phase, we would need to define 6 more input parameter values such as the saturation required for nucleation and growth. The time complexity is linear in the number of inputs and domain knowledge table definitions. That is, the time complexity of the experiment simulator is O(k*t), where k is a constant that represents the time taken to perform a computational run of simulator for 1 week of simulated time and t is the number of weeks for which the simulation is to be run.

5.8.3.5 Flexibility

The representation that is used to describe the solubility data is compact, and assumes that all solubility data for particular types of reagents follow a particular mathematical function. Thus, we could model all solubility data and other hidden variable dependencies such as the B-factor, probabilities of nucleation, growth-rate and critical mass required for nucleation with respect to saturation levels using mathematical functions that define the interactions of two variables. This representation was found to be fairly easily to encode and modify.

The representation of protein-specific information such as the number of solid phases and each phase information is parameterized input to the model that simulates the protein behavior.

The representation of experiments is very simple, and hence, we only need to change the values of the controllables that are input to the simulator in order simulate a new experiment. Also, any number of salts could be input. In actual crystallization experiments, two salts are used fairly commonly. The flexibility of the representations within the simulator model offer considerable power with respect to being able to simulate sufficiently different types of overall response behavior (that changes over time) that resemble different hypothetical proteins, given a definition of the overall search space or the space of experiments that are commonly searched. In section 5.8.1, we discuss experiments that produce 4 different classes of hypothetical protein response behavior simply by modifying certain key parameters within the model.

Chapter 6

Experiments and Analysis

The goal of the experiments and discussions in this chapter is to provide support for the central thesis, which is to show the *sufficiency* of the PEP framework developed in Chapter 4 for parallel experiment planning. Sufficiency of this framework is demonstrated by constructing a working program called the PEP system that allows such an evaluation. For purposes of discussion, we will view the framework as consisting of an environment and an agent. The entire framework then supports parallel experiment planning. The design of the environment is based on its utility with respect to the agent that is included in the framework. In this dissertation, the environment serves as a tool using which a human agent could explore and learn by observing effects of actions. It is immediately obvious that the nature of the feedback from the environment affects the kinds of reasoning tasks that the human agent would need to perform.

The weak claim states that the PEP system provides the information management for parallel experiment planning. This claim is supported by experiments with me and 2 other human subjects described in subsequent sections. The strong claim states that the PEP system acts as an intelligent assistant for decision-making. I present arguments that show the value of using the simulator and generalization program to identify the boundaries between successful and unsuccessful experiments as an intelligent aid for deciding the experiments that form a new trial. In order to provide intelligent assistance, it is necessary to manage the information about parallel experiments.

The sufficiency of the PEP framework has been shown by constructing an experiment simulator as described in Chapter 5, and embedding it within the model that represents the parallel experiment planning environment. This simulated environment interacts with a human agent, the experiment designer, by asking questions. The human agent interacts with the environment by answering the questions and typing in his/her choices for parallel actions. The environment simulates partial results at user-defined observation times and provides the user with experiment summaries, parallel action or trial summaries, and a summary of the state of the world or global summary after each observation action is performed by the user.

The reason that this implementation of the PEP framework works is because:
(a) the main components have been identified and represented adequately,
(b) the instruments for manipulation and interpretation include computational procedures and book-keeping as well as a human designer,
(c) the environment is controlled and hence, errors such as incorrect recording of observed outcomes do not occur,
(d) the interaction between environment and agent is facilitated smoothly and easily in a question-answer fashion that most humans are used to and seem to like, and
(e) even though the experiment simulator described in Chapter 5 is fairly sophisticated, it is possible to find a solution if one exists within a reasonable amount of time expended by a human-agent – the average time taken to learn to use and find one hit for a medium-difficulty simulated protein is about 1.5 hours of real time.

The PEP framework allows for the representation of any number of parallel experiments, that can be represented by data structures that describe:

1. the number of experiments N to be performed in parallel,
2. a set of constant values for all the experiments, that represents the *givens* of a protein or problem.
3. a set of values for control parameters of each experiment, that vary across the N parallel experiments,
4. a search type that represents a program or algorithm used for deciding how to vary the control values across the N experiments,
5. a variable cost function that depends on N that represents the difficulty associated with performing large numbers of experiments in the laboratory, and
6. constraints on material resources, such as amount of protein available. Future resources can also be represented if necessary. Thus, the constraints on resources could change dynamically.

The simulation model described in Chapter 5 is used to generate the partial result observed for each individual experiment. The PEP system with the experiment simulator has been specifically tuned to reflect very closely the way experimentation is actually carried out in the laboratory (that is, the types of commonly used values for many of the major variables in an experiment). A simple evaluation function can convert the quantitative outcome of the simulated partial result into one of three commonly observed classes of results, namely clear (or no result), precipitate, or crystal.

Summaries are provided at all levels of detail, starting with the partial results of an experiment over the observations so far. The best result observed for any experiment and its observation time are recorded and updated after each new observation. Cost of material, effort and time to observe each experiment is also recorded and maintained. The amount of protein or material that an experiment consumes is recorded as part of its summary. Experiment summaries also contain information about the controllables used for setting up the experiment. These summaries were found to be extremely useful by the human subjects (see section 6.3.4) as they represent snapshots of experimental data over time.

Trial summaries are useful in keeping track of the percentages of results belonging to each of the observed classes. The trial history helps the designer decide whether it might be worthwhile to close a trial for observation, especially if it seems rather unlikely that a crystal might appear any time soon in any of the conditions that were attempted by that set of parallel experiments. One reason to close a trial might be because it has not yielded any results over several observations. A more intelligent reason to close a trial would be based on the description of the evolving boundary of conditions that partition the space of experiments as being successful or unsuccessful based on partial results. Thus, if a trial includes a set of experiments that were far below the boundary that describes a clear result, it might suggest conditions that are too undersaturated. On the other hand, if a trial consisted of experiments far into the supersaturated region, indicated by precipitation, it is unlikely that a good quality crystal could be obtained due to the fast rate of growth.

The global summary is an essential part of the framework, and keeps track of the global information (variables, data structures, files) after each action/decision is made by the designer. The action is by nature parallel, as it results in either observation of several experiments, or the design of several new experiments followed by an observation action. One important aspect of this summary is reflected in the file containing the results of the machine learning program. This file contains rules that are learned from training data that are accumulated as each decision or parallel action is carried out. The rules are human-understandable and are extremely useful in depicting the boundary between the various classes of observed results in terms of the experimental control values and time of observation. This helps the designer understand the values of the most influential variables that are responsible for the difference between the observed outcomes.

The most important aspect of the PEP framework is the flexibility with respect to being able to utilize different policies for acting given an intelligent analysis of all partial results observed so far. Thus, it is possible to employ different tradeoffs at any step in the decision path, and observe the outcome of such an action (parallel action) and be able to adjust the weight of possibly performing such an action from the given state. In essence, this PEP framework can be used to effectively learn policy (or policies) that essentially enables the agent (or designer) to decide on the action that is more likely to take him/her closer to the goal. Interestingly, the machine learning component plays a big role in enabling the agent to form a representation of this policy by generating rules that help discriminate effects of actions.

6.1 Assumptions

1. The local policy for choosing next moves changes according to the partial results seen and past experience of the experimenter. It is difficult to automate entirely the analysis of partial results, and hence the need for a human in the loop. The main assumption behind this inclusion is that humans tend to employ different tradeoffs at different stages in the decision making process, which are difficult to enumerate and encode due to the enormous branching factor of related actions and the tacit knowledge of past experience.

2. The number of parameters chosen for inclusion into the model for simulating the macromolecular crystal growth process is a fraction of the actual number of influencing parameters. We assume that the model can be incrementally scaled up to include additional factors influencing growth of crystals. We also assume that the existing model is sufficient for purposes of demonstration of the main theses.

6.2 Demonstration of Sufficiency using Prototype

The inputs to the prototype implementation of the PEP framework include resource constraints - mainly protein at this time, and choice of cost functions, as well as the underlying protein response model that is input to the experiment simulator. The worst case inputs would be the following:

- very small amount of protein material, with no future sources
- a hard time deadline for obtaining result
- manual cost function, where costs increase exponentially after say 100 experiments
- a Diabolase-like protein with very small cross-section

A best case scenario would be:

- large amount of protein material, with possible future sources
- no hard time deadline for obtaining result
- robot cost function, where costs increase linearly in number of experiments
- a Globulase-like protein that crystallizes easily under many conditions

For purposes of demonstrating the sufficiency of the framework for PEP in the domain of macromolecular crystallization, I created two different types of hypothetical proteins. One is a medium *Diabolase* that has a fairly narrow cross-section at which it crystallizes, and the other is a *Precipo Globulase* that precipitates very easily, but also crystallizes at a fairly broad range of values. A Globulase that crystallized very easily would not be much of a challenge with respect to learning strategy.

6.2.1 Protein *Diabolase*

In order to create a protein that produced hits only over a small cross-section, I varied the exponentials in Equation 5.1, to ensure that the whole process proceeded very slowly. This is done simply by multiplying both τ_1 and τ_2 by a factor. The other givens that are constant for all experiments tried for this protein are described in Table 6.1 below -

Name	Value
Protein Name	Diabolase
Protein pI	8.0
Number of phases	2
Phase I Name	Amorphous
Nucleation saturation for Phase I	21.3
Growth saturation for Phase I	21.2
Maximum rate of growth (Phase I)	0.06
Number of heterogeneous nucleii (Phase I)	0
B-factor (Phase I)	25.0
Phase II Name	Crystalline
Nucleation saturation for Phase II	11.5
Growth saturation for Phase II	9.7
Maximum rate of growth (Phase II)	0.05
Number of heterogeneous nucleii (Phase II)	1
B-factor (Phase II)	12.0
Number of salts	1

Table 6.1: Annotated example of *givens* for a hypothetical protein Diabolase

The solubility tables were modified from those in Appendix C, to enable similar behavior in 2 or more salts by using the same values for the table definitions. The effect of protein being more soluble in some precipitants versus others was produced by decreasing the numbers of the mid-point value of the x axis for the solubility table definitions of some precipitants while raising the numbers for other precipitants in each of the solid phases.

6.2.2 Protein *Precipo Globulase*

This protein precipitates easily, but crystals can be obtained only at fairly high concentration of protein as well as precipitant. I was able to obtain this type of response behavior by making two main changes to the parameter values of the givens in a *Diabolase* type. These were: (1) a major increase in the saturation values required for nucleation and growth of the crystalline phase, and (2) a change in the number of heterogeneous nucleii available for each phase (for instance, 0 heterogeneous nucleii in the crystalline phase would mean that only homogeneous nucleation could occur, and hence lower the overall probability of nucleation of crystalline phase).

Let us take another look at Figure 5.7 that shows the manner in which two important dimensions, namely protein concentration and precipitant concentration, affect nucleation and growth of a protein crystal. We can see that a sufficient level of concentration of both protein and precipitant is required for nucleation of the crystalline phase to happen. This is reflected in the choice of givens for protein *Precipo Globulase*. The saturation levels for protein and precipitant for the crystalline phase are made much higher than that required for the amorphous phase in order to achieve the effect of being able to obtain precipitates more easily.

6.2.3 Experiments with Me in the Loop

6.2.3.1 Experiments with *Diabolase*

Several experiments were tried with different strategies. We will first discuss the results that characterize roughly the response surface of a protein with the above givens based on our model. Table 6.2 represents the summary or global context for 17280 experiments that change over time. We have saturated the space of possible experiments to a certain extent and assume that a robot would set up these experiments. We experimented with a space consisting of 4 salts, with 3 different concentrations, 3 precipitants (PEGS) and 3 different concentrations, 8 values of pH (ranging from 3.0 - 10.0) and 20 protein concentrations (1.0 - 20.0 mg/ml).

This experiment took 8 days of computer time to run, basically because each experiment is run serially, and state of experiment is not saved. Much of this time is also attributed to the fact that I looked at the partial results and analysis before performing another observation. These inefficiencies can be overcome by re-implementing the entire framework in a manner that allows for each experiment to be an independent process that starts at some time, and keeps running, and can be observed at different times, and manipulated by physical actions that can be represented computationally (for example, changing the temperature of a set of experiments). There is only one trial as we assume that the total number of experiments tried is sufficient to saturate the search space. Please note the total number of hits as described in this table are cumulative, that is, they include hits at previous observation times as well. Thus, if we wanted to know the total number of experiments that had a crystal as a result, we subtract the current number of hits from the number of hits at previous observation. This number is represented in the column labeled Actual #hits. Now, we can see why we name this protein a Diabolase. At any time, the number of experiments

Time (weeks)	*# Exps*	*Total #hits*	*Actual #hits*	*Direct Costs*	*Intangible Costs*	*Observe Costs*	*Total Costs*	*mg of Protein*
1.0	*17280*	*594*	*594*	17441.43	1728.00	2937.60	*22107.29*	*1814.06*
2.0	17280	*1686*	*1092*	17441.43	1728.00	5875.20	*25044.89*	1814.06
4.0	17280	*3462*	*1776*	17441.43	1728.00	8812.80	*27982.48*	1814.06
6.0	17280	*4958*	*1496*	17441.43	1728.00	11750.40	*30920.09*	1814.06

Table 6.2: The response recorded as global context summaries for protein Diabolase

that yield a crystal is between between 3 and 10% of the total number of experiments or the search space.

Experiment 1

This is conducted as described above. The take home heuristic from this experiment might be stated as:

Take bigger increments of protein concentration (not 1.0 μl).

Corollary to heuristic

Small amounts of protein concentration are sufficient to crystallize proteins whose saturation levels for nucleation and growth are small. Larger concentrations of protein will tend to produce precipitates or crystals with higher difflims and therefore, poorer quality.

This corollary heuristic is observed clearly from experiments with the framework, and particularly the model. If we are trying to crystallize a protein that is fairly easy to crystallize (such as lysozyme), the strategy employed can be quite different from what works for the protein described in this experiment.

Experiment 2

In this experiment, I tried a different approach to crystallizing this same protein. I more or less followed the strategies suggested in the decision tree in Figure 5.24. I conducted 3 different sets of strategies, and found that I actually learned an interesting heuristic that I might use in the future.

- In the first set, I started with 100 experiments, and then gradually added another 100 experiments with higher precipitant concentrations and then finally higher protein concentration. This strategy worked very well.

- In the second set, I started with 300 experiments which had some high precipitant concentrations. But, when they were unable to yield any crystals or precipitates, I had started another 324 with higher protein concentrations this time, and higher precipitant concentrations. These did yield crystals, but the process turns out to be very costly, both in terms of amount of protein consumed as well actual costs.

- From the above two experiments, I concluded that it might be worthwhile to simply start with higher protein and precipitant concentrations in the initial trial, and observe the effects. This does work, and consumes very little protein - I simply had only 36 experiments in the trial. It is interesting to note that different types of proteins might benefit from different types of initial screening strategies. One hopes that the research described in this dissertation would one day lead to the discovery of useful heuristics that can strengthen existing crystallization protocols.

The above experiments gave rise to the heuristic stated below:
If a protein does not precipitate easily, use a high concentration of protein and a high concentration of precipitant. Lower concentrations will probably not yield crystals for a long time.

6.2.3.2 Experiments with *Precipo Globulase*

I summarize my experiments using the PEP program - after trying a few different paths that took me to the goal, that is at least one hit, a global strategy emerged for crystallizing such proteins. Even if the protein precipitates easily, try higher concentrations of precipitant and protein. It might be the case that the crystalline phase requires higher levels of saturation in order for nucleation to happen. So, if we do not observe any crystals but heavy precipitation occurs for a large range of conditions, trying two sets of different trials might help. These are:

- Try combinations of lower concentration of precipitant with respect to protein to determine whether precipitation is occurring because of higher precipitant concentration proportional to protein.

- Try higher concentrations of both protein and precipitant to determine whether the crystalline phase possibly requires larger levels of both.

6.2.3.3 Experiments with Machine Learning Programs

The objective of trying more than one type of learning program that identifies the variables (and the values) that are most effective in distinguishing the outcome class was to ensure that the results did not vary based on the tool being used. For this purpose, I ran experiments, wherein two different types of metrics were used for solving the boundary distinction problem. The results were exactly identical. The metrics used were the information gain criterion of C4.5 (equation 2.3) and the certainty factor measure of RL (equation 2.4).

Herein, I will describe two main types of experiments performed using the C4.5 machine learning program (and RL) to induce rules that generalize the set of training examples presented to it. We can classify the two experiments as representative of two types of cases, namely, one in which all the training examples are correctly classified by the program, and another in which some examples are incorrectly classified. As we shall see, the machine learning program can yield useful boundary information in both cases.

6.2.3.4 Experiment with All Training Examples Correctly Predicted

I will describe this experiment as a series of steps that I executed using the PEP program. The outcome at each step, and the rules produced by the C4.5 program will be summarized. This experiment comprises a set of givens that represent a protein that precipitates fairly easily, and yields some crystals only at very high protein concentrations.

Step 1 I generated 60 experiments, whose control parameters are shown below. The first time of observation was after 1 week.

Control Parameter Name	Value(s)
Salt Name	Ammonium Sulfate
Salt Concentrations	0.1, 1.0
Precipitant Name	Mama PEG
Precipitant Concentrations	10.0, 20.0, 30.0 % w/v
pH	5.0, 6.0, 7.0, 8.0, 9.0
Protein Concentrations	5.0, 10.0 mg/ml

Step 2 I generated 8 new experiments, whose control parameters are shown below. The next observation was after 1 week.

Control Parameter Name	Value(s)
Salt Name	Ammonium Sulfate
Salt Concentrations	0.1, 1.0
Precipitant Name	Mama PEG
Precipitant Concentrations	40.0, 50.0 % w/v
pH	7.0
Protein Concentrations	7.0, 10.0 mg/ml

Step 3 I generated 4 more new experiments, whose control parameters are shown below. The next observation was after 1 week.

Control Parameter Name	Value(s)
Salt Name	Ammonium Sulfate
Salt Concentrations	0.1
Precipitant Name	Mama PEG
Precipitant Concentrations	40.0, 50.0 % w/v
pH	7.0
Protein Concentrations	15.0, 20.0 mg/ml

Step 4 Do not start something new, observe all experiments after 1 week.

Step 5 Do not start something new, observe all experiments after 1 week.

Step 6 Do not start something new, observe all experiments after 1 week.

The table below shows the results that describe the summary of each trial over time.

Trial#	*#Expts*	*Protein Amt (mg)*	*Observation time (weeks)*	*%class1 results*	*%class2 results*	*%class3 results*
1	60	4.5	1.0	0.0	50.0	50.0
			2.0	0.0	50.0	50.0
			3.0	0.0	80.0	20.0
			4.0	0.0	100.0	0.0
			5.0	0.0	100.0	0.0
			6.0	0.0	100.0	0.0
2	8	0.68	1.0	0.0	50.0	50.0
			2.0	0.0	100.0	0.0
			3.0	0.0	100.0	0.0
			4.0	0.0	100.0	0.0
			5.0	0.0	100.0	0.0
3	4	0.7	1.0	50.0	50.0	0.0
			2.0	50.0	50.0	0.0
			3.0	50.0	50.0	0.0
			4.0	50.0	50.0	0.0

The decision at each step in the strategy described above was taken based on the results obtained from the C4.5 program on the set of partial results observed up to that point in time. After the first week of observation of the 60 simulated crystallization experiments, it is found that 50% of the partial results were precipitates, and the other 50% were clear. When we look at the rules that describe the boundary between these two classes of partial results, we find that the lower protein concentration was not yielding any results. Hence, for the next trial, we try a new set of 8 experiments, with increased precipitant concentration and slightly higher protein concentration. This was found to yield 50% precipitates and 50% clear results after 1 week, so for round 3, we choose much higher concentrations of protein, and this is found to yield favorable results. The machine learning program can pick the most influential parameters and their values that are responsible for the division among the classes. The symbolic inductive learning programs, C4.5 and RL (described in Chapter 2) provide useful generalization from partial results due to their abilities to:

(a) represent the boundary in human-understandable form (e.g., tree, rules), and

(b) identify the holes that exist in the evolving theory of where the boundary lies.

6.2.3.5 Experiment with Some Training Examples Incorrectly Predicted

In this subsection, I will demonstrate the usefulness of incorrect prediction of training examples in the choice of the next set of experiments to perform. Incorrect prediction of any training example by the set of learned rules or model implies an incorrect labeling of the example as belonging to a particular target class, when it actually belongs to another class. These incorrectly predicted examples fall generally into two classes - the false positives and the false negatives. Buchanan, et al. [10] have labeled training examples as near misses, solid misses, near hits and solid hits. The false positive predictions represent training examples that are near misses with respect to the target theory, while the false negative predictions constitute the near hits. Below, we explain how near misses and near hits constitute effective training examples, and how their identification enables active experimentation.

Generalization from training examples is possible because sufficient number of examples exist in order to lend credibility to the proposed hypothesis, or to its lack thereof. If we evaluate the basis for which one or more training examples are incorrectly predicted by evolving theory (described maybe by a disjunction of conjunctive rules), we might attribute the error in classification to one or more of the following reasons:

1. Insufficient number of examples to allow the formation of a general rule that could cover these training examples.
2. The actual theory itself consists of exceptions to rules.
3. The set of features that describe each training example is insufficient.
4. The set of training examples do not have sufficient variability and therefore are not representative of the target concept.

If we assume that reasons 2, 3 and 4 do not apply in our domain, then we are left with one main reason (1) that is responsible for the incorrect classification of some of the training examples. This raises an important issue that involves the choice of training examples - namely, which ones and how many to include. It is pretty apparent that this is a vital issue, as the choice of the training examples directly impacts the generalizability of an evolving theory. Now, if we look at the process of generation of training examples, we could see the parallel with experiment design. Here, the choice of the next set of experiments to perform is equivalent to deciding what training examples to choose. The outcome of each experiment chosen is crucial to learning, and thereby determines the next course of action (i.e., choice of next set of experiments to perform).

Buchanan et al. [10] classify examples as *near hits* (positive examples just inside the boundary of the target theory), *solid hits* (positive examples farthest from the boundary of the

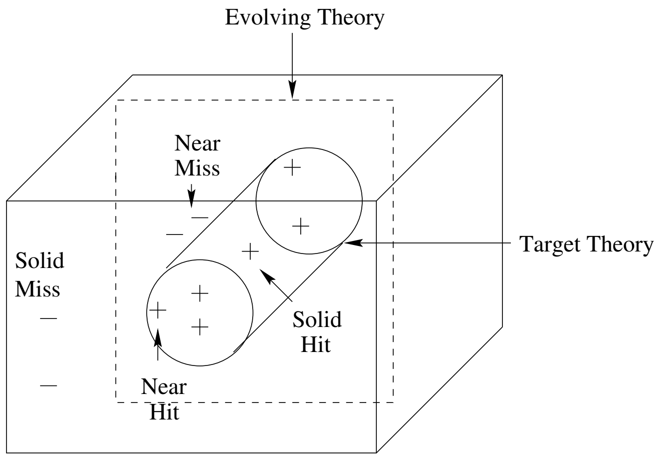


Figure 6.1: Generalization from training examples

target theory), *near misses* (negative examples just outside the boundary of the target theory) and *solid misses* (negative examples farthest from the boundary of the target theory). The near hits would be the positive examples with most power with respect to generalization, that is the the boundary definition that can predict that near hits belong to the target class would also be able to predict that the solid hits belong to the target class. Negative examples are the key to convergence on a target theory in top-down learning systems such as RL. Since we start with the most general theory and slowly shrink it to achieve the target theory, the evolving theory is always shrinking and therefore, includes the target theory as a subset. Positive examples keep the system from shrinking too far, while the negative examples determine the rate at which the shrinking occurs toward the target theory. Thus, near misses would be the negative examples with the maximum shrinking power. Figure 6.1 shows the evolving boundary, target theory and identifies the different types of training examples.

Three important conclusions have been noted by Buchanan et al. in their paper. I will restate them below, and attempt to explain/verify them using experiments in this domain.

1. Incremental learning will converge with fewer training examples than batch learning as the selection of the new training examples that are more effective is aided by the knowledge of the partially learned evolving theory that describes the evolving boundary.

2. Looking at anything but effective examples is a waste of effort when we assume that no errors are present in the descriptions and classifications of examples.

3. The extent to which an example shrinks the hypotheses space is a function of its distance from the target theory. The authors call this the efficiency of an example. Efficiency is inversely related to distance from the target theory given a suitable definition of distance, such as the root mean-squared deviation on the feature vectors.

What is interesting to note is that simply by observing the kinds of costs involved with strategies that involve one-step trials where a large number of experiments are screened in the first round, versus strategies where you perform a few (say 100) experiments in the initial round and then utilize partial result evaluations to design a new trial, it clear that the latter is more cost-effective and consumes less material resources than the former. This tells us that some knowledge about the evolving theory about observed partial results is possibly useful. This provides a basis for the first conclusion noted above by Buchanan et al. in that incremental learning is better than batch learning. Of course, the issue now is whether our representation of boundary as learned by a learner such as RL or C4.5, would be useful in intelligent example selection.

What is also important to note is that generation of newer training examples based on incorrect predictions of existing training examples is mainly useful when the designer has changed goals and is trying to learn where the boundary lies between the observed classes of results. Learning the exact boundary definition is equivalent to convergence on the target theory, and would be useful for picking the optimal conditions for crystal growth when laboratory results are reproduced. In the domain of crystallization, optimization refers to a search for those sets of conditions that give rise to optimal crystal growth. Thus, obtaining just one hit is insufficient, since we do not know a priori if that is the crystal with best possible resolution limit with respect to X-ray diffraction. Thus, crystallographers often follow certain protocols that prescribe systematic search for optimal crystal. As the data builds up, we will be able to appreciate the need for programs that can automatically learn the descriptions of the boundary between classes. We will also notice that sometimes it is not possible to correctly predict all the training examples because of insufficient data. Identifying those incorrectly predicted examples is useful since we can perform additional experiments that can help fill in the holes in the evolving theory.

The RL rule-learning program is the better of the two programs for describing the experiment. RL can read in prior rules and therefore can eliminate the need to cover training examples already covered by the evolving theory. Therefore, I will describe my experiments with intelligent example selection using the RL program. Let us consider the following example - this was a set of experiments that were generated during the final stages of evaluation of the crystal growing model. Several experiments were tried at that time with the RL program to understand the impact of intelligent selection of training examples, since at that time, we were assuming that we were going to look at very large sets of data that were representative of the response space of a hypothetical protein. Thus, the issue that we were mainly concerned with was whether a subset of the training examples would be sufficient in terms of yielding the same accuracy in terms of predictive power. Coverage of training examples would also be a factor that would need to be studied.

I generated 7939 examples that covered a fairly large part of the parameter space of possible experiments. I was able to sort these experiments easily since at each time only one variable value changed via systematic experimentation (or generation). Simply by picking those examples that lay on either side of the boundary between classes (i.e., I picked those examples in the sorted sequence where the target classes changed. In order for me to do this, I had to assume that the discrete-valued attributes (such as salt and precipitant) could be ordered in some way. Since the number of salts were small, I simply labeled them numerically. The precipitants were ordered based on molecular weight.

RL was run several times with a subset of the training data (594 cases). I tested the learned rules on 7939 examples to evaluate accuracy and coverage. The result was that with just 6 rules, I was able to obtain a coverage of 100% over the 7939 training examples with 95.8% accuracy. These were excellent results. One main point that this showed was that it is possible to use only *near misses* and *near hits* to learn the boundaries between classes. These were probably the most effective training examples and hence sufficient to enable accurate learning and fast convergence. The mixture of training examples contained 7350 examples that were classified as crystals, 422 examples as precipitates and 167 as clear (no result). The rules found were able to classify the "clear" class accurately all the time. The confusion matrix revealed clearly that the boundary between "precipitate" and "crystal" was unclear. Note that the 594 training examples contained mostly near misses and near hits. These examples were shown to be sufficient with respect to classifying with 95.8% accuracy the set of training examples (7939) that included solid hits and solid misses as well. If we used all 7939 examples as the training set, we still get prediction accuracy of 95% or less because the majority of examples were hits and thus the generalization program would classify any example that was not covered by its rules, as a hit that is belonging to the class of crystals.

I further used RL to point out the training examples that were being incorrectly predicted. It turned out that as I added/generated more experiments around the conditions that were being misclassified, I was able to generate one or more additional rules that were responsible for increasing the coverage of training examples ($594 + x$). Also, it is noticed that as we increase the training data by generating new examples, as anticipated, we notice that the incorrect predictions do not disappear, it is just that some other training examples might be predicted incorrectly - since the evolving theory still needs to converge on the target theory.

6.2.4 Experiments with 2 Different Human Subjects in the Loop

Two human subjects were used for purposes of testing the PEP system. Subject 1 is a graduate student in Physics with a some laboratory experience in macromolecular crystallization.

Subject 2 is a person who does laboratory work on a day to day basis. Both subjects were given description of the same two hypothetical proteins to crystallize (in the simulator) in two sessions each. Protein 1 was Precipo Globulase and Protein 2 was a Diabolase. The aim of the experiments were:

1. To identify whether the subjects were able to notice a difference between the two protein response behaviors while attempting to find conditions that crystallize the proteins?
2. Did they try different strategies?
3. Did they learn anything about the model and strategy? And,
4. Was the machine learning program useful?

Their answers were a definite YES to all questions. Table 6.3 describes a verbatim summary of their comments. These comments were obtained after the second session. In the first session, both subjects were presented with Protein 1, a Precipo Globulase. In the second session, they were given Protein 2, a medium Diabolase. The questions were posed in this context. This was the first time that both subjects were introduced to the PEP prototype with the simulator acting as the evaluation function. I acted as the interface between the subjects and the machine, by describing to them initial choices of actions, and typing subsequent actions in. The subjects felt that the interface was easy enough that they could do it themselves as well. Both subjects spent less than 1.5 hours with each protein to get used to the program and try a sequence of trials that got them each to the goal. This includes the time taken to simulate each experimental result. The total number of experiments tried by each subject for each protein was less than 400 with the number of observations performed being less than 5.

A summary of the final comments and quantitative scores assigned by both subjects is described in Table 6.4. All categories received a score of 9 out of 10 and an overall score of 9 out of 10 for the PEP implementation.

Some comments are necessary on the types of strategies the two subjects adopted. It was interesting to note that:

1. The first time I started the PEP program with Subject 2, I imposed a resource constraint on protein amount that was too small (10 mg). Subject 2 started by setting up experiments very parsimoniously - 10 experiments at first. When it was found that these very not likely to get successful results soon, the subject went to the other extreme and tried to set up 1000 experiments. Needless to say, the search ended since there was not enough protein to set up that many experiments. We restarted, and this time I allowed a fairly reasonable amount of protein (100 mg) each time, so that there would be enough material for the subjects to try various things.

Criterion	*Subject 1 Comments*	*Subject 2 Comments*
Different Protein?	Lower pH Results only for narrower range of conditions Took longer time at lower protein concentrations	Definitely more difficult to crystallize - quite a narrow range to produce a "good" crystal
Different Strategy? (with more protein) (with less protein)	More observations	 Experiment with all variables without worrying about using that much protein since you have that much already Precipitant concentration in mid ranges Protein concentration very low (2, 3, 4, 5 mg/ml)
Learnt anything?	Protein 2 took longer time at lower protein concentrations	"Could cut down in half what you do in the lab" "probably eliminate a lot of the pH and protein conc trials" The program helps you to not waste a a lot of time especially if you have no idea how to crystallize something in the lab as it has the same flavor of commonly used reagents Saves time and probably reduces cost of materials and time for people to do the trials in the lab.
Machine Learning program helpful?	Very helpful for finding boundaries, and which variables were most important	Definitely - simple and easy to use Helps someone learn about how to crystallize rather than just trying anything at all.

Table 6.3: Evaluation by the human subjects - comments after experiments with protein 2

2. Subject 2 was very consciously trying to reduce costs in terms of material, effort and time initially by choosing small amounts of material resources for setting up experiments. Upon observing the partial effects at time of observation t, subject 2 would deliberately raise concentrations of precipitant and then protein - almost trying a policy that is similar to the decision tree in Figure 5.24. However, once subject 2 figured that the simulated results could be obtained fairly quickly, broader screening was tried for the new protein to be crystallized. After having the experience of crystallizing protein Precipo Globulase, subject 2 actually decided to pick fairly large protein concentrations for the very first trial of protein Diabolase and managed to get a hit. It was interesting to note that the overall costs for obtaining at least 1 hit was lower for subject 2 than for subject 1.

3. Subject 1 was very conscious of the fact that there was a physical model sitting behind the PEP implementation. Also, this subject was totally happy that partial results were being

Question	*Comments*	*Score (1-10)*
Interesting?	Yes, makes life easier - Summaries are especially helpful at all levels.	9
Useful?	Very useful because it is easy to use. Easy to follow - clear and concise.	9
Interface?	Interactive - do not need a graphic - easy to understand already Rules are very helpful - limit your choices and help you get a crystal faster.	9
Overall Score?	Sufficient for PEP	9

Table 6.4: Overall evaluation of PEP prototype by the human subjects

simulated, and that experiments were not being set up in the lab. This subject decided to go about the experiment design very systematically, but always set up more experiments in the initial trial than subject 2. While subject 2 varied the initial trial (including number of experiments and the parameters of search), this subject decided to start with the same initial trial.

It is easily observed through experiments with the two different human agents that they each employ different policies and tradeoffs. The costs and time taken to achieve a single hit therefore vary. Using the PEP prototype, it is possible for a human agent to actually learn by performing simulated experiments and obtaining useful feedback from the environment part of the framework. Both subjects felt that the feedback from the environment was both useful and sufficient to enable them to decide which controls to use for setting up a new trial.

6.3 Evaluation of the PEP Framework

The parallel experiment planning framework will be evaluated using the following criteria: usability, completeness or sufficiency, utility, complexity and flexibility. Each of these will be discussed in separate subsections.

6.3.1 Usability

I demonstrate sufficiency of the PEP framework with the use of a working program (the PEP system) that implements the ideas that form the basis of the framework. I will discuss usability with respect to this implementation. In the PEP system, the agent is a human user and the environment is simulated using a computer that runs the unix operating system. The model used for the experiment simulator is not visible to the user. Thus, the user essentially plays a game. He/she is given a certain amount of protein to start with, along with a set of allowed inputs that form the definitions of experiment controls (see Appendix B) to the simulator. The environment poses question to the human agent, and the human agent replies with answers and other inputs.

The environment was liked by both human subjects who were involved in the experiments mentioned earlier. Particularly, they liked the simple nature of interaction and the ease with which they could set up several experiments in parallel, and watch the simulator come up with partial results for each experiment. Different parallel actions are performed at different start times and the environment keeps track of all this information. The statistical summaries provided were particularly welcomed. The environment keeps track of when the best result was observed for each experiment. Trial summaries were useful in pointing to the regions of the space that were yielding no results at all.

The global summary included the rules that describe a possible model of where the boundaries between the observable classes lie as well as the most influential variables according to the information-gain criterion described in Chapter 2. The human subjects were quite happy to be able to use these rules. If these rules were unavailable to them, they would have to spend a very long time trying to elicit this boundary information from raw data that reflect partial results for each and every experiment. The number of experiments that are set up in parallel in each trail varies between 50 to 200 initially. Thus, the number of data points that need to be parsed though hard for a human being, is well within the reach of a computer. The theory of evolving boundary characteristics produced by the generalization program consists of a set of rules that also include time of observation as one of the features on the left hand side. Initially, the representation of this theory is simple (small number of rules). As the data accumulates, the evolving theory could become more complex. The usefulness does not diminish, through usability might. That is, it may not be easy for a user to look at a large set of rules and then decide what to do next.

We already noted in Figure 5.24 that the boundary between clear and precipitate results is important as it tells us which experimental conditions might lead to supersaturation and which are clearly undersaturated so far. An interesting point to note is that many crystallographers do not actually use boundary information if it is hard to elicit from the raw data. Instead, they set up experiments along each point where precipitates occurred. Certainly, the boundary information would be useful in deciding which set of experiments are far into the supersaturated zone and which lie along the boundary. The experimental conditions that lie close to the boundary between clear and precipitate results may be more likely to yield successful results.

6.3.2 Completeness

The argument for completeness of the PEP framework is based on its view as best-first heuristic search with the goal being to find a good quality crystal. The search is thus complete if the space of possible experiments is finite and there are no constraints on resources. We assume that the agent is rational and searches exhaustively. For simplicity, we also assume that the agent makes a sufficient number of observations at regular intervals so that a crystal is found if it appears in an experimental apparatus. If the space of possible experiments is infinite, or if there exists constraints on resources, there would be no guarantees of completeness of the PEP search.

6.3.3 Utility

The weak and strong claims represent the utility of the PEP system. The first claim views the framework as providing the information management necessary for parallel experiment planning, while the other views the framework consisting of the simulator and generalization programs as providing intelligent assistance for decision-making.

6.3.3.1 PEP System as an Information Management System

The goal of PEP in the crystallization domain is to obtain one good crystal given some resource constraints. If we view the PEP framework as an information management system, the simulator is a placeholder for actual experiments performed in the laboratory. The agent has to make the decisions as to whether or not to setup a new trial, and when he/she would like to observe some or all experiments that are running concurrently. The agent has knowledge of everything done so far, and is assumed to make decisions based on this knowledge.

The PEP framework viewed as a manager of information about parallel experiments, resources consumed and costs incurred, provides computational assistance to the experiment designer. The experiment summaries are useful in keeping track of the partial results observed over time. They also record the best result obtained and its time of observation for each experiment. This is useful information as it provides an insight into the rate at which the crystallization process.

We have already discussed in earlier sections the advantages of maintaining information about trials and global information. Costs incurred and resources consumed are useful information computed and stored within the PEP system. This information is necessary for deciding whether to design a new trial or simply observe. The PEP system that performs information management is a passive view of the framework, where the data structures and statistical summaries correspond to items that are entered, stored and queried using database management systems. The active view of the framework as an intelligent aid is discussed below.

6.3.3.2 PEP System as an Intelligent Decision-Making Aid

The computational assistance provided by the information management components of the PEP system is necessary in order to provide intelligent computational assistance. The PEP system with the simulator and generalization program constitutes the intelligent assistance for decision-making in the crystallization domain. The simulator acts as an approximate evaluation function for a subset of the search space for which there exists some theory based on definitions of solubility parameters.

The simulator model provides intelligent assistance due to the following reasons:

1. We could produce the solubility table information that is used as input to the simulator model using small amounts of protein thus making intelligent use of a scarce and expensive resource.

2. It is possible to simulate thousands of experiments within seconds, thus we can obtain experimental data from an approximate model quickly and easily. This data can be analyzed to yield a description of the boundary between successful and unsuccessful experiments, thereby identifying a set of initial screening conditions for trying in the laboratory.

3. The model incorporates a view of the crystallization process based on years of experience in the field of macromolecular crystallization, both in theory and practice. It can be used to enable novice crystallographers develop a better understanding of the domain.

The solubility of protein in individual reagents can be determined by using just a dab of protein on the tip of needle and dissolving it in some concentration of a particular chemical reagent. The simulator of hypothetical macromolecules utilizes this idea by incorporating a model that can take solubility information as input and simulate the interactions among the various chemicals, to yield experimental outcomes over time. The output from performing several simulated experiments would indicate the regions in the vast multi-dimensional search space that are more likely to yield successful results.

We can assume that it is possible to discretize the values of the continuous-valued variables such as pH and the concentrations of salt, protein and precipitant. The space of possible experiments that is modeled by the simulator is thus finite. Since N or the number of experiments is variable within the PEP framework, we could actually set up the simulator to run every possible experiment within the finite set of experiments, as defined above.

We now discuss the merit of including a generalization program that can evaluate partial results of experiments and produce a description of the boundary relating control values to observable effects. These summaries will be useful in both cases of use of the PEP framework - with simulator and without. This is because it is impossible for an agent to function efficiently without compressing information and representing it in a simple way for future use. One way of compressing information is to form generalizations that could hold given what an agent has seen so far. The agent tries to form a hypothesis using a more compact representation than raw sensory inputs that seems to best fit the data, using a certain criteria to determine what constitutes a "best" hypothesis.

Inductive learning algorithms are able to generalize from observed training examples to classify unseen examples using a policy that is known as their inductive bias. Mitchell [46] defines inductive bias of a learner as the set of additional assumptions sufficient to justify its inductive inferences as deductive inferences. The C4.5 algorithm uses a search strategy that prefers shorter trees to longer ones, that is simpler models to more complex ones. Also, the models or trees that contain high information gain attributes closest to the root are typically preferred by the C4.5 program. This constitutes the inductive bias of the algorithm. The ID3 algorithm that forms the basis of the C4.5 program searches a complete hypothesis space (that is all possible models that could fit the data), but searches it incompletely. The search is incomplete, since it stops with the first model that is consistent with the data, and since the models or hypotheses are ordered according to its preference or the inductive bias, it will find the simplest model that fits the training examples.

For the purposes of demonstrating sufficiency of the PEP framework developed in this dissertation, I would argue that the C4.5 program is sufficient with respect to providing a human-understandable model that describes the training instances in terms of features that separate the classes of observed partial results. This argument is based on the observation that experimentation is performed more or less in a systematic manner, and each time a small number of experiments, say 100 or so, are set up. The boundary definition is learnt based on these small number of examples that grow with each observation and each new trial. Typically, the boundary definition tends to be simple initially and this is sufficient for the purposes of experimentation in this domain.

6.3.4 Time Complexity

We have 6 parameters that are controllable variables for any experiment. It takes almost 1 second on a Digital Alpha machine to run a simulation of 1 experiment with 1 set of values for controllables for 1 week of simulated time. (The various concentrations and other entities are updated within the model at five minute intervals.) Now, if we are able to parallelize the framework, then n experiments would still take only 1 second to produce a result at the end of 1 simulated week of time. Since we are still running the simulation sequentially, it takes the order of n seconds to obtain the results for n experiments at the end of one week. Also, currently, we have not implemented a mechanism by which the simulated process could save its state and resume, so if an experiment is observed after 2 weeks, we would have to start the simulator again with the time variable set to 2 weeks. This inefficiency can be overcome fairly easily and would be part of future work. The current PEP system is sufficient for purposes of illustrating the main claims. A prototypical implementation that simulates the PEP environment in a serial fashion has been used together with a human decision-making agent to demonstrate proof of the main claim of sufficiency of the framework for parallel experiment planning. I will analyze the time complexity with respect to this prototype.

Let N_o be the number of observations so far, and N_T be the number of trials so far. Let us also assume that no experiments or trials are closed. So, if n_j is the number of experiments in trial j, and there are m_i trials at observation time t_i, and t_j refers to the start time of trial j, then the computation time required for making and reporting i observations at different times could be described as:

$$c \times N_T + \sum_{i=1}^{N_o} \sum_{j=1}^{m_i} n_j \times (t_i - t_j),$$

where c is a constant and cN_T represents the time taken by the human designer in the loop.
If we assume that observations are performed once every week, with new trials possibly starting after each new set of partial results are accumulated, then we can expand the summations as follows:

$$c \times N_T + n_1 \times 1 + (n_2 \times 1 + n_1 \times 2) + (n_3 \times 1 + n_2 \times 2 + n_3 \times 3) + \ldots$$

If we assume that the time taken to input new trials is negligible compared to simulating the experiments and generating partial results, the complexity of the framework (run in serial mode) at each decision step is $O(N_e N_{oe} T_{oe})$ where N_e is the total number of experiments, N_{oe} is the number of times each experiment was observed, and T_{oe} is the time of observation (relative to start time of experiment) each time the experiment is observed. Now, if we view the framework in its natural setting, i.e. as a parallel implementation, we can evaluate complexity as follows. Let us assume that there are d decision steps. At each step, the complexity of observing a partial result is given by $O(N_e)$, which is the number of experiments that are running in parallel at that time. Therefore given d decision steps, with N_{ed} being the number of experiments at decision step d, the overall complexity of the PEP framework would be $O(dN_{ed})$, assuming some constant set-up and observation times for each experiment, assuming drudge tolerance factor is 1.

6.3.5 Flexibility

The PEP framework is flexible in several respects as discussed below.

1. Utility: There are two different contexts in which this framework could be used for parallel experiment planning as discussed in section 6.3.3.

2. Representation: The overall representation of the framework as consisting of an agent and an environment is very general. Thus, agents as well as environments could be designed appropriately to facilitate experiment planning goals and tradeoffs. The complete PEP framework described in Figure 4.1 supports complex decision-making by permitting the human agent to apply different tradeoffs at different times during the path to the goal. A human agent could even change goals within this framework. For example, let us say that the agent decides that he/she is not satisfied with one good crystal, but would like to proceed to look for a crystal

with better difflim. The agent simply continues the search, assuming that there is sufficient protein material. The flexibility for changing goals exists because the agent is in charge of defining the stopping condition for the search. This is powerful with respect to solving real-world problems. The representation also permits different agents to utilize their unique biases, to try different strategies for arriving at a solution, and provides feedback as to the utility of each step of the strategy as well as the strategy as a whole, by computing and storing costs associated with each parallel action.

3. Modularity: The PEP framework is modular and should be easily adaptable to other domains. For example, we could substitute one machine learning program for another. We could also change the simulator that exists within the framework.

6.4 Extensibility of the Framework

As discussed earlier in section 6.3.4, the parallelization of this framework would result in almost linear time for generating partial results from experiments. In section 5.8.1.4, I describe experiment IV, where different possible solid phases (needles, plates, etc) could nucleate and grow possibly at different saturation conditions. The macromolecular crystallization model could most certainly scale up to handle different types of phases, different numbers of heterogeneous nucleii, and therefore different types of response behaviors. We have not included temperature effects in the model, and thus it remains to be studied whether its inclusion to the current model will require a change in representation.

One interesting aspect of the PEP framework is the ability to include different types of search methods (or experiment design methods such as partial factorial designs) into the evaluation of any strategy. Thus, it is hoped that using this framework, we might be able to evaluate the impact of different types of search (or experiment design methods) at different steps in the process of obtaining the goal (or solution). This would involve the computational implementations of each of these search methods, and then the use of the framework to evaluate the impact of each search strategy at different stages of decision-making. For the purposes of initial evaluation, we have used only full factorial search (grid screen designs).

In order to generate the several thousands of experimental data needed in order to validate and modify the simulator model, it is necessary to have a robot agent that can carry out these experiments in the laboratory. If the robot agent was equipped with visual capabilities, then it would be possible to perform observations and record results automatically. If visual capabilities were absent, then, it is possible for a human to take over the responsibilities of observing and recording the results for use by the robot. The PEP framework could be extended thus by embedding the framework with simulator inside a robot agent, which could then perform experiments based on the

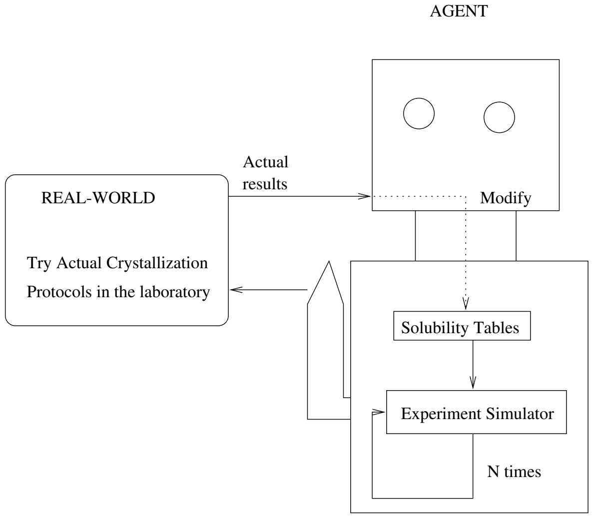


Figure 6.2: Robot agent that has experiment simulator as internal model

predictions from the simulator, and then use the actual response data from real experiments in the lab in order to modify the solubility tables and other parameters that define the model. Figure 6.2 represents an agent that would know how to modify the information that goes into defining the model of the simulator, namely solubility tables and other dependencies. The goal would be to evoke response behavior from the simulator that resembles real world results, and to thereby study the definition of the model that produces such behavior.

6.4.1 Extensibility to Other Domains

We believe that this framework could be used in other domains, such as in the evaluation of response surfaces for determining drug solubility. The evaluation of drug solubilities in different physico-chemical environments is one of the real-world problems that is similar to crystal growing. There are existing mathematical techniques that are broadly classified as Response Surface Methodology (RSM) [40]. The purpose of Response Surface Methodology is to predict the response variable as a function of the input variables as closely as possible to what we might obtain if we did the experiment. The central idea behind RSM is its notion of a center of interest, that is, if one were to do just one experiment, is it at that point that the experiment would be carried out. The mapping of response is performed in terms of a model equation.

There are several types of experimental designs that have been studied in the past, such as factorial designs, equiradial designs, uniform shell designs[40]. These are based on the types of domains and models involved. It is interesting to note that the domain of Pharmacokinetics has a lot in common with the crystal growing domain, which makes it difficult to develop precise mathematical models. Hence, it would require modeling that resembles that described in Chapter

5. Several researchers have been involved over the past few decades in drug design and modeling [63]. Pharmacokinetics is concerned with the rate at which drug molecules cross cell membranes to enter the body, to distribute within it and leave the body, as well as with the structural changes (metabolism) to which they are subject within it [39]. The goal of pharmacokinetics is to initiate the desired drug action, i.e. to achieve the right effect at the right intensity, at the right time, for the right duration, with minimum risk of unpleasantness or harm.

The time course of drug concentration and effect can be seen as a first-order (exponential) process or zero-order process (saturation kinetics). In the former, a constant *fraction* of drug is metabolized/eliminated in unit time. In the latter, a constant *amount* of drug is metabolized/eliminated in unit time. When a drug is infused at a constant rate, the amount in the body and with it the plasma concentration rise until a state is reached at which the rate of administration of drug to the body is exactly equal to rate of elimination (the steady state). It is important to know when the steady state is reached, for prescribing the dosage schedule. Metabolism reactions tend to make a drug molecule progressively more water-soluble and so favor its elimination in the urine.

Modeling this process along the lines of the model developed in Chapter 5, would enable us to see the response of a drug in a simulated environment with variable plasma concentrations, metabolic reactions, and the like. The PEP framework would be very useful in two main ways:

1. enabling a novice student to understand the specifics involved in drug interaction behavior, and

2. enabling the search of effective strategies to understand overall drug response behavior by intelligent design of experiments.

6.5 Where it could be improved

The implementation of the ideas inherent in this framework can be vastly improved. The framework lends itself to a naturally parallel solution. In the development of the prototype the parallel tasks are actually performed sequentially by a serial program. Parallelizing the implementation will enable the framework to be effectively utilized by providing the capability to try several strategies in a short time. This would require research into the types of parallel models that could be designed for this framework and their evaluation and implementation (Foster[25] introduces concepts and tools for designing and building parallel programs).

The model of the macromolecular crystallization process could benefit from an automated procedure whereby its parameters could be tuned to produce a desired response as described in a particular representation. The challenges that this task provides are:

(a) to find an appropriate representation for actual response space and

(b) to design a program such as a Genetic Algorithm ([29, 33, 47, 7, 55, 5, 48, 60]) that can take as input the model and modify it to reduce the error between the response from the model and actual response.

6.6 Summary

The main claims are to show sufficiency of the Parallel Experiment Planning framework developed in Chapter 4 for information management (passive framework) and as an intelligent aid for decision-making (active framework). Table 6.5 depicts the section numbers of the arguments and experiments in this chapter that provide support for these claims.

Claim	**Supported by description in section**
Passive Framework	5.8, 6.2, 6.3.3.1, 6.3.5, 6.4
Active Framework	5.2 - 5.8, 6.2, 6.3, 6.4

Table 6.5: Pointers to arguments and experiments that support the theses

Chapter 7

Conclusions and Future Work

7.1 Conclusions

The results obtained from experiments described in the previous chapter lead us to believe the following:

1. The PEP framework is sufficient for parallel experiment planning.

2. The PEP system provides the necessary information management for parallel experimentation.

3. The PEP system with the simulator and generalization tool can act as an intelligent decision-making aid for planning experiments in parallel.

4. The representation of the framework is flexible and powerful, since it allows a human-agent to employ different tradeoffs and observe effects. The agent can also change goals, for example, the agent can decide not to stop experimenting after obtaining a single crystal, but instead try to find a crystal with better resolution limit of diffraction.

5. Using the PEP framework, it is possible to evaluate the efficacy of different global strategies for planning parallel experiments. The human in the loop can therefore infer or learn heuristics that are useful in describing cost-effective strategies for parallel experiment planning, that vary depending on the type of response behavior or the crude local evaluation function. Thus, using the PEP system it is possible to learn heuristics that represent crystallization protocols or strategic plans.

6. Some representation of evolving boundary between classes of partial results is necessary in order to for overall planning goal (i.e., obtaining a good quality crystal) to be met in a cost-effective way. Results indicate that machine learning algorithms would be one useful representation of choice.

7. The type of feedback from the environment directly affects the choices of the agent. It is seen clearly that when a human agent is not given a representation of the boundary, he/she would tend to pick different portions of the space to search next.

8. The summaries provide useful information that can assist in forming hypotheses about which regions and what types of granularity with which one needs to search next.

9. It is possible to build an approximate model of macromolecular crystal nucleation and growth, using which we can generate different flavors of hypothetical protein response behavior in various types of physico-chemical environments.

10. From a domain standpoint, what I have noticed repeatedly from my experience with varying the parameters in the model, is that experiments with higher concentrations of protein tend to yield results faster (in a week or so). This could simply be because of the way in which the model itself is described. However, if we believe that the simulator of hypothetical macromolecules has any kind of semblance with the proteins in real world, then one could argue that the bottleneck that exists in macromolecular crystallization could very well be resolved simply by spending more time and resources in obtaining larger amounts of protein and purifying it.

11. Using the PEP framework, we can see that it is more cost-effective to search the vast parameter space incrementally, rather than performing a very large initial screen.

7.2 Future Work

The work described in this dissertation directly leads to the following future directions:

1. Parallelization of the entire implementation of the framework. Simple math will indicate that enormous savings in computation time can be achieved by simply parallelizing the implementation. The entire problem lends itself to parallelization since each experiment is independent of another and could be treated as a separate computational process. Therefore, we would have multiple copies of the same program running simultaneously in parallel, with a small number of changes in the input parameters for each. We could save immense time by actually simulating the experiments in parallel. Then, the time taken to collect and observe the results would mirror the reality of observing the results of each experiment over time.

2. Strengthening the physical-chemistry behind the model. This is another direction that can be immediately useful in making the framework acceptable. The research involved would be to make the model more flexible by studying and incorporating actual solubility table information. Such information would be available in journal articles that report the crystallization behavior of various proteins, or from actual experimentation in the lab.

3. Designing and developing an automatic problem generator (i.e. to automatically produce a protein response behavior from specification of desired response behavior, stated as a series of results for a collection of experimental conditions). This is equivalent to reverse engineering or figuring out the model that has the least error with respect to the response data provided. This will be an important step for several reasons, such as,
   - providing us with an ability to quantify the error that exists in the overall model, and
   - providing an automatic way to generate problem instances to be given to the problem-solving agent such as a human or a software program or a robot (eventually).
4. Coming up with a computational representation for the model that resides within the agent in a manner that enables automatic learning while problem-solving.
5. Putting together a set of initial screen design procedures such as the partial factorial screen design [13] and the probabilistic screen design [34], in a manner that could be used as part of the PEP framework implementation to study the impact of different initial screen designs on the nature of the paths taken to achieve the goal for different types of simulated proteins.
6. Developing a simple tutoring system that can enable novice crystallographers to benefit form the experience available within the simulator model. The PEP system can be viewed as providing such a benefit already. It will need to be made accessible to crystallographers world-wide by providing the appropriate links via the internet. Then, we will be able to evaluate the system as a tool based on feedback from the community of users.
7. Applying PEP framework to other domains such as clinical trial design for testing drug efficacy and safety, where the goal is to minimize dosage and maximize benefits while trying to distinguish between toxic and therapeutic effects of a drug.

Domains such as macromolecular crystallization would greatly benefit from computational tools that can:

- first, help in identifying the problems within the domain for which computational assistance could be provided;
- second, actually perform one or more of these identified tasks within a unified framework that facilitates automation; and
- third, analyze and learn effectively from the data collected and stored electronically.

A major portion of research time is actually spent in defining a very ill-defined problem, as is the case in any work that involves the co-operation and collaboration among people in two or more diverse fields, such as macromolecular crystallization and computer science. Once the problems have been identified and well-defined, the process of automation is usually easier, but herein lies another time-consuming part, namely, that of distributing the tool, and educating potential users of the computational tool, such that it may be used in conjunction with actual users to obtain actual data. The collected data will then need to be cleaned (identify and eliminate errors as much as possible) before performing several different types of analyses to uncover patterns or test hypotheses.

In this particular domain, we are faced particularly with all of the above problems, and therefore, a lot of initial work has to be done without actual data, and hence the need for a model. Once again, after the model is built, it can be used in several different ways, and depending on the goals of the users in the domain community and the quality of predictive power of the model, the use will dictate the types of analyses that need to be performed using the data obtained from the model. In this dissertation, I have demonstrated one such use, namely the inclusion of certain types of analyses on partial data as an intelligent aid to the user for designing experiments in parallel.

Certainly, there are several other ways that the model could be used, such as, for producing response data for several (tens of thousands) different proteins for analyses by clustering algorithms, that can group these proteins by response. Another use of the model would be for pure prediction purposes. In order to achieve this goal, the error in the model would need to be quantified, by performing actual experiments in the laboratory and obtaining the true outcomes. These could then be used as training data by a Genetic Algorithm (GA) that could tune the parameter values within the model in a way that minimizes the error between the actual value and that obtained by the simulation. Even though this appears to be a straightforward method to tune the model and enable quantification of error, so that it could be used for prediction, it is by no means an easy task. For one, the time taken to produce actual results in the laboratory is formidable. Also, the number of possible values that could be changed by the GA would be of the order of several thousands. Once the collection of laboratory data using the XtalGrow [58] program has stabilized, this might become a task that it both easier to perform as well as definitely useful from a domain perspective.

As part of the identification of the problems within this domain, I attempted to identify the types of temporal information in this domain, so that the temporal data may be captured in electronic form. Some preliminary work has also been done to develop methods for representing and learning temporal relationships between experimental variables[30]. This work may be also form the basis for future work along these dimensions.

Appendix

Appendix A

A.1 Variables Influencing Macromolecular Crystallization

The following list of twenty two parameters that influence crystallization are based on notes from the crystallization laboratory manual[8].

1. Concentration of precipitant (% w/v)
2. Concentration of macromolecule (mg/ml)
3. pH
4. Purity of the protein or nucleic acid
5. Temperature (4, 20, 30 and 37 degrees centigrade)
6. Humidity
7. Pressure
8. Gravity
9. Substrates, coenzymes and ligands
10. Level of reducing agent or oxidant
11. Preparation and storage of macromolecule
12. Proteolysis and fragmentation
13. Volume of crystallization drop
14. Metal ions
15. Seeding
16. Vibration and sound

17. Degree of denaturation

18. Buffer choice

19. Cleanliness

20. Organism or species from which macromolecule was isolated

21. Age of macromolecule

22. Homogeneity of protein preparation

Appendix B

B.1 Protein *Givens* Defined as Input to Simulator

1. **Protein Name**: Name of protein such as *Alpha Globulase*.
2. **Isoelectric Point (pI)**: The pH at which the protein is electrostatically balanced or neutral.
3. **Number of phases**: The number of solid phases that the protein could exist in. The following set of inputs (4 - 9) are required for each phase.
4. **Phase Name**: Name of solid phase such as *Amorphous*.
5. **Nucleation saturation of a phase**: The minimum saturation level of protein within hanging drop solution required for nucleating that particular solid phase of protein.
6. **Growth saturation of a phase**: The minimum saturation level of protein within hanging drop solution required for growing that particular solid phase of protein. Please note that generally the saturation required for growth is lower than that of saturation required for nucleating that solid phase. Thus, if nucleation occurs at a very high concentrations of protein in the highly supersaturated region, the nucleii could grow very rapidly and deteriorate with respect to internal order represented as the B-factor.
7. **Maximum rate of growth**: This is a real number that represents the maximum rate of growth of nucleii in a particular solid phase of protein.
8. **Number of heterogeneous nucleii**: This is an integer number that represents the number of heterogeneous nucleii that could possibly influence nucleation of the solid phase.
9. **B-factor**: A real number that represents the internal order that exists within molecules that bind together in the solid phase. This number is chosen to be much larger for the amorphous solid phase than the crystalline solid phase. The B-factor is smallest at the time of nucleation and gradually increases with growth of the solid mass or nucleus. Smallest number indicates best internal order and therefore, better crystalline arrangement for X-ray diffraction purposes.

B.2 Experiment *Controls* Defined as Input to Simulator

1. **Number of salts:** The number of salts used in the experiment. For prototype system we typically set this number to one.

2. **Salts**: The simulator contains solubility table definitions for the following four commonly used crystallization salts: Sodium Chloride, Magnesium Chloride, Potassium Chloride and Ammonium Sulfate.

3. **Salt concentration**: Real number that varies between 0.01 to 0.3 M. The unit M refers to molar concentration, or the number of moles of salt per litre of solvent[3].

4. **Precipitating agent**: The simulator contains solubility table definitions for three types of polyethylene glycols based on their molecular weights: Baby PEG, Mama PEG, and Grandpa PEG. Baby PEG is a low molecular weight PEG such as PEG 1000, Mama PEG is equivalent to PEG 6000 or such, and Grandpa PEG is equivalent to PEG 10000 or higher.

5. **Precipitant concentration**: Real number that varies between 0.0 to 100.0 % w/v (weight/volume).

6. **pH**: real number normally between 3.0 and 10.0.

7. **Protein concentration**: Real number between 2 and 20 mg/ml.

Another input to the simulator model is the time of observation or time for which the simulation is to be run. This number is a real number specified in units of weeks.

Appendix C

C.1 Domain Knowledge for a Globulase - Solubility Graphs

The solubility tables were found to be a flexible way to represent the knowledge about solubility of a protein in any reagent. The graphs below indicate the types of values and mathematical functions that were used to produce the desired effects in our model described in Chapter 5. The actual tables were represented as data structures that contained the relevant x and y values for the different parameters that characterized the particular mathematical functions.

The first set of solubility graphs depict the solubility of the two phases of the hypothetical protein termed *Alpha Globulase* with respect to the concentrations of three different precipitants, namely Grandpa PEG or high molecular weight PEG, Mama PEG or medium molecular weight PEG and Baby PEG or low molecular weight PEG.

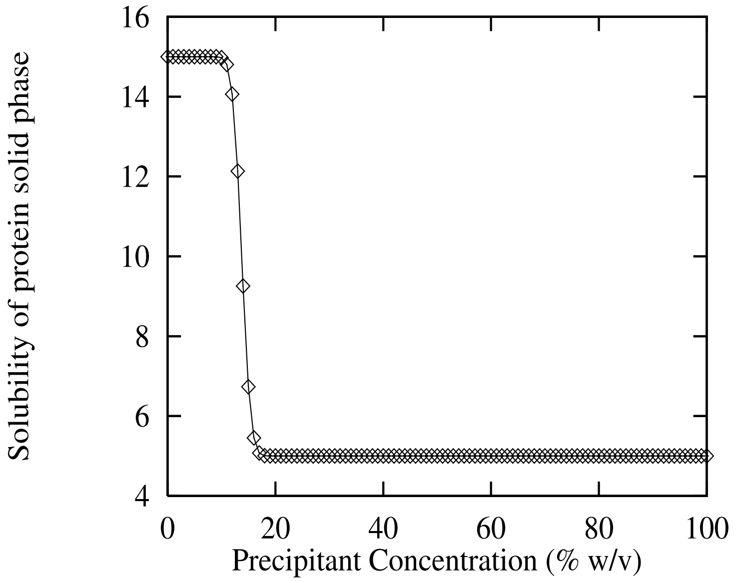


Figure C.1: Solubility of the amorphous phase of protein with respect to **Grandpa PEG** concentration

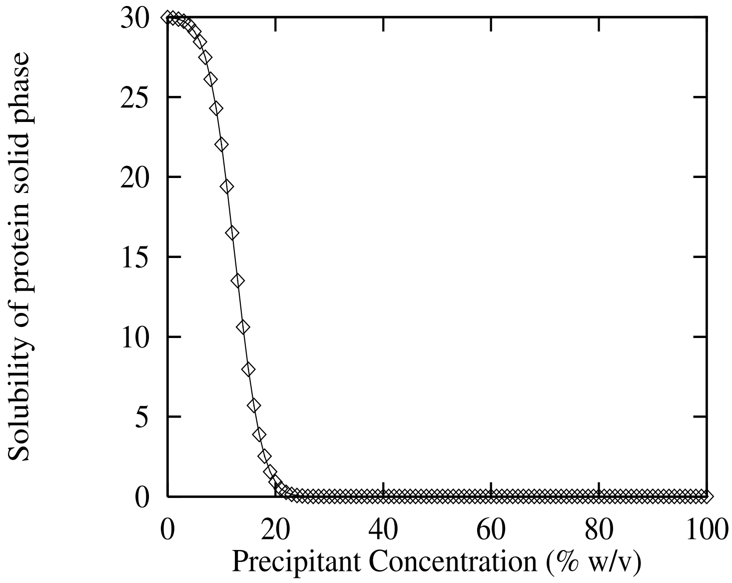


Figure C.2: Solubility of the crystalline phase of protein with respect to **Grandpa PEG** concentration

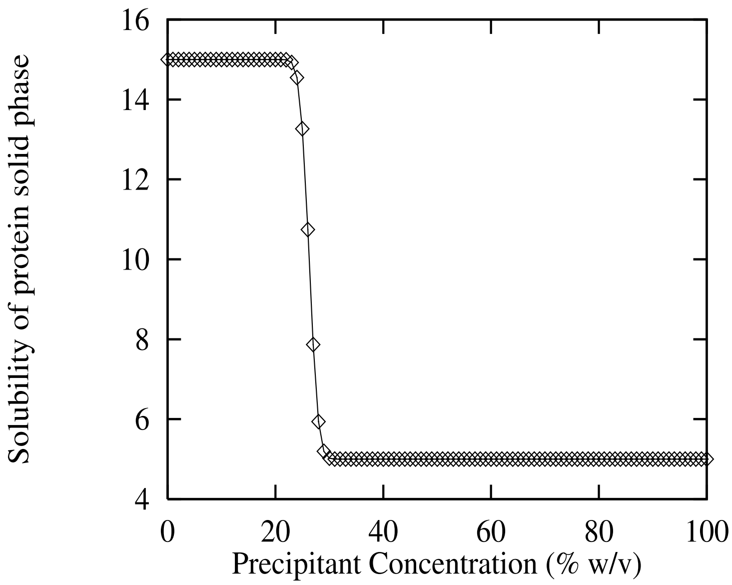


Figure C.3: Solubility of the amorphous phase of protein with respect to **Mama PEG** concentration

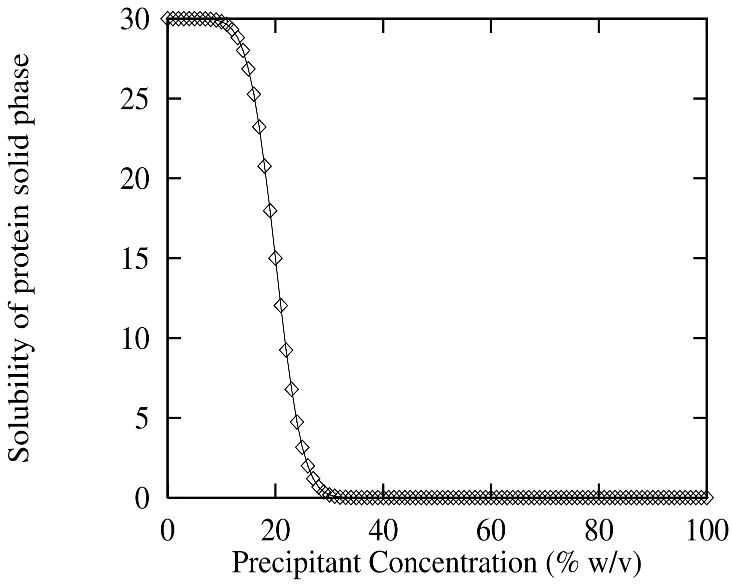


Figure C.4: Solubility of the crystalline phase of protein with respect to **Mama PEG** concentration

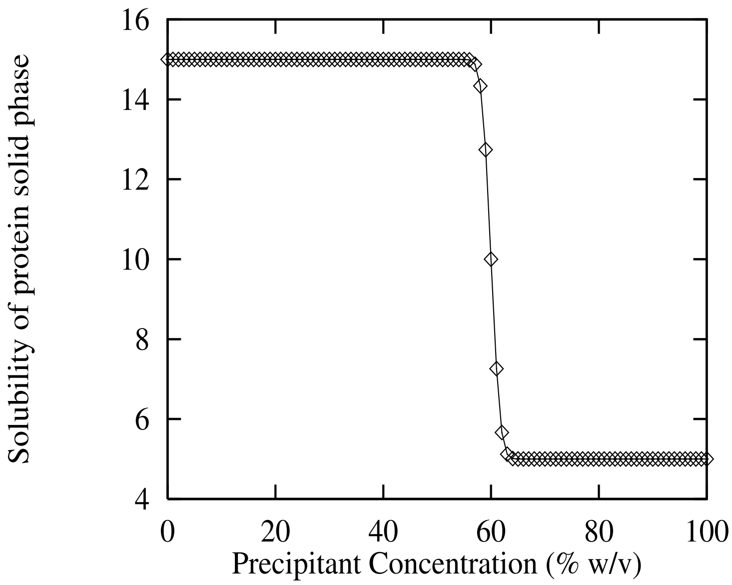


Figure C.5: Solubility of the amorphous phase of protein with respect to **Baby PEG** concentration

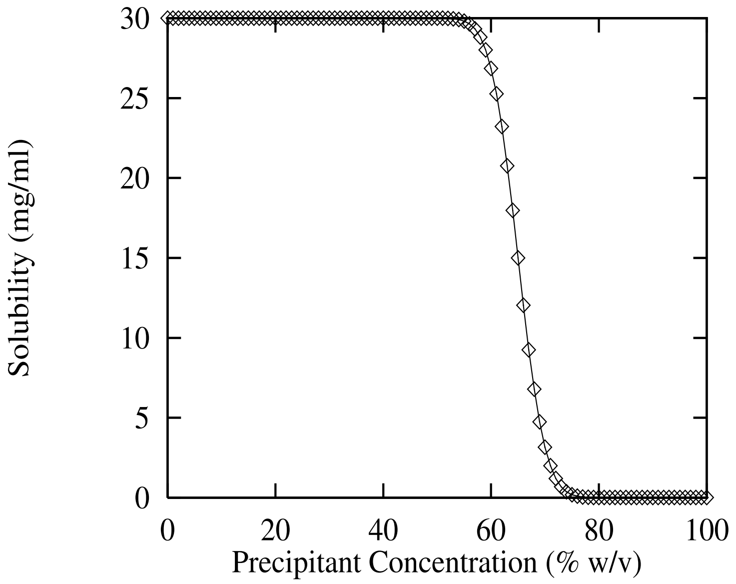


Figure C.6: Solubility of the crystalline phase of protein with respect to **Baby PEG** concentration

The next set of graphs depict the solubility of each protein phase with respect to salt concentration. The commonly used salts are Sodium Chloride, Potassium Chloride, Magnesium Chloride, and Ammonium Sulfate.

A subsequent set of solubility graphs represent the solubility of protein as a function of pH and Ionic strength. Solubility with respect to pH is modeled as an inverted Gaussian, with its lowest point at pH equal to the pI of the given protein. For this graph, we assume that pI of protein is 7.7.

The two-point function that defines the relation between Ionic Strength (or salt concentration) and solubility of protein with respect to I or the concentration of ions in solution is given by the figures in Figure C.17. Please note that the starting solubility S is obtained from the solubility function that models its dependency on pH. Macromolecules tend to have a pronounced minimum solubility at their pI or isoelectric point, when the net charge is zero. Moreover, the phenomenon of *salting-in* is found to occur often at lower ionic strength, since solubility varies very much with pH, and macromolecules tend to be able to form mutually attractive bonds close to their pI.

The critical mass required for nucleation is a function of saturation. It is modeled as shown in Figures C.21 - C.23. The B-factor represents the internal degree of order in a nucleus. The B-factor is best at lower degrees of supersaturation (see Figures C.19 and C.20).

The probability of finding a nucleus of size equal to critical mass required for nucleating different types of nucleii, homogeneous and heterogeneous is modeled as a Gaussian distribution with mean and variance. The mean represents the critical mass of nucleii that are likely to exist given the level of supersaturation. The mean and variance themselves therefore, vary with saturation. These are modeled as shown below.

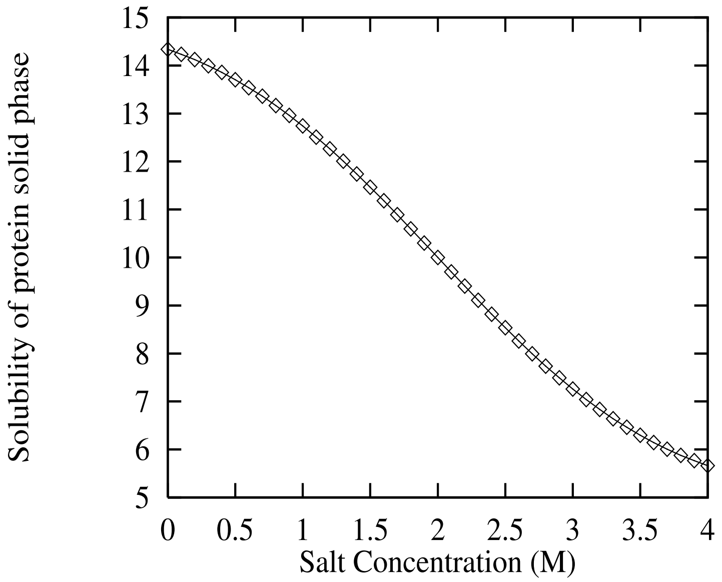


Figure C.7: Solubility of the amorphous phase of protein with respect to **Sodium Chloride** concentration

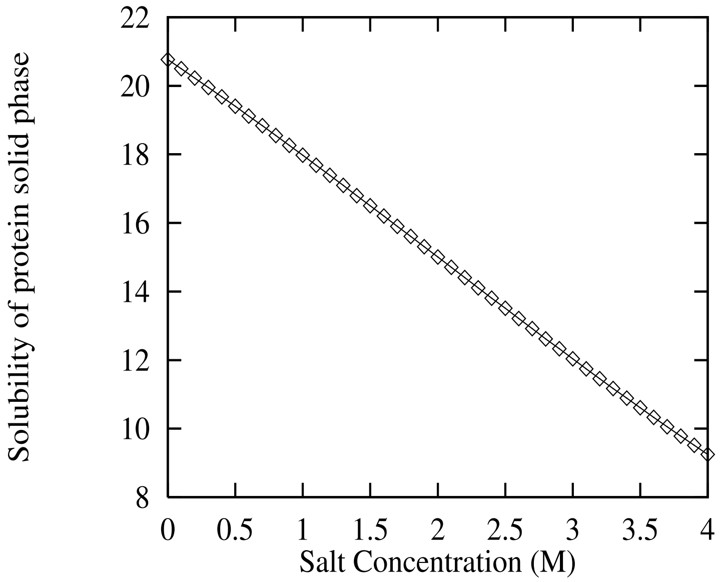


Figure C.8: Solubility of the crystalline phase of protein with respect to **Sodium Chloride** concentration

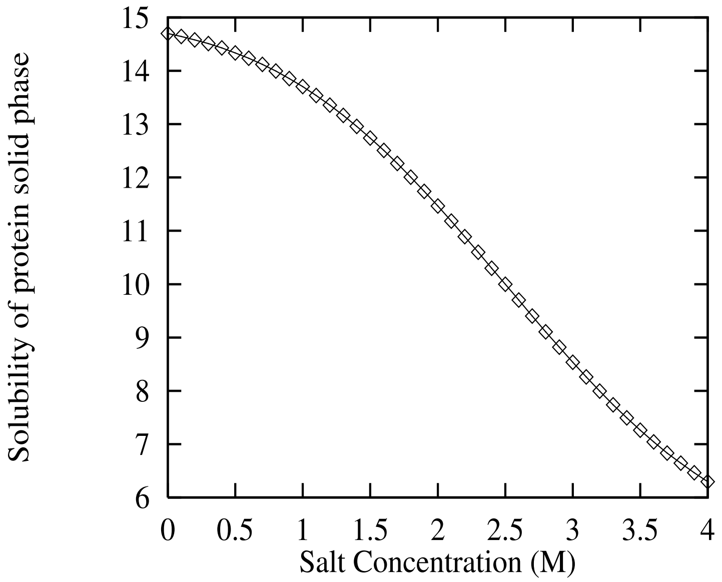


Figure C.9: Solubility of the amorphous phase of protein with respect to **Potassium Chloride** concentration

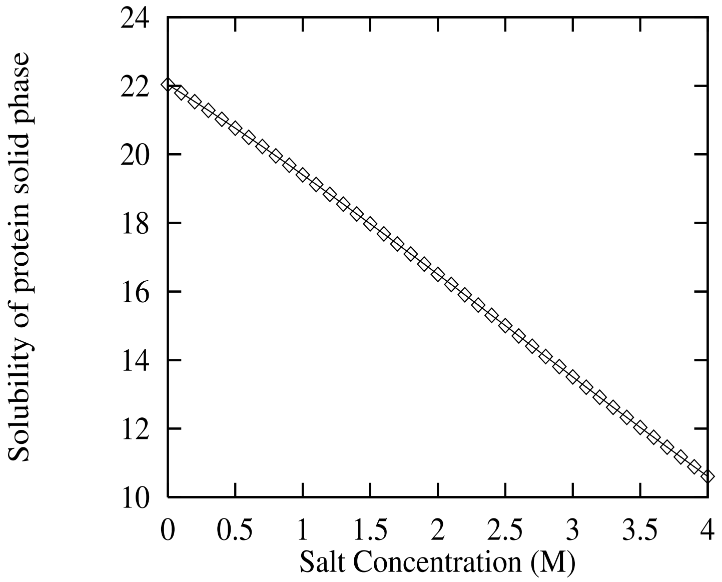


Figure C.10: Solubility of the crystalline phase of protein with respect to **Potassium Chloride** concentration

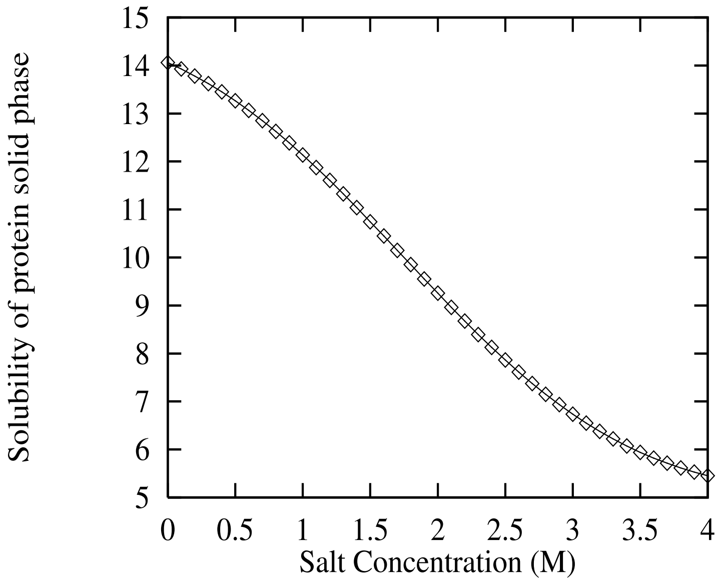


Figure C.11: Solubility of the amorphous phase of protein with respect to **Magnesium Chloride** concentration

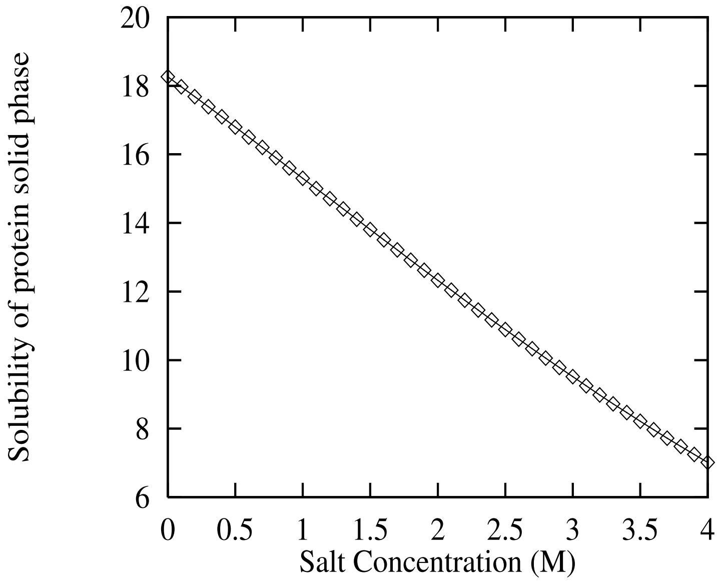


Figure C.12: Solubility of the crystalline phase of protein with respect to **Magnesium Chloride** concentration

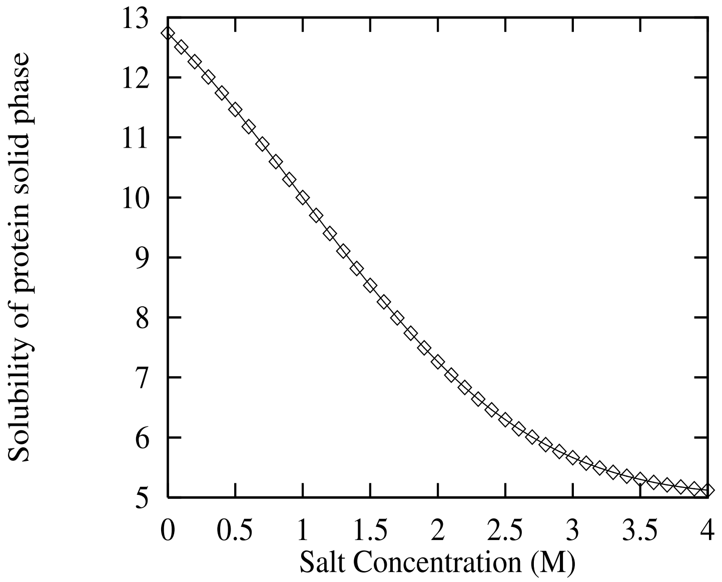


Figure C.13: Solubility of the amorphous phase of protein with respect to **Ammonium Sulfate** concentration

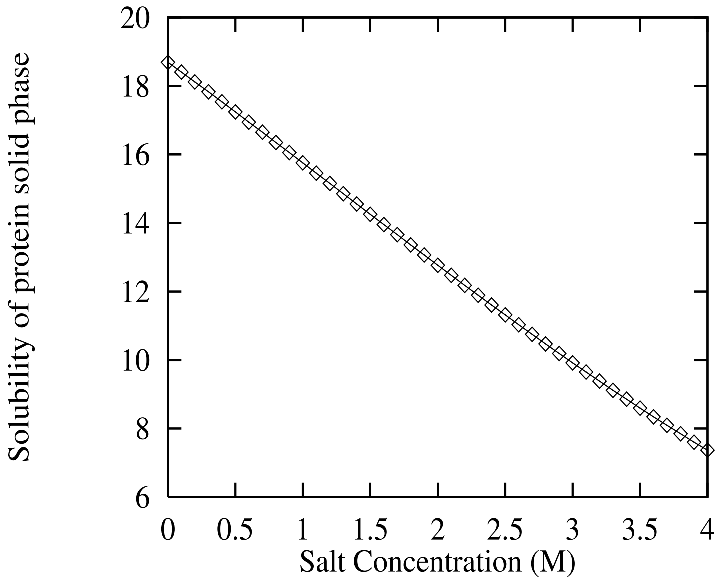


Figure C.14: Solubility of the crystalline phase of protein with respect to **Ammonium Sulfate** concentration

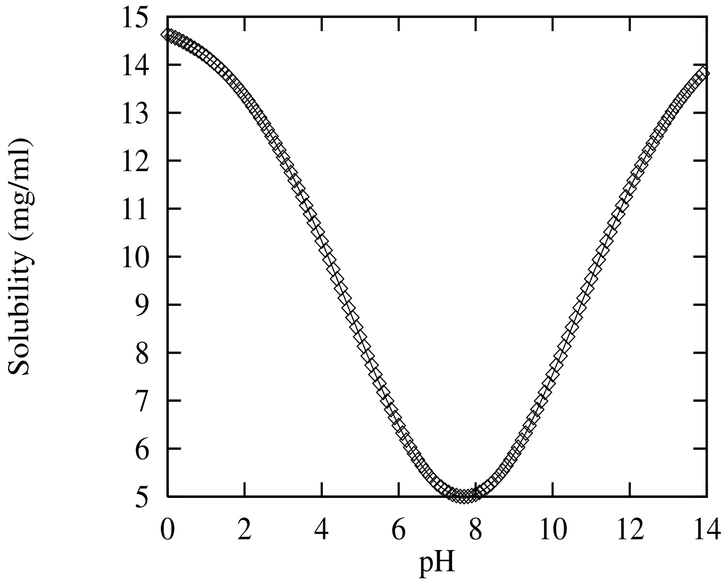


Figure C.15: Solubility of the amorphous phase of protein with respect to **the pH** of the solution

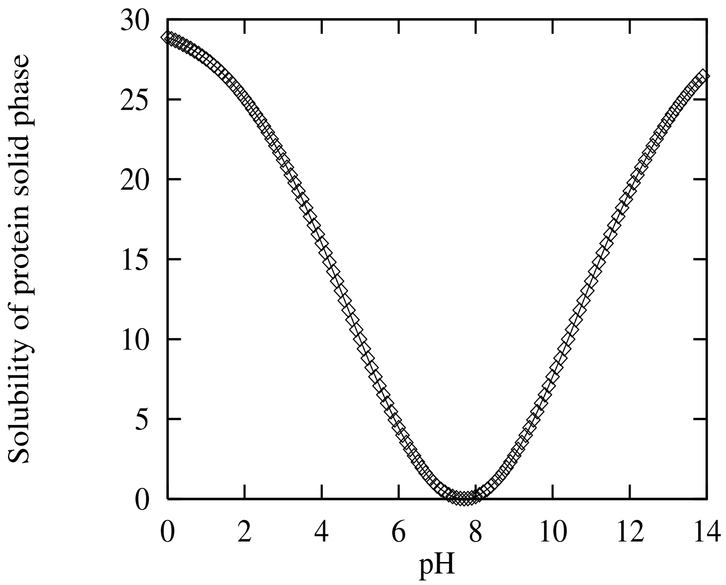


Figure C.16: Solubility of the crystalline phase of protein with respect to **the pH** of the solution

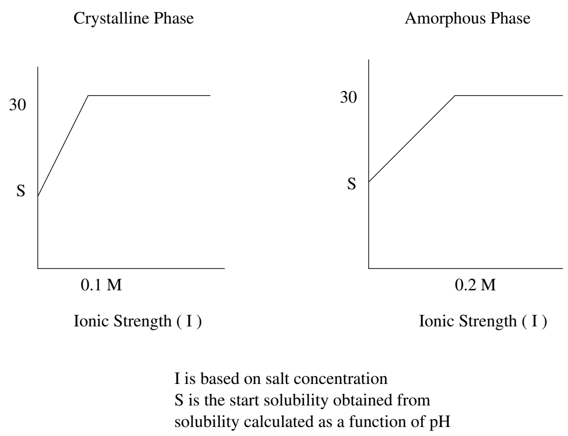


Figure C.17: Solubility of the phases of protein with respect to **the Ionic Strength** of the solution

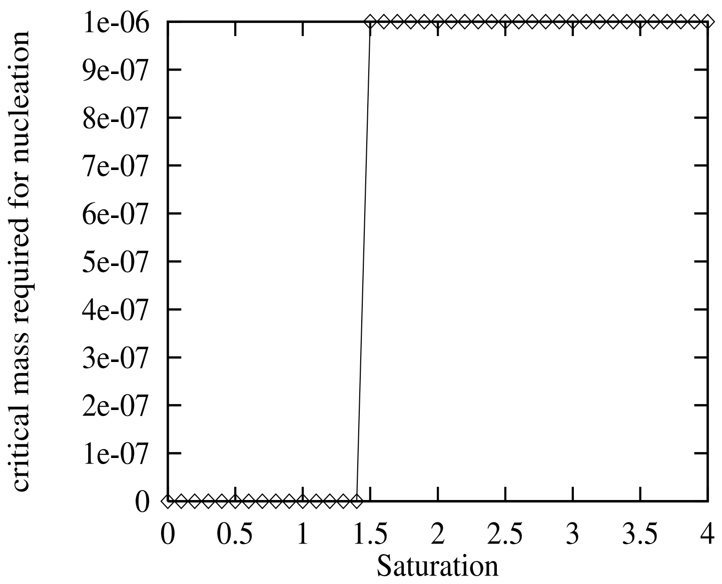


Figure C.18: Saturation vs. critical mass required for nucleating a solid phase

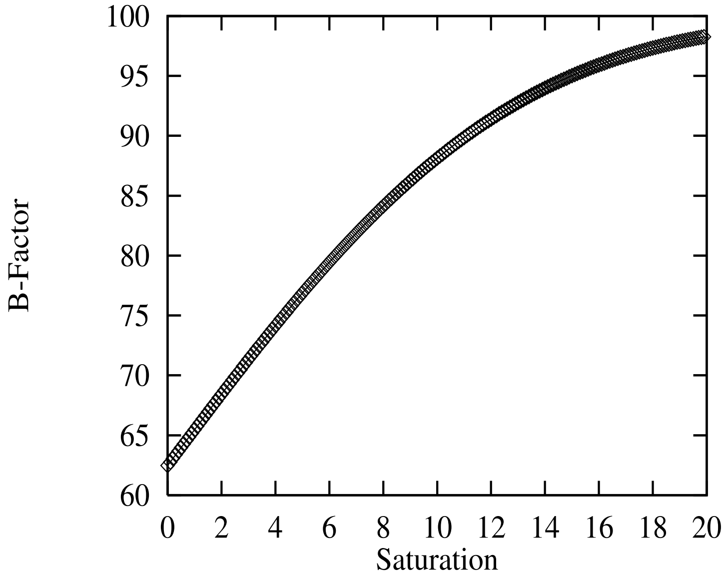


Figure C.19: Saturation vs. B-factor (internal order of nucleus in amorphous phase)

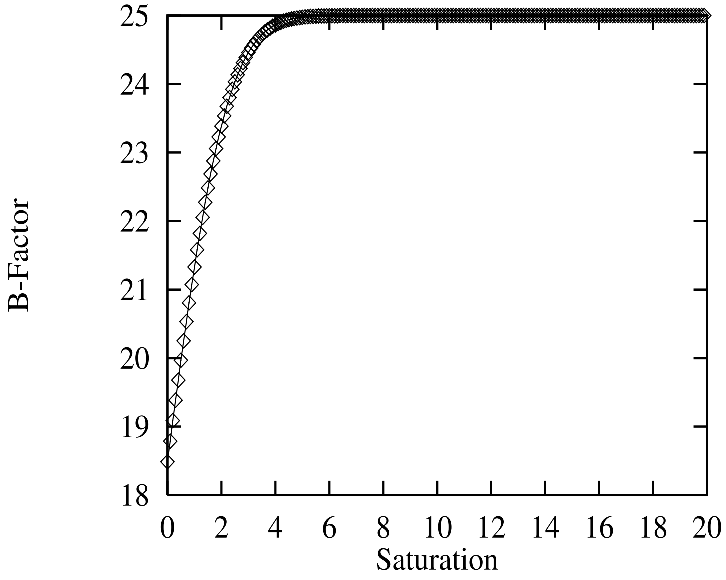


Figure C.20: Saturation vs. B-factor (internal order of nucleus in crystalline phase)

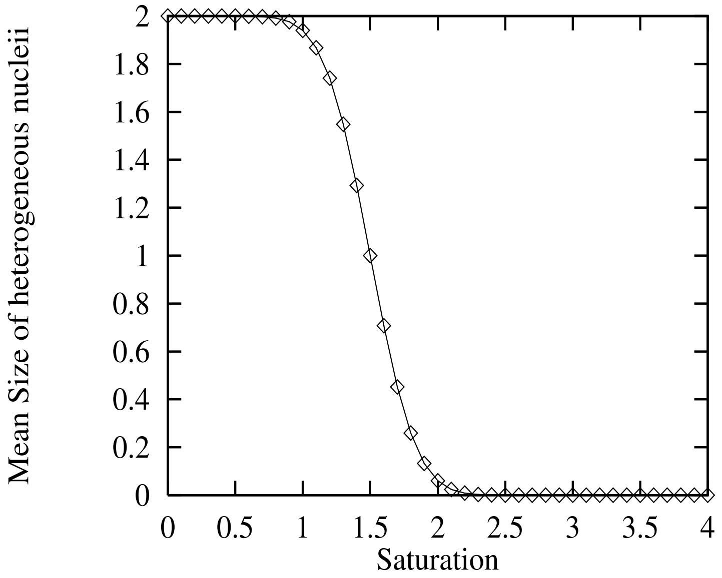


Figure C.21: Saturation vs. Likelihood of critical mass existence for heterogeneous nucleus

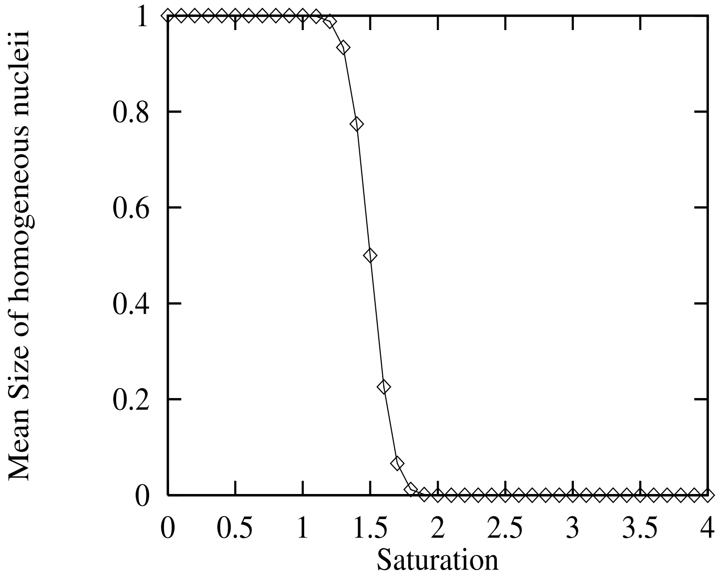


Figure C.22: Saturation vs. Likelihood of (mean) critical mass existence for homogeneous nucleus

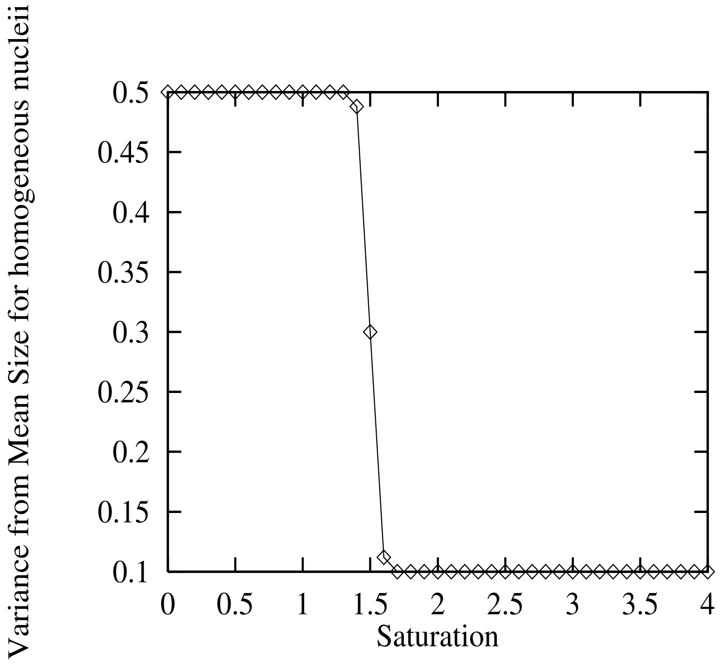


Figure C.23: Saturation vs. Likelihood of (variance) critical mass existence for homogeneous nucleus

The growth rate of a nucleus is modeled such that the nucleus grows very slowly at lower levels of supersaturation, but grows very fast at higher levels. By varying the maximum growth rate that is input to the simulator, we can see the effect as increasing resolution limits of diffraction that represents a bad quality crystal.

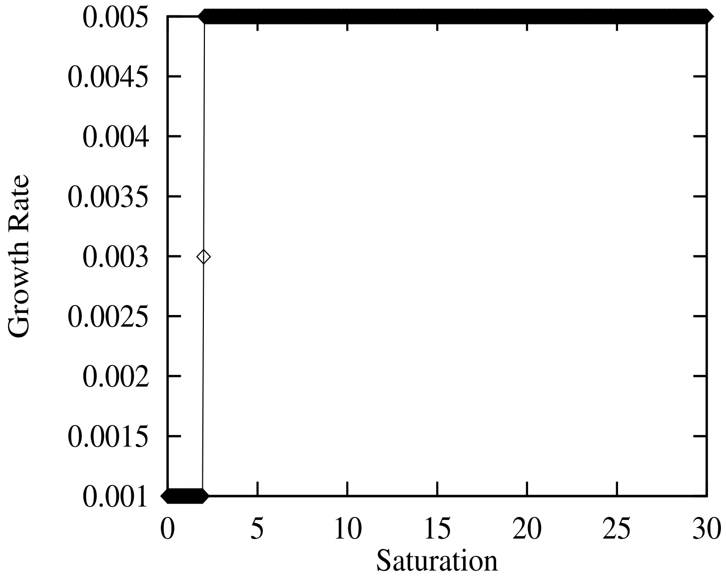


Figure C.24: Saturation vs. Growth rate in amorphous phase

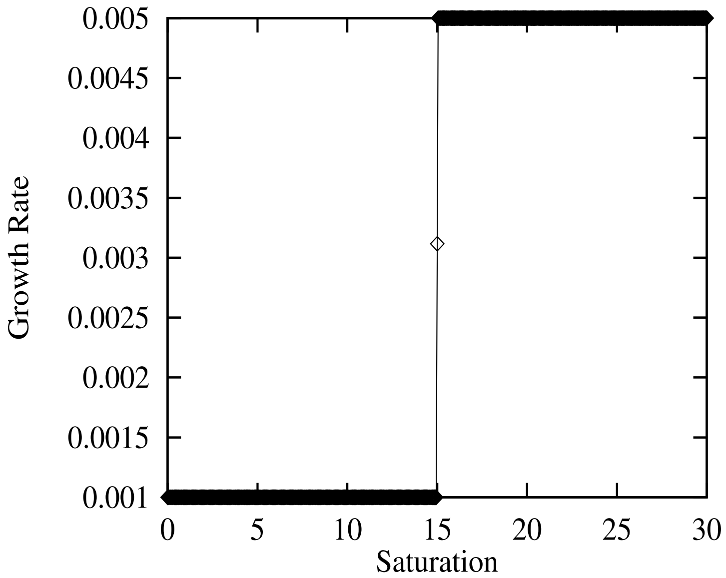


Figure C.25: Saturation vs. Growth rate in crystalline phase

C.2 Definition of Solubility Graphs as Solubility Tables Input to Simulator

A sample description of the domain knowledge in the form of tables is shown below:

```
set_tables()
{
    int i;

   /****** set domain properties of commonly used salts, ppts... *********/

     /****** First set of values for Sodium Chloride in Amorphous phase ***/

     salt_table[0].common_name =
                    calloc(strlen("Sodium Chloride")+1, sizeof(char));
     strcpy(salt_table[0].common_name, "Sodium Chloride");
 /** printf("salt_table[0].common_name = %s\n", salt_table[0].common_name); **/
     salt_table[0].phase[0] = 'A';
     salt_table[0].xy_values[0].x_midpt = 2.0;
     salt_table[0].xy_values[0].y_start = 15.0;
     salt_table[0].xy_values[0].y_end = 5.0;
     salt_table[0].xy_values[0].slope = -3.0;
```

```
/****** Next set of values for Sodium Chloride in Crystalline phase ***/

 salt_table[0].phase[1] = 'C';
 salt_table[0].xy_values[1].x_midpt = 2.0;
 salt_table[0].xy_values[1].y_start = 30.0;
 salt_table[0].xy_values[1].y_end = 0.01;
 salt_table[0].xy_values[1].slope = -3.0;

 /****** First set of values for Potassium Chloride in Amorphous phase ***/

 salt_table[1].common_name =
                calloc(strlen("Potassium Chloride")+1, sizeof(char));
 strcpy(salt_table[1].common_name, "Potassium Chloride");
 salt_table[1].phase[0] = 'A';
 salt_table[1].xy_values[0].x_midpt = 2.5;
 salt_table[1].xy_values[0].y_start = 15.0;
 salt_table[1].xy_values[0].y_end = 5.0;
 salt_table[1].xy_values[0].slope = -3.0;

/****** Next set of values for Potassium Chloride in Crystalline phase ***/

 salt_table[1].phase[1] = 'C';
 salt_table[1].xy_values[1].x_midpt = 2.5;
 salt_table[1].xy_values[1].y_start = 30.0;
 salt_table[1].xy_values[1].y_end = 0.01;
 salt_table[1].xy_values[1].slope = -3.0;

 /****** First set of values for Magnesium Chloride in Amorphous phase ***/

 salt_table[2].common_name =
                   calloc(strlen("Magnesium Chloride")+1, sizeof(char));
 strcpy(salt_table[2].common_name, "Magnesium Chloride");
 salt_table[2].phase[0] = 'A';
 salt_table[2].xy_values[0].x_midpt = 1.75;
```

```
 salt_table[2].xy_values[0].y_start = 15.0;
 salt_table[2].xy_values[0].y_end = 5.0;
 salt_table[2].xy_values[0].slope = -3.0;

/****** Next set of values for Magnesium Chloride in Crystalline phase ***/

 salt_table[2].phase[1] = 'C';
 salt_table[2].xy_values[1].x_midpt = 1.1;
 salt_table[2].xy_values[1].y_start = 30.0;
 salt_table[2].xy_values[1].y_end = 0.01;
 salt_table[2].xy_values[1].slope = -3.0;

 /****** First set of values for Ammonium Sulfate in Amorphous phase ***/

 salt_table[3].common_name =
                 calloc(strlen("Ammonium Sulfate")+1, sizeof(char));
 strcpy(salt_table[3].common_name, "Ammonium Sulfate");
 salt_table[3].phase[0] = 'A';
 salt_table[3].xy_values[0].x_midpt = 1.0;
 salt_table[3].xy_values[0].y_start = 15.0;
 salt_table[3].xy_values[0].y_end = 5.0;
 salt_table[3].xy_values[0].slope = -3.0;

/****** Next set of values for Ammonium Sulfate in Crystalline phase ***/

 salt_table[3].phase[1] = 'C';
 salt_table[3].xy_values[1].x_midpt = 1.25;
 salt_table[3].xy_values[1].y_start = 30.0;
 salt_table[3].xy_values[1].y_end = 0.01;
 salt_table[3].xy_values[1].slope = -3.0;

/****** First set of values for Grandpa PEG in Amorphous phase ***/

 PEG_table[0].common_name =
                  calloc(strlen("Grandpa PEG")+1, sizeof(char));
```

```
 strcpy(PEG_table[0].common_name, "Grandpa PEG");
 PEG_table[0].phase[0] = 'A';
 PEG_table[0].xy_values[0].x_midpt = 13.75;
 PEG_table[0].xy_values[0].y_start = 15.0;
 PEG_table[0].xy_values[0].y_end = 5.0;
 PEG_table[0].xy_values[0].slope = -3.0;

/****** Next set of values for Grandpa PEG in Crystalline phase ***/

 PEG_table[0].phase[1] = 'C';
 PEG_table[0].xy_values[1].x_midpt = 12.5;
 PEG_table[0].xy_values[1].y_start = 30.0;
 PEG_table[0].xy_values[1].y_end = 0.01;
 PEG_table[0].xy_values[1].slope = -3.0;

 /****** First set of values for Mama PEG in Amorphous phase ***/

 PEG_table[1].common_name =
                  calloc(strlen("Mama PEG")+1, sizeof(char));
 strcpy(PEG_table[1].common_name, "Mama PEG");
 PEG_table[1].phase[0] = 'A';
 PEG_table[1].xy_values[0].x_midpt = 26.25;
 PEG_table[1].xy_values[0].y_start = 15.0;
 PEG_table[1].xy_values[0].y_end = 5.0;
 PEG_table[1].xy_values[0].slope = -3.0;


/****** Next set of values for Mama PEG in Crystalline phase ***/

 PEG_table[1].phase[1] = 'C';
 PEG_table[1].xy_values[1].x_midpt = 20;
 PEG_table[1].xy_values[1].y_start = 30.0;
 PEG_table[1].xy_values[1].y_end = 0.01;
 PEG_table[1].xy_values[1].slope = -3.0;
```

```
    /****** First set of values for Baby PEG in Amorphous phase ***/

    PEG_table[2].common_name =
                     calloc(strlen("Baby PEG")+1, sizeof(char));
    strcpy(PEG_table[2].common_name, "Baby PEG");
    PEG_table[2].phase[0] = 'A';
    PEG_table[2].xy_values[0].x_midpt = 60.0;
    PEG_table[2].xy_values[0].y_start = 15.0;
    PEG_table[2].xy_values[0].y_end = 5.0;
    PEG_table[2].xy_values[0].slope = -3.0;

   /****** Next set of values for Baby PEG in Crystalline phase ***/

    PEG_table[2].phase[1] = 'C';
    PEG_table[2].xy_values[1].x_midpt = 65;
    PEG_table[2].xy_values[1].y_start = 30.0;
    PEG_table[2].xy_values[1].y_end = 0.01;
    PEG_table[2].xy_values[1].slope = -3.0;

  /******* Values for pH in Amorphous phase ***/

    PH_table.common_name = calloc(strlen("pH")+1, sizeof(char));
    strcpy(PH_table.common_name, "pH");
    PH_table.phase[0]  = 'A';
    PH_table.xy_values[0].x_mid = initial_state->protein.prot_PI;
    PH_table.xy_values[0].y_level = 15.0;
    PH_table.xy_values[0].y_min = 5.0;
    PH_table.xy_values[0].x_sigma = 3.0;

/******* Values for pH in Crystalline phase ***/

    PH_table.phase[1]  = 'C';
    PH_table.xy_values[1].x_mid = initial_state->protein.prot_PI;
    PH_table.xy_values[1].y_level = 30.0;
    PH_table.xy_values[1].y_min = 0.01;
```

```
 PH_table.xy_values[1].x_sigma = 3.0;

/******* Herein are values for nucleation (homo and hetero) *****/

    homo_mid_table.xy_values[0].x_midpt = 1.5;
    homo_mid_table.xy_values[0].y_start = 1.0;
    homo_mid_table.xy_values[0].y_end = 0.0;
    homo_mid_table.xy_values[0].slope = -3.0;

    homo_mid_table.xy_values[1].x_midpt = 1.5;
    homo_mid_table.xy_values[1].y_start = 1.0;
    homo_mid_table.xy_values[1].y_end = 0.0;
    homo_mid_table.xy_values[1].slope = -3.0;

    homo_width_table.xy_values[0].x_midpt = 1.5;
    homo_width_table.xy_values[0].y_start = 0.5;
    homo_width_table.xy_values[0].y_end = 0.1;
    homo_width_table.xy_values[0].slope = -3.0;

    homo_width_table.xy_values[1].x_midpt = 1.5;
    homo_width_table.xy_values[1].y_start = 0.5;
    homo_width_table.xy_values[1].y_end = 0.1;
    homo_width_table.xy_values[1].slope = -3.0;

    if (debug)
    {
     printf("Done setting homo-mid and width tables\n");
     printf("#nuc = %d\n", initial_state->protein.phase[0].fixed_num_het_nuc);
     printf("#nuc = %d\n", initial_state->protein.phase[1].fixed_num_het_nuc);
     };
    for (i=0; i<initial_state->protein.phase[0].fixed_num_het_nuc; i++)
    {
    het_mid_table[i].xy_values[0].x_midpt = 1.5;
    het_mid_table[i].xy_values[0].y_start = 1.0;
    het_mid_table[i].xy_values[0].y_end = 0.0;
```

```
    het_mid_table[i].xy_values[0].slope = -3.0;

    het_width_table[i].xy_values[0].x_midpt = 1.5;
    het_width_table[i].xy_values[0].y_start = 0.5;
    het_width_table[i].xy_values[0].y_end = 0.1;
    het_width_table[i].xy_values[0].slope = -3.0;
    }
    for (i=0; i<initial_state->protein.phase[1].fixed_num_het_nuc; i++)
    {
    het_mid_table[i].xy_values[1].x_midpt = 1.5;
    het_mid_table[i].xy_values[1].y_start = 2.0;
    het_mid_table[i].xy_values[1].y_end = 0.0;
    het_mid_table[i].xy_values[1].slope = -3.0;

    het_width_table[i].xy_values[1].x_midpt = 1.5;
    het_width_table[i].xy_values[1].y_start = 0.5;
    het_width_table[i].xy_values[1].y_end = 0.1;
    het_width_table[i].xy_values[1].slope = -3.0;
    }
    if (debug) printf("Done setting hetero-mid and width tables\n");

/**** Herein are values for critical mass vs. sat for each phase ****/

    cmass_table.xy_values[0].x_midpt = 1.5;
    cmass_table.xy_values[0].y_start = 0.000001;
    cmass_table.xy_values[0].y_end = 0.0000001;
    cmass_table.xy_values[0].slope = 3.0;

    cmass_table.xy_values[1].x_midpt = 1.5;
    cmass_table.xy_values[1].y_start = 0.000001;
    cmass_table.xy_values[1].y_end = 0.0000001;
    cmass_table.xy_values[1].slope = 3.0;

/**** Herein are values for growth_rate vs. sat for each phase ****/
```

```
    growth_table.xy_values[0].x_midpt = 2.0;
  growth_table.xy_values[0].y_start = initial_state->protein.phase[0].max_rate;
    growth_table.xy_values[0].y_end = 0.001;
    growth_table.xy_values[0].slope = 3.0;

    growth_table.xy_values[1].x_midpt = 15.0;
  growth_table.xy_values[1].y_start = initial_state->protein.phase[1].max_rate;
    growth_table.xy_values[1].y_end = 0.001;
    growth_table.xy_values[1].slope = 3.0;

/**** Herein are values for difflim vs. growth_rate for each phase ****/

    diflim_table.xy_values[0].x_midpt = 0.01;
    diflim_table.xy_values[0].y_start = 100.0;
    diflim_table.xy_values[0].y_end = initial_state->protein.phase[0].max_diflim;
    diflim_table.xy_values[0].slope = 3.0;

    diflim_table.xy_values[1].x_midpt = 0.005;
    diflim_table.xy_values[1].y_start = 25;
    diflim_table.xy_values[1].y_end = initial_state->protein.phase[1].max_diflim;
    diflim_table.xy_values[1].slope = 3.0;
}
```

Bibliography

Bibliography

[1] J. Ambros-Ingerson and S. Steel. Integrating planning, execution and monitoring. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 735–740. Morgan Kaufmann, 1988.

[2] R. Ambrosino, Y. Lee, H.S. Rosenkranz, D.R. Mattison, F.J. Provost, and J. Gomez. The use of a knowledge-based program to predict chemical carcinogenesis in rodents. In *Proceedings of the International Workshop of Predicting Chemical Carcinogenesis in Rodents*, 1993.

[3] P. W. Atkins. *The elements of physical chemistry.* Oxford University Press, 1993.

[4] R. Azencott. *Simulated Annealing: Parallelization Techniques.* John Wiley & Sons, 1992.

[5] U. Bartling and H. Muhlenbein. Optimization of large scale parcel distribution systems by the breeder genetic algorithm (bga). In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 473–480, 1997.

[6] A. Barto, S. Bradtke, and S. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[7] D. Beasley, D.R. Bull, and R.R. Martin. An Overview of Genetic Algorithms: Part 2, Research Topics. *University Computing*, 15(4), 1993.

[8] T. Bergfors. The crystallization lab manual. Technical report, University of Pittsburgh, Biological Sciences, October 1990.

[9] R. Boistelle and J.P. Astier. Crystallization mechanisms in solution. In R. Geigé, A. Ducruix, J. C. Fontecilla-Camps, R. S. Feigelson, R. Kern, and A. McPherson, editors, *Crystal Growth of Biological Macromolecules.* North Holland, 1987. Proceedings of the Second International Conference on Protein Crystal Growth, Bischenberg, Strasbourg, France. A FEBS Advanced Lecture Course.

[10] B. G. Buchanan, E. L. Rissland, P. S. Rosenbloom, H. T. Ng, and J. Sullivan. Intelligent Example Selection. Technical Report ISL-89-9, University of Pittsburgh, Intelligent Systems Laboratory, 1989.

[11] B.G. Buchanan. The role of experimentation in artificial intelligence. *Phil. Trans. R. Soc. Lond. A*, 349:153–166, 1994.

[12] B.G. Buchanan and D.C. Wilkins. *Readings in Knowledge Acquisition and Learning: Automating Construction and Improvement of Expert Systems.* Morgan Kaufman, 1993.

[13] C.W. Carter and C. W. Carter. Protein crystallization using incomplete factorial experiments. *Journal of Biological Chemistry*, 254:12219–12223, 1979.

[14] G. A. Casey and W.W. Wilson. *Journal of Crystal Growth*, 122:95, 1996.

[15] S. Clearwater and F. Provost. RL4: A Tool for Knowledge-Based Induction. In *Proceedings of the Second International IEEE Conference on Tools for Artificial Intelligence*, pages 24–30. IEEE CS. Press, 1990.

[16] Sigma-Aldrich Co. *Biochemicals and Reagents for Life Science Research.* www.sigma-aldrich.com, 1999.

[17] Fulka Chemical Corp. *Laboratory Chemicals and Analytical Reagents.* www.sigma-aldrich.com, 1999.

[18] M. J. Cox and P. C. Weber. An investigation of protein crystallization parameters using successive automated grid searches (sags). In R. Geigé, A. Ducruix, J. C. Fontecilla-Camps, R. S. Feigelson, R. Kern, and A. McPherson, editors, *Crystal Growth of Biological Macromolecules.* North Holland, 1987. Proceedings of the Second International Conference on Protein Crystal Growth, Bischenberg, Strasbourg, France. A FEBS Advanced Lecture Course.

[19] K. W. Currie and A. Tate. O-Plan: the Open Planning Architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[20] A. Ducruix and R. Geige (editors). *Crystallization of Nucleic Acids and Proteins: a Practical Approach.* Oxford Press, 1992.

[21] R. S. Feigelson. The relevance of small molecule crystal growth theories and techniques to the growth of biological macromolecules. *Journal of Crystal Growth*, 90(1), 1988.

[22] E.A. Feigenbaum and B.G. Buchanan. DENDRAL and Meta-DENDRAL: roots of knowledge systems and expert system applications. *Artificial Intelligence Journal*, 59(1), 1993.

[23] R.E. Fikes and N.J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

[24] S. Fortier, J. Glasgow, and F.H. Allen. The Design of a Knowledge-Based System for Crystal Structure Determination. In H. Schenk, editor, *Direct Methods of Solving Crystal Structures.* Plenun Press, London, 1991.

[25] I. T. Foster. *Designing and Building Parallel Programs.* Addison-Wesley Publishing Company, 1995.

[26] M. R. Gary and D. S. Johnson. *Computers and Intractability: A Guide to the theory of NP-Completeness.* H. Freeman, 1978.

[27] G. C. Gilliland. A biological macromolecule crystallization database: a basis for a crystallization strategy. In R. Geigé, A. Ducruix, J. C. Fontecilla-Camps, R. S. Feigelson, R. Kern, and A. McPherson, editors, *Crystal Growth of Biological Macromolecules.* North Holland, 1987. Proceedings of the Second International Conference on Protein Crystal Growth, Bischenberg, Strasbourg, France. A FEBS Advanced Lecture Course.

[28] F. Glover and M. Laguna. *Tabu Search.* Kluwer Academic Publishers, 1997.

[29] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[30] V. Gopalakrishnan and B.G. Buchanan. Representing and Learning Temporal Relationships among Experimental Variables. In *Proceedings of the Fifth International Workshop on Temporal Representation and Reasoning(TIME-98)*, pages 148–155, May 1998.

[31] V. Gopalakrishnan, D. Hennessy, B. Buchanan, D. Subramanian, P. A. Wilcosz, K. Chandrasekhar, and J. M. Rosenberg. Preliminary Tests of Machine Learning Tools for the Analysis of Biological Macromolecular Crystallization Data. Technical Report ISL-94-17, University of Pittsburgh, Intelligent Systems Laboratory, 1994.

[32] V. Gopalakrishnan, D. Hennessy, B. G. Buchanan, and D. Subramanian. The Crystallographer's Assistant. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, page 1451, 1994.

[33] J.J. Grefenstette, C. L. Ramsey, and A. C. Schultz. Learning sequential decision rules using simulation models and competition. *Machine Learning*, 5:355–381, 1990.

[34] D. Hennessy, B. G. Buchanan, D. Subramanian, P. A. Wilkosz, and J. M. Rosenberg. Statistical methods for the objective design of screening procedures for macromolecular crystallization. *submitted to Journal of Applied Crystallography*, 1999.

[35] D. Hennessy, V. Gopalakrishnan, B. G. Buchanan, D. Subramanian, and J. M. Rosenberg. Induction of Rules for Biological Macromolecule Crystallization. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 179–187, 1994.

[36] J. Jancarik and S. H. Kim. Sparse Matrix Sampling: A Screening Method for Crystallization of Proteins. *Journal of Applied Crystallography*, 24:409–411, 1991.

[37] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[38] J. E. Ladner, J. T. Finch, A. Klug, and B. F. C. Clark. High-resolution X-ray diffraction studies on a pure species of transfer RNA. *J. Mol. Biol.*, 72:99, 1972.

[39] D. R. Lawrence, P. N. Bennett, and M. J. Brown. *Clinical Pharmocology.* Churchill Livingstone Publishers, 1997.

[40] G. A. Lewis, D. Mathieu, and R. Phan-Tan-Lau. *Pharmaceutical Experimental Design.* Marcel Dekker, Inc., 1999.

[41] S. Mahadevan. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *Machine Learning*, 22(1):159–195, 1996.

[42] D. McAllester and D. Rosenblatt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on AI*, pages 634–639, 1991.

[43] A. McPherson. Current approaches to macromolecular crystallization. *European Journal of Biochemistry*, 189:1–23, 1990.

[44] A. McPherson. *Crystallization of Biological Macromolecules.* Cold Spring Harbor Laboratory Press, 1999.

[45] R.S. Michalski and R. Stepp. Learning from Observation: Conceptual clustering. In *Machine learning: An artificial intelligence approach.* San Mateo, CA: Morgan Kaufmann, 1983.

[46] T.M. Mitchell. *Machine Learning.* McGraw Hill, 1997.

[47] D. E. Moriarty, A. C. Schultz, and J.J. Grefenstette. Evolutionary Algorithms for Reinforcement Learning. *Journal of Artificial Intelligence Research*, 1999.

[48] D. S. Mullen and R. M. Butler. Genetic algorithms in optimization of adjacency constrained timber harvest scheduling problems. In *Proceedings of the Second Annual Conference on Genetic Programming*, page 379, July 1997.

[49] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley,Reading,Massachusetts, 1984.

[50] C. D. Pegden, R. E. Shannon, and R. P. Sadowski. *Introduction to Simulation Using SIMAN.* McGraw-Hill, Inc., 1990.

[51] J. Penberthy and D. Weld. A sound, complete, partial order planner for ADL. In *Proceedings of the Third International Conference on Knowledge Representation and Reasoning*, pages 103–114, 1992.

[52] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing.* Cambridge university Press, 1993.

[53] F.J. Provost and B.G. Buchanan. Inductive Policy. In *Proceedings of AAAI-92*, 1992.

[54] F.J. Provost and B.G. Buchanan. Inductive Strengthening: The Effects of a Simple Heuristic for Restricting Hypothesis Space Search. In *Proceedings of the Third International Workshop on Analogical and Inductive Inference*, October 1992.

[55] D. Quagliarella, J. Periaux, C. Poloni, and G. Winter. *Genetic Algorithms and Evolution Strategy in Engineering and Computer Science: Recent Advances and Industrial Applications.* John Wiley & Sons, Jan 1998.

[56] J. R. Quinlan. Discovering rules by induction from large collections of examples. *Machine Learning*, 1:81–106, 1986.

[57] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

[58] J. M. Rosenberg and et. al. Intelligent computational aids for crystal growth. In *Acta Crystallogr. Conference*, 1999.

[59] F. Rosenberger, P.G. Vekilov, M. Muschol, and B.R.Thomas. Nucleation and crystallization of globular proteins - what we know and what is missing. *Journal of Crystal Growth*, 5170, 1996.

[60] C. D. Rosin, R. S. Halliday, W.E. Hart, and R. K. Belew. A comparison of global and local search methods in drug docking. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 221–228, 1997.

[61] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[62] C. Samudzi, Fivash, and J. Rosenberg. Cluster analysis of the biological macromolecular crystallization database. *Journal of Crystal Growth*, 123:47–58, 1992.

[63] C. Silipo and A. Vittoria. *QSAR: Rational Approaches to the Design of Bioactive Compounds.* Elsevier, 1991.

[64] M. Stefik. Planning with constraints (MOLGEN:Part 1). *Artificial Intelligence*, 16(2):111–139, 1981.

[65] R. Sutton. Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44, 1988.

[66] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

[67] P. C. Weber. Gridscreen. *Advances in Protein Chemistry*, 41:1–36, 1991.

[68] D. B. Weiner and W. B. Williams. The role of X-ray crystallography in structure based rational drug design. In *Chemical and structural approaches to rational drug design.* CRC Press, Boca Raton, Florida, 1994.