MULTISCALE COMPUTATIONAL METHODS FOR MORPHOGENESIS
AND
ALGORITHMS FOR PROTEIN-PROTEIN INTERACTION INFERENC

A Dissertation

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Chengbang Huang, B.S., M.S.

Jesús A. Izaguirre, Director

Graduate Program in Computer Science and Engineering
Notre Dame, Indiana
July 2005
MULTISCALE COMPUTATIONAL METHODS FOR MORPHOGENESIS
AND
ALGORITHMS FOR PROTEIN-PROTEIN INTERACTION INFERENCEn

Abstract

by

Chengbang Huang

Biocomplexity is the study of the complex relationships among biological entities that are responsible for life. This dissertation addresses two important computational in biocomplexity: morphogenesis and protein-protein interaction networks. Morphogenesis is the development of multicellular organisms. I develop models, algorithms, and software that integrate the genetic regulatory network and physical or generic cellular mechanisms that explain how cells interact to form tissues. The genetic regulation is modeled by a combination of a rule-based state automaton and a set of partial differential equations (PDEs); the generic cellular mechanisms include cell adhesion, cell differentiation, cell growth, mitosis, secretion of morphogens, haptotaxis and chemotaxis.

Protein networks are part of signaling, metabolic, and other biological pathways important to cells and organisms. We develop a new algorithm called Maximum Specificity Set Cover (MSSC) to predict protein-protein interactions. The predictions by MSSC preserve not only the topological characteristics of protein interaction networks, but also the protein co-expression. Our method outscores other prediction methods in quality.
To my parents Anqing Huang and Jinyuan Guo, for their tremendous love and support to me! To my sister, my brother and their families, with whom I have lots of good memories! To my XZBB Lingling, who is always there with me!
CONTENTS

FIGURES .......................................................................................................................................................... vii

TABLES ............................................................................................................................................................ xiii

ACKNOWLEDGMENTS .................................................................................................................................. xiv

I MULTISCALE COMPUTATIONAL METHODS FOR MORPHOGENESIS ...................................................... 1

CHAPTER 1: INTRODUCTION ......................................................................................................................... 2
  1.1 Biological Background ............................................................................................................................ 3
  1.2 Cellular Potts Model (CPM) .................................................................................................................. 5
  1.3 Reaction-Diffusion Mechanism ............................................................................................................. 6
  1.4 Integration of Submodels ...................................................................................................................... 6
  1.5 Simulation Framework ........................................................................................................................... 7
  1.6 Related Work ......................................................................................................................................... 8
    1.6.1 Continuum Equations ....................................................................................................................... 8
    1.6.2 Cellular Automata ............................................................................................................................. 9
  1.7 Directional Finite Difference Method .................................................................................................. 10
  1.8 Contributions ........................................................................................................................................ 11
  1.9 Organization of the Rest of the First Part .............................................................................................. 11

CHAPTER 2: MULTI-MODEL FRAMEWORK .................................................................................................. 13
  2.1 Integration of Sub-models ...................................................................................................................... 13
    2.1.1 Extended Cellular Potts Model ....................................................................................................... 13
    2.1.2 Chemotaxis and Haptotaxis ............................................................................................................ 16
    2.1.3 Mitosis .............................................................................................................................................. 17
    2.1.4 Active Zone ..................................................................................................................................... 18
    2.1.5 Progress Zone ................................................................................................................................... 18
  2.2 Algorithms ............................................................................................................................................. 18
  2.3 Overall Diagram .................................................................................................................................... 20
CHAPTER 7: SUMMARY AND FUTURE WORK ........................................ 71
  7.1 Summary of Results ....................................................... 71
  7.2 Future Work ............................................................... 72

II ALGORITHMS FOR PROTEIN-PROTEIN INTERACTION INFERENCE ........ 74

CHAPTER 8: INTRODUCTION .................................................... 75

CHAPTER 9: MATERIALS AND PREVIOUS METHODS .......................... 78
  9.1 Materials ........................................................................ 78
  9.1.1 Protein Interactions .................................................... 79
  9.1.2 Protein Domains .......................................................... 80
  9.1.3 Microarray Data .......................................................... 80
  9.1.4 Conserved Network Features ........................................ 80
  9.1.5 Assessment of Protein Interactions .................................. 81
  9.1.6 Quality Measures ........................................................ 82
  9.1.7 Orthologous Data ........................................................ 83
  9.2 Previous Prediction Methods ............................................ 83
  9.2.1 Association Method (AM) ............................................. 84
  9.2.2 Maximum Likelihood Estimation (MLE) ............................ 84

CHAPTER 10: MAXIMUM SPECIFICITY SET COVER (MSSC) ............... 85
  10.1 Set Cover Problem ......................................................... 85
  10.1.1 Classical Set Cover ..................................................... 86
  10.1.2 Generalize Set Cover ................................................... 87
  10.2 Transformation of Protein Network to Set Cover Problem ........... 89
  10.3 MSSC Approach ............................................................. 92
  10.4 Prediction ..................................................................... 97

CHAPTER 11: RESULTS OF MSSC AND DISCUSSION ........................... 98
  11.1 Performance Against Other Methods .................................... 98
  11.2 Robustness of MSSC ....................................................... 100
  11.3 Conservation of Network Characteristics ............................... 101
  11.4 Results with High Quality Interactions ............................... 104
  11.5 Results with Orthologous Interactions .................................. 106
  11.6 Disjoint Training and Testing Sets ...................................... 107
  11.7 Dissemination .................................................................. 107
  11.8 Discussion ...................................................................... 108
FIGURES

1.1 Schematic representation of a developing vertebrate limb. There are three axis: anterior–posterior (AP), proximal–distal (PD) and dorsal–ventral (DV). ................................................................. 4

1.2 The limb is divided into two large zones: the frozen zone and the active zone. For simplicity, I assume there are no cell activities in the frozen zone, and cells migrate, differentiate, grow and mitose only in only in the active zone. There is a special area at the end of the active zone in which cells do not differentiate and always stay as non-condensing. I call this area the progress zone. ......................... 7

2.1 Cellular Potts Model: Each cell is assigned a unique integer (index). Extracellular Matrix has index of 0. \( N_i \)'s are the \( i \)-th neighbors of the pixel \( S \). ................................ 14

2.2 Cell type change from non-condensing to condensing: A cell change its type according to the Chemotaxis (TGF\( \beta \)) concentration. Chemotaxis is the activator in a reaction-diffusion system, for example in Equation (2.11). ......................................................... 17

2.3 The overall diagram of COMPUCELL3D. ...................... 21

3.1 Limb Growth Simulation Result 1: The figure shows results at two different stages: the initial state and the state at step 341. At each stage, there are 3 small windows: one for the limb bud, one for the SAM concentration and the other for the activator concentration. The condensing cells in the limb bud are as shown in the figure. For SAM and activator concentration, the dark area represent a high value. Condensing Cells follow the pattern of the high concentration. The progress zone is located at the very tip of the limb bud, and there is no condensing cells in it. ......................................................... 27

3.2 Limb Growth Simulation Result 2. The figure is continuous from Figure 3.1, and shows results at step 681 and step 1061. ............... 28
3.3 2-d simulation without the progress zone: Without the progress zone, non-condensing cells inside the limb bud are separated by the condensing cells, and they cannot move cross the limb. Since the RD bone-like pattern is not absolutely symmetric, one side of the limb grows faster than the other. 29

3.4 3D chicken limb growth and patterning visualized with Ogle. On top of a visualization of all cells, I superimpose one showing only Condensing cells (in grey) for clarity. Apical Zone cells are pink. No condensation occurs in the Apical Zone. NonCondensing cells outside the apical zone are red. 31

4.1 Developmental timeline of chick-limb skeletal patterning. 33

4.2 2-d illustration: x-axis intersects with a closed surface. \( p_1, p_2, \cdots, p_n \) are the intersections, and \( p_0 \) and \( p_{n+1} \) are two points on the x-axis outside the closed surface. The number \( n \) is even. 34

4.3 An 2-d example to clarify how to use an array of vectors to store the limb surface. -1 indicates that the corresponding vector is empty. If the vector is not empty, the number of entries in it must be even. The vector stores the \( y \)-coordinates of the line segments on the closed curve. 35

4.4 Two lattices I use: the dotted-line one for the storage of the limb shape, and the solid-line one for the actual CPM simulation. I use the dotted lattice to approximate the limb surface. A grid point from the solid lattice is actually the center of mass of some pixel in the dotted lattice. 36

4.5 Simulation with Fixed Limb Shape: the limb-like shape is stored in an 2-d array of vectors, and the CPM steps only take place inside the limb. 38

4.6 \( S_1 \) and \( S_2 \) are two limb bud shapes, and \( S_2 \) is the next stage after \( S_1 \). \( S_2 \) does not contain \( S_1 \) completely. The force from the boundary of \( S_1 \) pushes those points toward \( S_2 \). 39

4.7 A CompuCell3D run of the avian limb bud simulation with the irregular shape. Mitosis is turned off. I change shape after certain steps. None of the shapes contains completely others. Cells outside of current shape can be pushed inwards gradually. The number of cells is fixed. 40

4.8 The irregular limb shape with the TGF\(_{\beta}\) pattern inside. 41

4.9 The CPM simulation with irregular limb shapes. 41
5.1 Fitz-Hugh Nagumo kinetics: The red area represents high concentration. (a) A initial wave is generated at the left-hand side. (b) The wave propagates toward right. (c) At certain stage, remove one third of the wave. (d) the left wave becomes a spiral wave. (e) The spiral wave continues to break into smaller spiral waves. 

5.2 2-d Schnakenberg Kinetics: $\gamma$ is a step function which takes values 60.0, 350.0 and 500.0. Different value corresponds to different number of digits, 60.0–1 digit, 350.0–2 digits and 500.0–3 digits.

5.3 3-d Schnakenberg Kinetics: Besides applying different values for $\gamma$, I also need to vary the diffusion in $x$ and $y$ directions to get different number of digits. Compared with 2-d case, 3-d is much more sensitive to the parameters.

5.4 Cross sections of the 3-d pattern: I take 3 different cross-sections of Figure 5.3, each cross-section has a spot-like pattern. The number of spots corresponds to the number of digits.

5.5 Horizontal stripe: In stead of getting a spot-like pattern, the system is more likely to evolve into a stripe-like pattern. This is an example of a horizontal stripe. Mathematically, the stripe-like pattern is more stable than the spot-like pattern.

5.6 Vertical stripes: Sometimes I have several stripes and all of them are vertical.

5.7 3-d simulation for Equations (5.16): Similar to the 3-d modified Schnakenberg Equations (5.11), I apply different values for $\gamma$ and adjust diffusions in $x$ and $y$ directions accordingly to get expected patterns: 1, 2 and 3 digits.

5.8 An example that I can get 4-digit pattern when applying different value $\gamma$.

6.1 An example of triangulation: An irregular domain is divided into a set of small triangles.

6.2 A typical grid point in a triangulation: A point with edges connected it to its neighbors.

6.3 An ghost point for Neumann boundary condition: Point $p_4$ doesn’t exist in the triangulation, add it for the boundary condition implementation. Function value at $p_4$ is defined according to the boundary conditions.

7.1 Limb Shapes $S_1$ and $S_2$ at two different stages: when we map $S_1$ and $S_2$ to the interval [0 1], $p_1$ and $p_2$ are mapped to the same point, then we assume $p_1$ evolve to $p_2$. 
9.1 The fundamental units of proteins (shaded areas) are the domains (geometrical figures), mediating a distinct structure and biological functionality. We assume that the underlying protein domain architectures facilitate the interactions among proteins, allowing us to design a novel method for the inference of protein interactions in *S. cerevisiae.*

10.1 $X$ is the set to cover, and $\mathcal{F} = \{S_i, 1 \leq i \leq t\}$ is a family of subsets of $X$ that can cover $X$.

10.2 The generalized set cover problem: $X$ is a subset of $Y$, and $\mathcal{F} = \{S_i, 1 \leq i \leq t\}$ is a family of subsets of $Y$.

10.3 An interacting protein pair $(P_1, P_2)$: $P_1$ contains 3 domains: $d_1$, $d_2$, and $d_3$, and $P_2$ contains two: $d_4$ and $d_5$. There are 6 domain pairs that are possibly responsible for the interaction $(P_1, P_2)$.

10.4 $d_1$ is contained by 3 proteins: $P_1$, $P_2$, and $P_3$, and Domain $d_2$ is contained by 4: $P_4$, $P_5$, $P_6$ and $P_7$. There are 12 protein protein pair combinations between $d_1$ and $d_2$. Some of the protein pairs interact (solid lines), and others don’t.

10.5 Set of all protein pairs: A solid segment represents an interaction and a dashed segment represents a non-interacting pair. We further group those solid segments (interactions) together.

10.6 A domain pair $(d_1, d_2)$ is a subset of protein pairs (solid or dashed segments). Each $S_i$ ($1 \leq i \leq t$) represents a domain pair.

10.7 Transformation from the protein interaction problem to the set cover problem: The big set $Y$ is taken to be the set of all potential protein pairs, the subset $X$ is taken to be the set of the given protein-protein interactions and finally the family $\mathcal{F}$ is taken to be the set of all the domain pairs.

10.8 The shaded area is already covered by $C$. $U$ is the unshaded area in $X$. The candidate set $S$ is divided into 4 parts a, b, c and d. MSSC chooses a set $S$ with the minimum $\frac{b+c}{a}$ while MESC chooses one with the minimum $\frac{b+c+d}{a}$. The greedy algorithm for MSSC allows overlapping of subcover inside $X$. The overlap actually increases the interaction probability for a protein pair.
11.1 (a) Using CombUI as both the training set and the testing set, we compare the performance of MSSC, AM [78] and MLE [15]. MSSC shows significantly higher specificity and sensitivity. The data for MLE is the original data in [15]. (b) Using DIP as both training set and testing set, we observe that the MSSC algorithm clearly outscores MSC, allowing us to conclude that the design of the MSSC is much more suitable for the appropriate detection of potential protein interactions. We carried out an error analysis by running each algorithm 15 times (inset), allowing us to conclude that the performance of MSSC is consistent.

11.2 With the MLE implemented by us: (a) Using CombUI as both the training set and the testing set, MSSC outscores AM and displays a wider range of sample points than MLE. (b) Using DIP as both training set and testing set, MSSC has the similar result with MLE. Both outscore AM.

11.3 The testing set is the whole DIP set, and different training sets are tried: (a) 10%, (b) 20%, (c) 40%, (d) 60%, (e) 80% and (f) 100% of DIP. The result shows that MSSC is consistent and outscores AM and MSC. The error bars are obtained by performing 10 runs for randomly chosen training sets.

11.4 The testing set is DIP-5, and different training sets are tried: (a) 10%, (b) 20%, (c) 40%, (d) 60%, (e) 80% and (f) 100% of DIP-5. The result shows that MSSC consistently outscores AM and MSC. The error bars are obtained by performing 10 runs for randomly chosen training sets.

11.5 To assess the quality of protein interaction predictions of MSSC, we determine the statistical characteristics of the networks which emerges from our predictions. (a) Utilizing DIP, we observe that the degree distribution follows a power-law (inset). Accounting for interactions which score above a certain probability, we find that the power-law dependence of the degree remains largely unchanged. Similarly, we observe that the power-law dependencies of the clustering coefficient $C(k)$ from the degree as exemplified by the experimental data does largely not depend on the choice of the cutoff value. (b) Utilizing a set of co-expression data, we observe a bell-shaped distribution curve. Accounting only for interactions that score above certain cutoffs we observe that proteins which participate in interactions scoring higher probability strongly tend to be co-expressed.
11.6 (a) Observing that interactions which are embedded in a highly clustered neighborhood exhibit a significantly higher degree of quality, we calculated the hypergeometric clustering coefficient $C_{vw}$ of each link in the original protein-protein interaction network of Yeast. A network composed of interactions that score above a certain threshold is the basis for our predictions, which outscores the original predictions in terms of specificity and sensitivity. (b) Compared to a comprehensive set of coexpression values of protein links (background), the quality of the underlying stripped protein interaction network increases with increased threshold (inset). Since we predict interactions evaluated by an occurrence probability, we only consider predicted pairs that score above $p > 0.9$. The quality of predicted pairs thus obtained is well reflected by an elevated tendency of coexpression, measurements that are supported by low P-values ($P < 10^{-3}$) of a Student’s test.

11.7 (a) Proteins that have orthologs in *H. sapiens*, *M. musculus*, *D. melanogaster*, *C. elegans* and *A. thaliana* are embedded in a highly clustered neighborhood. Elucidating the links between orthologs, we used subnetworks thus emerging for our predictions. We observe that the results thus obtained are comparable to the distributions we obtained with interactions that score above a certain clustering threshold (Fig. 11.6). In particular, we observe that the predictions with a subnetwork of proteins that have an ortholog in all organisms (cross-section) under consideration scores better than with organism specific ones. (b) Compared to a set of coexpression values of protein links (background), the quality of the underlying orthologous protein-protein interaction network increases (inset) by shifting to higher values of coexpression. Considering only predicted pairs that score above $p > 0.9$, the tendency of coexpression further enhances, further supported by low P-values of a Student’s t-test ($P < 10^{-9}$). For these tests $R = T$.

11.8 Performance of MSSC in the test of disjoint training and testing sets. The training sets are 80% of *DIP* and *DIP* - 5, and the testing sets are the remainder. The predicted interactions in *DIP*-5 exhibit a significantly higher degree of quality.

11.9 The web portal for protein-protein interaction: MSSC, AM and MLE are provided for users to test on different sets.

12.1 $Y$: set of all the points; $X$: set of points between the two solid circles; $S_i$ ($1 \leq i \leq 8$): set of points inside the only ellipse which $S_i$ is in; $\mathcal{F} = \{S_i, 1 \leq i \leq 8\}$.

12.2 Domain combination can improve the quality of specificity vs. sensitivity. *DIP* is used as training and testing sets.
TABLES

3.1 PARAMETER VALUES FOR THE 2-D LIMB GROWTH SIMULATION. ........................................... 26
3.2 PARAMETER VALUES FOR THE 3-D LIMB GROWTH SIMULATION. ........................................... 30
5.1 PARAMETER VALUES FOR THE 2-D SCHNAKENBERG KINETICS. ........................................... 51
5.2 PARAMETER VALUES FOR THE 3-D SCHNAKENBERG KINETICS. ........................................... 54
5.3 PARAMETER VALUES FOR 1-2-3 PATTERN. ................................................................. 57
5.4 PARAMETER VALUES FOR 1-2-4 PATTERN. ................................................................. 58
I would like to thank Dr. Izaguirre and Dr. Alber for their support, advice and encouragement. This collaboration was partially supported by an NSF biocomplexity grant IBN-0083653.

I would like to thank Dr. Rajiv Chaturvedi for his work in the same CompuCell project, Dr. Bogdan Kazmierczak for the discussion with him about reaction diffusion systems and advice on the simulations, Mr. Joseph Coffland for optimizing the code, and Dr. Thierry Matthey for his tremendous help on the visualization.

I would like to thank Dr. Chen for his quick insight and precious suggestion on the protein-protein interaction problem, and Dr. Stefan Wuchty for his valuable information and profound discussion on the same problem. Thanks to Dr. Gregory Madey and Dr. Amitabch Chaudhary for serving on my dissertation committee.

I would like to thank all our collaborators, Dr. J. Glazier, Dr. S. Newman, Dr. G. Forgacs, Dr. M. Hentschel and Dr. T. Glimm.

Finally, I would also like to thank my colleague fellows, Trevor Cickovski, Simon Kanaan, Todd Schneider and Kyle Marks for the wonderful work together.
Part I

MULTISCALE
COMPUTATIONAL METHODS
FOR MORPHOGENESIS
CHAPTER 1

INTRODUCTION

This dissertation addresses two computational problems in biocomplexity and consists of two parts: the first part is about how to build a multi-model framework to simulate morphogenesis, and the second part is about new prediction algorithms to infer protein-protein interactions.

For morphogenesis, specifically I simulate the early development of the avian limb starting from experimental measurements of gene regulatory networks, cell and extracellular matrix (ECM) properties, and cell-cell and cell-microenvironment interactions. Our software framework COMPUCELL3D has been developed to model early morphogenesis [9–11, 43].

In the fields of bioinformatics and computational biology an aim is to link the wealth of data (e.g. genetic sequences and genetic regulatory networks) to the understanding of biological processes such as development of multicellular organisms (morphogenesis). Morphogenesis is the structural development of an organism and its organs. It involves cell differentiation, migration, growth and division, bulk changes in tissue shape, and the secretion, resorption and diffusion of extracellular materials (e.g. proteins). Relatively little integrated modeling of multicellular organisms exists; examples are: development models of the Drosophila embryo [61], and analysis of the gene regulatory network of developing sea urchin [14]. These efforts are confined mainly to the modeling of gene regulation. For morphogenesis,
we need to model cell behaviors like release and absorption of diffusible factors, adhesion, and motility [56], in addition to differential regulation of gene activity. The first part of my dissertation is dedicated to this purpose.

Genetic regulation is modeled by a combination of a rule-based state automaton and a set of differential equations, including subcellular ODEs and domain-level reaction-diffusion partial differential equations (PDE). This controls the differentiation of the cells and the cell-cell and cell-matrix interactions that give rise to cell pattern formation and rearrangements such as cell condensation. Cell dynamics is modeled using the extended Cellular Potts Model (CPM), a stochastic model that accurately reproduces cell movement and rearrangement.

1.1 Biological Background

In most organisms the genome seems to strictly guide the organism’s development, but during early multicellular evolution, morphogenesis would be less strictly genetically controlled. Newman and Muller [65] came up with the view that “the present relationship between genes and phenotype is a derived condition, a product of evolution rather than its precondition”. Therefore, it is fitting for us to study the physical mechanisms of the early development of morphogenesis and simulate the process.

Figure 1.1 is a schematic representation of a developing chicken limb. In the avian forelimb these are the humerus, shown as already differentiated (dark grey), followed by the radius and ulna, which are in the process of forming (light gray). Still to form are the wrist bones and digits. The apical ectodermal ridge (AER), a narrow strip of the ectoderm, runs along the distal tip of the limb approximately between the two points intersected by the arrow indicating the anterior-posterior axis.
The AER is necessary for elongation and patterning of the limb. It releases fibroblast growth factors (FGFs), which control mitosis (division) of the mesenchymal cells (which bear an appropriate receptor) in the proximal direction. Rates of cell division in the avian limb have been experimentally determined [50].

Experimentally, the process of limb growth and pattern formation has been understood by considering the space within the developing limb to be comprised of three zones—the apical zone in which only growth takes places, an active zone in which cells rearrange locally into precartilage through mesenchymal condensation [31,63], and a frozen zone in which the condensations have progressed to differentiated cartilage and no additional patterning takes place. Growth continues in both the active zone and the frozen zone.

In the active zone, one or more members of the TGFβ family of growth factors act as the activating morphogen (Newman, 1996 [64]). A laterally inhibitory mor-
phogen, is released from sites on incipient condensation by cells bearing a second FGF receptor. Cells sense the TGF$_{\beta}$ signal. If the signal is strong enough, then they differentiate into cells capable of producing the ECM glycoprotein fibronectin (Frenz et al, 1989 [23]). Fibronectin causes cells to become more adhesive to each other, and then chondrogenesis occurs, and some cells condense into bones finally. It is possible to measure the cell-cell and cell-fibronectin adhesivity from tissue surface tension measurements (Forgacs et al, 1998 [21]).

Biological cells interact with each other by two major means: local interaction by cell adhesion between cells in direct contact or between cells and their surrounding ECM, and longer range interactions such as signal transmission and reception mediated by a diffusing chemical field.

1.2 Cellular Potts Model (CPM)

The CPM, generalized by Glazier and Graner [27] from the Potts model, is currently a commonly used and promising stochastic method for modeling cell and tissue dynamics. In the CPM, cells are represented by points with the same state on a lattice. The transition of the states only happens between two neighboring points with different states. Transition probabilities depend on the energy change. Therefore, the cell dynamics can be manipulated by the method used to compute the energy.

Several additions and improvements have been made to CPM, including cell growth, cell division and cell differentiation [37], chemotaxis [73], haptotaxis, the simulation of extracellular materials [54] and cell polarity [94].
1.3 Reaction-Diffusion Mechanism

Alan Turing was the first one to apply the reaction-diffusion mechanism to the study of biological morphogenesis in his seminal paper “The Chemical Basis of Morphogenesis” [81]. He developed a simple PDE model, simulating a system with two chemicals. One of the chemicals is called an activator, and the other an inhibitor. When the inhibitor diffuses more quickly than the activator, spatial patterns arise from a homogeneous, perturbed initial condition.

In this dissertation, different reaction diffusion systems are studied and proposed to model the TGF\textsubscript{\textbeta} concentration. I am trying to find a good system which can generate bone-like spatial patterns for TGF\textsubscript{\textbeta} in both 2-d and 3-d, and these patterns are used as pre-patterns in the simulation. The pre-patterns direct the later cell condensation into the typical chondrogenic pattern.

1.4 Integration of Submodels

A contribution of our work is to integrate the stochastic CPM with continuum reaction diffusion PDEs to allow the various mechanisms to work in a coordinated fashion.

TGF\textsubscript{\textbeta} plays the activator role in the reaction diffusion equations I use. Its concentration, which has a bone-like pattern, can be simulated by choosing proper parameters for the equations.

Modeling cell dynamics is undertaken by the CPM. Cells sense TGF\textsubscript{\textbeta} concentration which is generated from a reaction diffusion system. If the sensed concentration is greater than a certain threshold, then the cell changes its state to a condensing cell. Condensing and non-condensing cells have different cell-cell adhesions. A condensing cell secretes the substrate adhesion molecule (SAM) and starts responding haptotactically to SAM and condensing into bones. Cell growth happens at the
Figure 1.2. The limb is divided into two large zones: the frozen zone and the active zone. For simplicity, I assume there are no cell activities in the frozen zone, and cells migrate, differentiate, grow and mitose only in only in the active zone. There is a special area at the end of the active zone in which cells do not differentiate and always stay as non-condensing. I call this area the progress zone.

1.5 Simulation Framework

For our simplified model, I divide the whole limb bud into two large zones: the active zone and the frozen zone. Suppose the limb grows upwards. The active zone sits on the top of the frozen zone, as shown Figure 1.2.

No cell activities happen in the frozen zone. All the cells in the active zone are capable of migrating, growing and mitosing. I separate a special area from the active zone and call it the progress zone. Only cells outside of the progress zone can differentiate into condensing cells. Cell dynamics in the active zone are modeled using the CPM. The limb growth is realized by the upward movement of the active
zone. During this movement, the bottom of the active zone is left to be a part of the frozen zone.

The cell differentiation depends on the TGF$_\beta$ concentration. The interaction of the activating morphogen TGF$_\beta$ with the corresponding inhibitory morphogen is modeled using a pair of reaction diffusion (RD) equations. Several RD systems are studied, including the Schnakenberg kinetics, see Equation (5.5). It turns out that the Schnakenberg kinetics can generate bone-like patterns. This is exactly what I want. I use this pattern to guide the precartilage condensation as described above. Another more realistic reaction diffusion system (5.12) is also studied, and similar bone-like patterns are generated successfully.

1.6 Related Work

There is extensive literature on models for developmental biology and morphogenesis in particular. Early mathematical treatment of development was done by Thompson [79] and Waddington [85], which is well summarized by Ransom [68]. Models are typically either reaction-advection-diffusion equations or discrete cellular automata. Our work, implemented in COMPUCELL3D, uses a combined model. There are few programs that offer the multi-model capabilities described here.

1.6.1 Continuum Equations

Purely continuum models of vertebrate limb development have been developed [16, 17]. These models typically consist of reaction-advection-diffusion equations. There are several challenges in these models: the stability of the integrators, which may be severely affected by the advection term; handling the moving boundary for domain growth and the complex geometry; and linking to the biological system.

The approach presented in this dissertation exploits the computational advantages of a continuum reaction-diffusion model while avoiding the difficulties in han-
dling advection by instead using a discrete model. Our model also allows fitting of more detailed biological and biophysical experimental data, including results of genetic regulation.

1.6.2 Cellular Automata

Impressive simulations of morphogenesis have been developed by Marée and Hogeweg [55, 56]. Using the extended Potts model with reaction-diffusion terms (although not continuum equations), they model the self-organization of Dicty into a fruiting body. The processes included are cAMP signaling, differential adhesion, cell differentiation, and production of extracellular matrix. The mechanisms revealed by their model can account for many previously unconnected and unexplained experimental observations. Recently, Merks et al. (2003) [58, 59] have used a combination of lattice-gas automata, advection-diffusion and a discrete model of branching to model coral reef growth.

The work of Levine and collaborators, [47, 49], combines chemotactic fields governed by reaction-diffusion at the cell level. Cells are represented as point sources with cellular-automaton governed dynamics. They use one pixel cells (“bions”) that have internal states for activation and relay of cAMP for Dicty multi-cellular dynamics. An advantage of this model is its simplicity, requiring only one field for each cell type. A disadvantage is that cells have neither extension nor amoeboid-like movement.

More recently, Jiang, Levine and Glazier [45] show that to model the formation of a tip made of pre-stalk cells in Dicty mounds, one needs to use both reaction-diffusion and differential adhesion. Their model is a combination of Levine’s discrete reaction-diffusion mechanism at the cellular level, and Glazier’s extended Potts model. Use of only differential adhesion leads to sorted regions of cells, but with-
out the tip formation. Conversely, reaction-diffusion produces the tip, but without differentiating among cells that compose it.

Simulations that do not use a reaction diffusion mechanism and others that use lattice-gas-model cellular automata to model in-vitro experiments of chicken limb cell condensation are described in [1]. These results also indicate that we need both reaction diffusion and haptotaxis mechanisms to obtain the pattern formation present in avian limbs.

I use a continuum reaction-diffusion model that interacts with discrete cells. I also incorporate mitosis and domain growth as indicated before. The advantage of this model is that it gives a time-dependent representation of transitions to different numbers of parallel elements, conforming more accurately to the continuous patterning of limb development. The continuum/discrete model also offers significant computational savings and is believed to be correct in the biological limit of high cell density during limb bud growth and formation.

1.7 Directional Finite Difference Method

Solving PDEs is a big issue in the whole project. When the domain is confined to the rectangular lattice, the finite difference method (FDM) works well. FDM is conceptually simple and easy to implement. However, I extend the avian limb growth simulation to more realistic geometry eventually. Correspondingly, PDEs have to be solved on the irregular domain. Irregular domains are usually discretized by irregular grids. I can not use the classical FDM anymore. Of course there are other methods for solving PDEs on irregular grids, for example, finite element method (FEM) and the finite volume method (FVM), but they are harder to implement than FDM.

In this dissertation, the directional finite difference method (DFDM) is intro-
duced to extend FDM to irregular grids. Formulae have been developed for second order approximations to the first and second order partial derivatives.

1.8 Contributions

The contributions of this work are:

1. Integrate genetic regulatory network and physical or generic mechanisms to simulate morphogenesis.

2. Design algorithms to realize individual cellular dynamics: chemotaxis, haptotaxis, cell differentiation, cell growth, mitosis, secretion of morphogens and cell density control.

3. Create mathematical 2-d and 3-d models (reaction-diffusion equations) to simulate the skeletal patterning formation.

4. Extend the Finite Difference Method (FDM) to the irregular grid.

1.9 Organization of the Rest of the First Part

1. In Chapter 2, I will present the multi-model framework for CompuCell3D. All the submodels and their realization algorithms will be given there.

2. In Chapter 3, I will present the limb growth simulation results with regular domain for both 2-d and 3-d cases.

3. Chapter 4 is about the limb growth simulation results with irregular domain.

4. In Chapter 5, the reaction-diffusion equations will be discussed. A system of particular reaction-diffusion equations can generate the skeletal patterns which I use for the previous two chapters.
5. In Chapter 6, I will discuss how to extend the Finite Difference Method (FDM) to the irregular grid. I call this new method Directional FDM (DFDM).
CHAPTER 2

MULTI-MODEL FRAMEWORK

2.1 Integration of Sub-models

CompuCell3D integrates multiple models. I create different models to simulate different aspects of cellular dynamics, and then combine them. CompuCell3D is based on the extended Cellular Potts Model (CPM).

2.1.1 Extended Cellular Potts Model

The CPM provides a well-defined, cell-centered model for simulations of morphogenesis [27]. The CPM is a grid-based stochastic model designed to accurately simulate cell interactions and movement. It can reproduce cell membrane fluctuation in a way that matches experiments on cell dynamics, even though it neglects cellular substructures like the cytoskeleton. It models mesenchymal cells, which are relatively isotropic, without requiring further extensions.

For the 2-d CPM, a lattice is defined $L$ as in Figure 2.1, and assign a spin $\sigma_{ij}$ to each lattice point $L(i, j)$. The set of all points with the same spin $\sigma$ define a cell. Each cell has an associated cell type $\tau$.

The energy per unit surface area $J_{\tau\tau'}$ depends on cell type. For our simulation, there are two different cell types, condensing and non-condensing. A condensing cell and a non-condensing cell have different cell-cell adhesion energies. We introduce the cell volume and membrane area as a target area $A_\tau$ and a target perimeter $l_\tau$. 
Figure 2.1. Cellular Potts Model: Each cell is assigned a unique integer (index). Extracellular Matrix has index of 0. $N_i$’s are the $i$-th neighbors of the pixel $S$.

The total energy is:

\[
E = E_{\text{contact}} + E_{\text{area}} + E_{\text{perimeter}},
\]

where

\[
E_{\text{contact}} = \sum_{\sigma,\sigma'} J_{\sigma\sigma'} (1 - \delta_{\sigma,\sigma'}),
\]

\[
E_{\text{area}} = \lambda_a \sum_{\sigma} (A(\sigma) - A_\tau)^2,
\]

\[
E_{\text{perimeter}} = \lambda_p \sum_{\sigma} (l(\sigma) - l_\tau)^2.
\]

Here $\delta_{\sigma,\sigma'}$ is the Kronecker $\delta$-function, which is defined as

\[
\delta_{\sigma,\sigma'} = \begin{cases} 
1, & \text{if } \sigma = \sigma', \\
0, & \text{otherwise}. 
\end{cases}
\]

$A_\tau$ is the target area for cell type $\tau$, and $A(\sigma)$ is the actual cell area of cell $\sigma$. $l_\tau$ the target perimeter for cell type $\tau$, and $l(\sigma)$ is the actual perimeter of cell $\sigma$. $\lambda_a$ and $\lambda_p$ are two scaling factors for area and perimeter energy, respectively. Usually $E_{\text{area}}$ and $E_{\text{perimeter}}$ are together called the volume energy. For 3-d, the volume
energy should consist of $E_{\text{surface}}$ and $E_{\text{volume}}$, where

\[
E_{\text{surface}} = \lambda_s \sum_{\sigma} (S(\sigma) - S_\tau)^2, \quad (2.6)
\]
\[
E_{\text{volume}} = \lambda_v \sum_{\sigma} (V(\sigma) - V_\tau)^2. \quad (2.7)
\]

Similarly, $S_\tau$ is the target surface area for cell type $\tau$, and $S(\sigma)$ is the actual cell surface area of cell $\sigma$. $v_\tau$ the target volume for cell type $\tau$, and $v(\sigma)$ is the actual volume of cell $\sigma$. $\lambda_s$ and $\lambda_v$ are again two constants for surface energy and volume energy, respectively.

The “extended” model includes a term called chemical energy that provides for haptotaxis of cells in response to chemical gradients of morphogens in the extracellular space. A morphogen might be produced as a result of exposure of cells to relatively high concentrations of an activator. Therefore,

\[
H = E_{\text{contact}} + E_{\text{area}} + E_{\text{perimeter}} + E_{\text{chemical}}, \quad (2.8)
\]

where

\[
E_{\text{chemical}} = \sum_{(i,j)} \mu(\sigma_{ij})C(i, j). \quad (2.9)
\]

$C(i, j)$ represents the chemical concentration at the point. For our model, $C(i, j)$ is the Fibronectin concentration. Fibronectin is also called SAM–substrate adhesive molecules.

The extended CPM follows an energy minimization process. At each step, we apply the Metropolis algorithm, choosing a point at random. If it is a cell boundary pixel, we calculate the energy change after flipping it with one of its neighbors with different spin, and accept the proposed flip with a Boltzmann transition probability.
dependent upon temperature $T$:

$$P(\Delta E) = \begin{cases} 
1, & \text{if } \Delta E < 0, \\
 e^{-\Delta E/(\kappa T)}, & \text{otherwise.}
\end{cases} \quad (2.10)$$

where $\kappa$ is the Boltzmann constant. Therefore, state transition is more likely from higher energy state to a lower energy state, but it is also possible in the opposite direction.

### 2.1.2 Chemotaxis and Haptotaxis

Cells can respond to chemical signals by moving along diffusible or substrate-bound concentration gradients of a signal molecule. The first mechanism is *chemotaxis*, the second is *haptotaxis*. We have two types of cells in our model. Initially, all cells are non-condensing cells. A cell senses the TGF$_\beta$ concentration. If the concentration is above a certain threshold, then the cell changes into a condensing cell, see Figure 2.2. The TGF$_\beta$ concentration is modeled by reaction-diffusion equations. A condensing cell has a smaller $J$ with a non-condensing cell, and also it secretes fibronectin to the site it resides on. Fibronectin stimulates in haptotaxis.

Fibronectin concentration contributes directly to the chemical energy calculation, but TGF$_\beta$ concentration controls the cell types and fibronectin secretion. Biologically, chemotaxis causes the precartilage condensation. I take TGF$_\beta$ as the activator in RD system, e.g. the Schnakenberg kinetics:

$$\begin{align*}
\frac{\partial u}{\partial t} &= \gamma(a - u + u^2v) + \nabla^2u = \gamma f(u, v) + \nabla^2u, \\
\frac{\partial v}{\partial t} &= \gamma(b - u^2v) + d\nabla^2u = \gamma g(u, v) + d\nabla^2v.
\end{align*} \quad (2.11)$$

After studying Schnakenberg kinetics, I found that it can generate bone-like patterns for 2-d case. For the more complicated 3-d case, the system needs to be modified to obtain the bone-like pattern I need. I will talk about it later.
Figure 2.2. Cell type change from non-condensing to condensing: A cell change its type according to the Chemotaxis (TGF\( _\beta \)) concentration. Chemotaxis is the activator in a reaction-diffusion system, for example in Equation (2.11).

Another more realistic reaction-diffusion system (Equation 5.12) is also studied. I successfully generate 3-d bone-like pattern with it too, see Chapter 5.

2.1.3 Mitosis

Mitosis is a cell division process. New cells are generated by mitosis. As new cells are formed the active zone grows upwards. In our model, all cells in the active zone are capable of mitosing.

In mitosis, a mitosis doubling time \( T_m \) is set. The cell’s target volume (as described in CPM) is increased successively so that cells can double their volume in \( T_m \). Once the volume of a cell doubles, the cell splits into two child cells of equal size by using breadth-first search. One new cell still uses the old spin for the parent cell; the other is assigned a new spin.

The breadth-first search algorithm is given in [12]. For the breadth-first search, each cell pixel has information about its immediate neighbors in the lattice, allowing the search to navigate the graph through an implied topology. First, a seed pixel
is randomly chosen on the cell’s boundary. Analogous to breadth-first search on a graph, I treat every lattice point as a graph node. Any two points that share an edge are considered to be connected by an edge. Then breadth-first search can be continued. If a pixel outside of the cell is encountered it is ignored.

2.1.4 Active Zone

As I said in the introduction, I assume the cell dynamics, including differentiation, migration, growth and mitosis, happens only in the active zone. The active zone is the head part of the limb bud, and it moves forwards gradually during the limb growth.

2.1.5 Progress Zone

Progress zone is the top part of the active zone. Cells inside do not differentiate, i.e. do not change into condensing cells. It resembles the apical zone in some sense. The progress zone is crucial for 2-d simulation, because it allows cells to move around easily (no precartilage condensation prevents cells from moving horizontally). When the TGF$_\beta$ concentration is not symmetric, without the progress zone one side of the limb may grow faster than the other. We will see more details in the next chapter later.

2.2 Algorithms

Here I put together the algorithms for all the sub-models.

1. Algorithm 1 is the algorithm for the main loop of COMPUCELL.

2. Algorithm 2 is the algorithm for CPM.

3. Algorithm 3 is the algorithm for mitosis (cell division).

4. Algorithm 4 is the algorithm for cell differentiation.
**Algorithm 1** Main Loop.

<table>
<thead>
<tr>
<th>For total number of combined steps do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Solve N steps of the CPM (Algorithm 2)</td>
</tr>
<tr>
<td>Do cell differentiation</td>
</tr>
<tr>
<td>Grow domain for RD and CPM</td>
</tr>
<tr>
<td>Solve RD</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

**Algorithm 2** CPM.

<table>
<thead>
<tr>
<th>For number of grid points in Potts lattice do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attempt substitution of random pixel by a neighbor</td>
</tr>
<tr>
<td>Compute energy change $\Delta H$</td>
</tr>
<tr>
<td>Apply Metropolis criterion</td>
</tr>
<tr>
<td>If cell is growing, attempt division (Algorithm 3)</td>
</tr>
<tr>
<td>end</td>
</tr>
</tbody>
</table>

**Algorithm 3** Mitosis.

<table>
<thead>
<tr>
<th>Breadth-first search do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start from selected cell boundary pixel</td>
</tr>
<tr>
<td>Keep track of visited pixels</td>
</tr>
<tr>
<td>Keep track of neighbors awaiting processing</td>
</tr>
<tr>
<td>Ignore pixels outside dividing cell</td>
</tr>
<tr>
<td>end</td>
</tr>
<tr>
<td>If $S_{\text{target}}$ = the number of visited pixels then</td>
</tr>
<tr>
<td>rename them as a new cell</td>
</tr>
</tbody>
</table>

**Algorithm 4** Cell Differentiation

| If cell type is noncondensing and activator concentration $>$ threshold then |
| cell type := condensing |
| haptotaxis to SAM := on |
| SAM production := on |
| end |
2.3 Overall Diagram

Figure 2.3 is the integration diagram for COMPUCELL3D.

First, we randomly select a lattice point and calculate the energy change $\Delta E$ (Equation (2.8)) for this potential flip. If the flip is not accepted (Equation (2.10)), we return to the starting point. If the flip is accepted, then we check whether the cell should differentiate according to the chemotaxis (TGF$_{\beta}$) concentration and secrete fibronectin (haptotaxis) at the current spot. If mitosis is not turned on, we can go directly to the reaction-diffusion update.

If mitosis is turned on, we check whether the cell volume doubles. If so, the cell mitoses; otherwise, we just jump to the next step to check current cell density. If the cell density is less than a certain threshold, we update the intermediate cell target volume. If the cell density is greater than a certain threshold, we grow the grid then update the intermediate target volume if it is time to grow the grid, otherwise we jump directly to the next step for the RD update. Notice that here we do not update the cell intermediate target volume, if the grid does not grow. This will allow the cells to drift around (migrate) while not growing in volume, so that the cells can be distributed evenly.

2.4 Summary

In this chapter, I introduced the multi-model framework COMPUCELL3D. COMPUCELL3D is based on the extended CPM. More features such as cell differentiation, chemotaxis, haptotaxis, cell growth, mitosis and different zone implementation, are added. I gave algorithms for all the submodels and show how to integrate them.

In the next two chapters, I will present the simulation results for limb growth with regular or irregular domains.
Figure 2.3. The overall diagram of CompuCell3D.
In this chapter, I will present the simulation results with the regular limb shape for both 2-d and 3-d cases. Simulations with the regular limb shape are much easier to implement. I start with the easy case first, and improve our simulation gradually.

Before talking about the actual results, I first present the algorithm I use to control the cell density. Normal cell density inside the real limb is about 70%. Cell density in the simulation has to be manipulated carefully. It is unrealistic to be too dense or too sparse.

3.1 Cell Density Control in Regular Domain

Cellular dynamics happens only in the active zone. Cells in the active zone differentiate, migrate, grow and mitose. To keep the cell density around a certain value, for example 70%, I need to find a way to move the active zone (limb growth) effectively. Otherwise, the active zone will be completely packed with cells (if the active zone moves too slow) or the number of cells in the active zone becomes fewer and fewer (if the active zone moves too fast).

The active zone is a rectangle, so the limb can only grow upwards. I keep track of the cell density in the active zone, and once the density reaches the threshold I move the top and bottom of the active zone upward by a certain number of pixels, see Algorithm 5. It is equivalent to say that I exclude some cell pixels from the
Algorithm 5 Cell Density Control.

| Keep track of total number $t$ of grid points in the active zone |
| Count the number $c$ of cell pixels in the active zone |
| If $c / t \geq \text{threshold}$ |
| Move the top of the active zone by $n_1$ pixels |
| Move the bottom of the active zone by $n_2$ pixels |
| end |

active zone and add some extra-cellular matrix pixels, and accordingly, the new cell density will eventually drop (below the threshold). CPM steps continue in the new active zone. Once the cell density rises above the threshold again, I repeat the same process to move the active zone. The cell density control algorithm will be inserted at the end of the loop for the CPM algorithm (Algorithm 2).

3.2 Metropolis Algorithm for Regular Domains

Algorithm 6 is the Metropolis Algorithm for simulation with regular domain. The cell density control algorithm is also plugged in.

3.3 2-d Simulation Result

In this section, I present the 2-d simulation result, see Figure 3.1 and Figure 3.2. These figures show four snapshots of the simulation.

There are three small windows for each snapshot - the limb bud (Cells) on the left, the Fibronectin concentration (SAM) in the center and the TGF$_\beta$ (Activator) on the right. The TGF$_\beta$ field comes from the solution of Schnakenberg kinetics (Equation 5.5). This field represents the concentration of the activator.

The initial limb height is one-sixth of the total height. Some cells are pre-positioned in the left window as the starting point. All the cells are initially non-condensing, and capable of mitosing. The active zone is the whole limb, so there is no frozen zone yet (see Chapter 1 for the zone division). The progress zone is just
Algorithm 6 Metropolis algorithm for regular domains.

begin
    Initialize the active zone and the progress zone;
    Initialize the Chemotaxis and Fibronectin concentration field;
    For i=1 to total number of steps do
        Randomly pick a cell boundary pixel p1;
        Randomly pick a pixel p2 from p1’s neighbors with different spin;
        Decide cell state according to Chemotaxis field;
        Calculate the energy change for the potential flip from p1 to p2;
        // Energy including interaction, surface, volume and haptotaxis.
        If flip accepted
            Update cell volume and surface area;
            Mitose if necessary;
            Secrete Fibronectin at current spot;
            Calculate cell density in the active zone;
            If cell density greater than threshold
                Update the active zone;
            Update Chemotaxis concentration field; // Reaction-diffusion.
        endif
    endfor
end
the top part of the active zone. The extra-cellular medium (ECM) is treated as a special type of cell.

When simulation begins, cells begin to evolve according to the CPM. At each Monte-Carlo step, the energy change is calculated according to the proposed flip between two neighboring points, then I calculate the probability for the proposed flip, and use this probability to decide whether the proposed flip should be accepted or not. Meanwhile, a cell senses the TGF$_\beta$ concentration. If the concentration is greater than a certain threshold, then the cell changes its type to a condensing cell (if it is not in the progress zone) and begins to secrete a new morphogen called SAM. The central window represents SAM concentration. A condensing cell has a smaller cell-cell adhesion energy $J$, which makes a flip more likely to be accepted because of the lower energy cost, so condensing cells are more likely to stick to each other.

A mitosis doubling time $T_m$ is set. The volume of a cell grows linearly until reaching twice the target volume in $T_m$, then the cell splits into two child cells. Once the simulation is initialized, mitosis is the only source for new cell generation.

Table 3.1 contains the parameter values used in the simulation.

I mentioned early the importance of the progress zone to 2-d simulation. Figure 3.3 is a simulation without the progress zone implemented. Without the progress zone, non-condensing cells inside the limb bud are separated by the condensing cells which stay almost fixed to the lattice, and they cannot move across the limb. Since the RD bone-like pattern is not absolutely symmetric, one side of the limb grows faster than the other. With the progress zone implemented, cells inside can move in each direction freely, so the cells can be distributed evenly.
TABLE 3.1

PARAMETER VALUES FOR THE 2-D LIMB GROWTH SIMULATION.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice size</td>
<td>$300 \times 100$</td>
</tr>
<tr>
<td>Active zone size</td>
<td>$50 \times 100$</td>
</tr>
<tr>
<td>$J_{\text{cell, cell(noncondensing)}}$</td>
<td>7.0</td>
</tr>
<tr>
<td>$J_{\text{cell, cell(condensing)}}$</td>
<td>0.5</td>
</tr>
<tr>
<td>$J_{\text{cell, medium}}$</td>
<td>2.0</td>
</tr>
<tr>
<td>$\lambda_a$</td>
<td>3</td>
</tr>
<tr>
<td>Initial cell size</td>
<td>$4 \times 4$</td>
</tr>
<tr>
<td>$S_{\text{target}}$</td>
<td>16</td>
</tr>
<tr>
<td>TGF$_\beta$ threshold</td>
<td>0.75</td>
</tr>
<tr>
<td>$\mu$ for chemical energy</td>
<td>25.0</td>
</tr>
<tr>
<td>SAM production rate</td>
<td>0.005 units/step</td>
</tr>
<tr>
<td>T, temperature</td>
<td>7.0</td>
</tr>
<tr>
<td>Mitosis doubling time</td>
<td>85 Monte-Carlo iterations</td>
</tr>
<tr>
<td>Active zone growth rate</td>
<td>1 pixel every 4 iterations</td>
</tr>
<tr>
<td>Total number of iterations</td>
<td>1100</td>
</tr>
</tbody>
</table>
Figure 3.1. Limb Growth Simulation Result 1: The figure shows results at two different stages: the initial state and the state at step 341. At each stage, there are 3 small windows: one for the limb bud, one for the SAM concentration and the other for the activator concentration. The condensing cells in the limb bud are as shown in the figure. For SAM and activator concentration, the dark area represent a high value. Condensing Cells follow the pattern of the high concentration. The progress zone is located at the very tip of the limb bud, and there is no condensing cells in it.
Figure 3.2. Limb Growth Simulation Result 2. The figure is continuous from Figure 3.1, and shows results at step 681 and step 1061.
3.3 2-d Simulation Result

Without the progress zone, non-condensing cells inside the limb bud are separated by the condensing cells, and they cannot move cross the limb. Since the RD bone-like pattern is not absolutely symmetric, one side of the limb grows faster than the other.

3.4 3-d Simulation Result

The 3-d simulation is the extension of the previous 2-d simulation. The RD pattern for 3-d simulation is generated by a modified Schnakenberg system, see Equation (5.12). Figure 3.4 is the 3-d simulation result. Table 3.2 contains the parameter values used in the simulation.

3.5 Summary

In this chapter, I presented the 2-d and 3-d simulation results in a regular domain. I also presented the cell density control algorithm which can keep the cell density evenly distributed at the required level. In the next chapter, I will present more simulation results with an irregular domain.
TABLE 3.2

PARAMETER VALUES FOR THE 3-D LIMB GROWTH SIMULATION.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lattice size</td>
<td>$71 \times 31 \times 211$</td>
</tr>
<tr>
<td>Active zone size</td>
<td>$71 \times 31 \times 20$</td>
</tr>
<tr>
<td>$J_{\text{cell, cell}}$ (noncondensing)</td>
<td>7.0</td>
</tr>
<tr>
<td>$J_{\text{cell, cell}}$ (condensing)</td>
<td>0.5</td>
</tr>
<tr>
<td>$J_{\text{cell, medium}}$</td>
<td>2.0</td>
</tr>
<tr>
<td>$\lambda_s$</td>
<td>3</td>
</tr>
<tr>
<td>$\lambda_v$</td>
<td>77</td>
</tr>
<tr>
<td>Initial cell size</td>
<td>$4 \times 4 \times 4$</td>
</tr>
<tr>
<td>$S_{\text{target}}$</td>
<td>16</td>
</tr>
<tr>
<td>$V_{\text{target}}$</td>
<td>96</td>
</tr>
<tr>
<td>TGF$\beta$ threshold</td>
<td>0.8</td>
</tr>
<tr>
<td>$\mu$ for chemical energy</td>
<td>25.0</td>
</tr>
<tr>
<td>SAM production rate</td>
<td>0.005 units/step</td>
</tr>
<tr>
<td>$T$, temperature</td>
<td>2.0</td>
</tr>
<tr>
<td>Mitosis doubling time</td>
<td>85 Monte-Carlo iterations</td>
</tr>
<tr>
<td>Active zone growth rate</td>
<td>1 pixel every 4 iterations</td>
</tr>
<tr>
<td>Total number of iterations</td>
<td>1100</td>
</tr>
</tbody>
</table>
Figure 3.4. 3D chicken limb growth and patterning visualized with Ogle. On top of a visualization of all cells, I superimpose one showing only **Condensing** cells (in grey) for clarity. Apical Zone cells are pink. No condensation occurs in the Apical Zone. **NonCondensing** cells outside the apical zone are red.
So far all the simulations are based on regular limb shape, either a rectangle for the 2-d case or a rectangular box for the 3-d case. A realistic limb actually has a irregular shape, so my simulation should eventually be able to handle the growth of the irregular shape. A complicated geometric shape brings some extra issues.

4.1 Storage of Irregular Limb Shape

COMPUCELL3D handles regular domains, a rectangle for 2-d case and a box for 3-d case. In reality, the chicken limb does not look like a rectangle (2-d) or a box (3D) and has an irregular shape, see Figure 4.1. I will discuss about how to store an irregular shape first, then discuss about the mechanisms to generate a limb-like irregular shape.

After the irregular shape is obtained, I need to find a proper way to store the limb boundary, a way that can be implemented easily and is also memory-efficient. More importantly, the storage method should be able to tell whether a point is inside or outside the limb. For the Potts Model, I choose a pixel first, then decide what to do next. If the pixel is inside the limb, the regular cellular automaton is carried on; otherwise, I just simply discard it. Therefore, the way I store a limb surface should be efficient to tell whether a randomly chosen pixel is inside or outside the limb.
Figure 4.1. Developmental timeline of chick-limb skeletal patterning.

Intuitively, a circle (or a rectangle) divides a plane into two parts with the circle as their common boundary. Jordan Curve Theorem [72] tells us that it also holds for a general simple closed curve.

**Jordan Curve Theorem 4.1.1.** A simple closed curve in a plane separates the plane into two regions of which it is the common boundary.

One region is said to be inside the simple closed curve, and the other region outside. To go from a point on one side of the curve to a point on the other side, one must necessarily cross the curve. This seemingly trivial and self-evident theorem is actually difficult to prove mathematically.

This theorem can be extended to the 3D case, i.e. a simple closed surface separates the 3D space into two regions of which it is the common boundary. What I are interested in is the number of intersections between a straight line and a simple closed surface.

**Lemma 4.1.2.** Suppose a line intersects with a simple closed surface in 3D space, then the number of intersections is even, if the tangent points (degenerate case of intersection) do count.

*Proof.* Without loss of generality, I assume the line is the x-axis and call the intersections $p_1, p_2, \cdots, p_n$ in increasing order (none of them are tangent points), see
Figure 4.2. 2-d illustration: x-axis intersects with a closed surface. \( p_1, p_2, \cdots, p_n \) are the intersections, and \( p_0 \) and \( p_{n+1} \) are two points on the x-axis outside the closed surface. The number \( n \) is even.

Figure 4.2. \( p_0 \) is a point on the x-axis to the left of all the intersections and \( p_{n+1} \) is a point to the right.

If we walk along the x-axis from \( p_0 \) to \( p_{n+1} \), every time we hit a intersection point, we either walk inside the surface or outside the surface. Since \( p_0 \) and \( p_{n+1} \) are both outside the surface, if we walk inside the surface, we have to walk outside later. “Inside” and “outside” are pairwise in this way, and so are the intersections. Therefore, the number of intersections \( n \) is even.

I use a 2-d array of vectors to store the limb surface. When I put the irregular limb shape into a \( l \times n \times m \) regular lattice which is big enough to accommodate the whole limb, and suppose the length of each small interval in all the directions is 1. The irregular shape will intersect with the lattice. I use the closest lattice points to approximate the intersections. I only keep all the enclosed cubes to represent the limb body, and throw away all the degenerate line segments and faces. I project all the intersections vertically to the xy-plane. For a vertical line (in the lattice), it may intersect with the irregular shape several times, or may not intersect at all. I store the z-coordinates of all the intersections for one vertical line in sorted order in one vector. If the intersection is a line segment, only store the two end points of the line segment only. If there is no intersection, then the corresponding vector is empty, see Figure 4.3. Figure 4.3 is a 2-d example. I use a 10 \times 10 \) grid to accommodate an
irregular closed curve. I can use an array of vectors to store the curve. The array is of size 10. The actual array of vectors is shown in the dotted rectangle below. I put a -1 into a vector to indicate that this vector is empty.

The lattice I use in COMPUCELL3D is not exactly the one I use to store the limb surface. Its size is \((l - 1) \times (n - 1) \times (m - 1)\), and each pixel in it corresponds to the center of mass of one small cube in previous lattice, see Figure 4.4 for a 2-d illustration.

During CPM simulation, every time a pixel \((x_0, y_0, z_0)\) (in the second lattice) is picked, I go to the corresponding vector according to \((x_0, y_0)\). If the vector is empty, then the picked pixel is outside the limb and I discard it; otherwise, I find
Figure 4.4. Two lattices I use: the dotted-line one for the storage of the limb shape, and the solid-line one for the actual CPM simulation. I use the dotted lattice to approximate the limb surface. A grid point from the solid lattice is actually the center of mass of some pixel in the dotted lattice.

the hypothetical location of $z_0$ in the corresponding vector, which is sorted. There are three possible cases:

1. The inserted $z_0$ divides the vector into two parts of even sizes, then the picked pixel is outside the limb.

2. The inserted $z_0$ divides the vector into two parts of odd sizes, then the picked pixel is inside the limb.

   (a) If $z_0 - \frac{1}{2}$ or $z_0 + \frac{1}{2}$ is in the vector, the picked pixel is a boundary pixel.

   (b) Otherwise, the picked pixel is an interior pixel.

4.2 Dynamical Update of the Limb Surface

During the simulation, the limb shape grows gradually, so the data structure I use needs to be updated dynamically. A grid point can be added to or removed from the current limb.
4.2.1 Add a Grid Point to the Current Limb Boundary

The grid point \((x_0, y_0, z_0)\) to add should be next to a boundary point of the limb. After making sure that the point to be added is indeed outside the limb, I go to the corresponding vector according to \((x_0, y_0)\).

1. If the vector is empty, i.e. contains -1 only, remove -1 from it, and store \(z_0 - \frac{1}{2}\) and \(z_0 + \frac{1}{2}\) in the vector.

2. If \(z_0 - \frac{1}{2}\) is already in the vector, replace it by \(z_0 + \frac{1}{2}\).

3. If \(z_0 + \frac{1}{2}\) is already in the vector, replace it by \(z_0 - \frac{1}{2}\).

4. Insert \(z_0 - \frac{1}{2}\) and \(z_0 + \frac{1}{2}\) in the vector.

4.2.2 Remove a Grid Point from the Limb Boundary

Suppose the grid point to remove is \((x_0, y_0, z_0)\).

1. If both \(z_0 - \frac{1}{2}\) and \(z_0 + \frac{1}{2}\) are already in the vector, remove them both.

2. If only \(z_0 - \frac{1}{2}\) is in the vector, replace it by \(z_0 + \frac{1}{2}\).

3. If only \(z_0 + \frac{1}{2}\) is in the vector, replace it by \(z_0 - \frac{1}{2}\).

4. Insert \(z_0 - \frac{1}{2}\) and \(z_0 + \frac{1}{2}\) in the vector.

4.3 Simulation with Fixed Limb Shape

One source of the irregular limb shape can be obtained from the experiment image. I can use the actual limb shape in the experimental as the input for my simulation. For simplicity, I construct an irregular limb-like shape by hand, and use it as the limb surface for my simulation. I use a 2-d array of vectors to store this shape. In this simulation, the boundary is always fixed. cells evolve inside the limb.
Figure 4.5. Simulation with Fixed Limb Shape: the limb-like shape is stored in an 2-d array of vectors, and the CPM steps only take place inside the limb.

The simulation result is shown in Figure 4.5, and it proves that my data structure works.

4.4 Simulation with Varying Limb Shape

During the growth of a chicken limb, it is possible that the limb shape at stage 1 $S_1$ is not completely contained by the next limb shape at stage 2 $S_2$, see Figure 4.6. I cannot discard those cell lattice points outside $S_2$ and inside $S_1$. I need to find a way to “push” those cells into $S_2$.

Dillon and Othmer [17] suggested that the limb boundary gives a force on the fluid inside the limb. I can use this force to make the cell lattice points outside $S_2$ tend toward $S_2$.

The algorithm I propose is as follows. After a certain number of Potts steps are executed on the first shape $S_1$, the new shape $S_2$ is introduced. If $S_1$ is not contained completely by $S_2$, I run the simulation inside $S_1 \cup S_2$ (the union of the
Figure 4.6. $S_1$ and $S_2$ are two limb bud shapes, and $S_2$ is the next stage after $S_1$. $S_2$ does not contain $S_1$ completely. The force from the boundary of $S_1$ pushes those points toward $S_2$.

two domains). The difference is that when a boundary point $p$ of $S_1$ is chosen and $p$ is outside $S_2$, I simply remove $p$ from the domain (boundary pushes $p$ inward), instead of doing a Monte Carlo step with $p$ as selected pixel. If $p$ is a cell lattice point, the cell volume changes after $p$ is removed. The removing of $p$ will take effect when other pixels of the cell are selected later by making contribution to the volume energy. When $p$ is removed, there are certain things I need to take care of:

1. Update the data structure for $S_1$, the 2D array of vectors.

2. Update the volume for the corresponding cell, if $p$ is a cell pixel.

Figure 4.7 shows the simulation with different limb shapes. The current limb shape does not completely contain the previous one. I place a certain number of cells in the first shape, and turn the mitosis off, then I run the simulation and change the limb shape after certain steps. Using the above algorithm just described, when the shape changes, cells outside of the current shape can be pushed inwards gradually. The number of cells is the same for all the shapes, so cells are not lost.
To model more realistic limb growth with irregular shape, I create a series of irregular limb shapes, and also map the pattern for TGF\(_{\beta}\) from a regular domain to this irregular domain, see Figure 4.8. The pattern for TGF\(_{\beta}\) is generated by reaction-diffusion partial differential equations which I will talk about it in next chapter.

With the above irregular shapes, I run the CPM simulation, see Figure 4.9. We can see that cell movement and arrangement follow the shape and pattern in Figure 4.8.

4.5 Limb Growth in Irregular Shape and Implementation of AER

Besides taking the experimental images as the source of irregular limb shape, the limb shape can also be modeled. Dillon & Othmer [17] have proposed a model in which AER plays an important role. Here I just want to address how to maintain the storage for the limb shape when modeling the growth of the limb shape.

1. Initialize limb. Generate an irregular shape, for example a half sphere, as the
Figure 4.8. The irregular limb shape with the TGF$_{\beta}$ pattern inside.

Figure 4.9. The CPM simulation with irregular limb shapes.
starting limb shape. The surface of the irregular shape is stored by using a 2D array of vectors. Let call this data structure $S$.

2. Simplify AER as one point (I will implement a actual zone as AER later). Choose the point at the very distal tip as AER, so AER is on the boundary.

3. Carry out the normal Potts steps. Set the Mitosis doubling time for each cell to be a function with respect to the distance from the cell to AER. The function can be linear, or piecewise linear (so that the mitosis doubling time could be very long if the cell is beyond certain distance). The farther the cell, the longer the mitosis doubling time. For example,

$$
T_{\text{mitosis}} = \begin{cases} 
T_{\text{base}} + k \frac{d}{D}, & \text{if } d \leq D, \\
\infty \text{ (mitosis off)}, & \text{otherwise},
\end{cases}
$$

(4.1)

where $T_{\text{mitosis}}$ is the mitosis doubling time, $T_{\text{base}}$ is the base value for mitosis doubling time, $d$ is the distance from cell to AER (self-variable), $k$ is a constant for the magnitude and $D$ is a constant too, usually taken as the maximum value for $d$ in order to normalize the distance.

4. Keep track of the total density, i.e. keep track of the total number of pixels inside the limb and the total number of cell pixels.

5. If the density is greater than certain threshold, then allow a flip between a cell pixel with another pixel outside the limb (limb growth).

6. To keep track of AER,

(a) Treat AER as a special type of cell, only one pixel big always.

(b) During Potts simulation, when AER is chosen, it must be flipped with another boundary pixel (target pixel). If the target pixel is not on the
boundary, reject the flip. If the flip is acceptable (target pixel on the boundary too), the target pixel becomes AER, and the old AER becomes a medium (AER is not bigger than one pixel).

(c) If any flip doesn’t involve AER, but the flip will make AER become an interior pixel, reject the flip, make the target pixel as new AER and old AER as a medium. (One can also say it’s a double flip: first accept the potential flip, then flip AER with the target pixel again. The second flip will overwrite the first one)

4.6 Summary

In this chapter, I presented how I handle limb growth simulations with irregular domains, including simulations with fixed irregular domain and simulations with varying domains. I designed a special data structure to store the limb surface to suit the requirement of the Metropolis algorithm (Algorithm 6), so that it is efficient to decide whether a pixel is inside or outside the limb. While the limb is growing, the data structure needs to be updated accordingly.

In the previous two chapters, I have used skeletal patterns generated by a set of PDEs. I will show how to generate those patterns in the next chapter.
A typical reaction diffusion equation has the following form:

$$\frac{\partial u}{\partial t} = f + \nabla \cdot (D \nabla u), \quad (5.1)$$

where $D$ is the diffusivity.

Alan Turing first proposed a reaction diffusion (RD) model as the chemical basis of morphogenesis in one of the most important papers in theoretical biology [81].

Especially we are interested in an interacting system with two (or more) equations, for example,

$$\begin{cases} 
\frac{\partial u}{\partial t} = D_1 \Delta u + f(u, v), \\
\frac{\partial v}{\partial t} = D_2 \Delta v + g(u, v),
\end{cases} \quad (5.2)$$

where $u$ and $v$ are two unknown functions, and $f$ and $g$ are function with respect to $u$ and $v$, and $\Delta = \nabla^2$ is the Laplacian operator. For 2-d, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$; For 3-d, $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}$.

A reaction diffusion system exhibits diffusion-driven instability or Turing instability if the homogeneous steady state is stable to small perturbations in the absence of diffusion but unstable to small spatial perturbations when diffusion is present.
5.1 Fitz-Hugh Nagumo Model

Fitz-Hugh Nagumo kinetics is a good example of interesting patterns generated by Reaction-Diffusion mechanism, see [55]. It consists of a pair of reaction-diffusion equations:

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \Delta u - f(u) - v, \\
\frac{\partial v}{\partial t} &= D\Delta v + \epsilon(u,v)(ku - v),
\end{align*}
\]

(5.3)

with

\[
f(u) = \begin{cases} 
C_1 u, & u < u_1, \\
-C_2 u + a, & u_1 \leq u \leq u_2, \\
C_3(u - 1), & u > u_2.
\end{cases}
\]

and

\[
\epsilon(u,v) = \begin{cases} 
\epsilon_1, & u < u_1, \text{ and } v < v_1 \\
\epsilon_2, & u > u_2, \\
\epsilon_3, & \text{otherwise}.
\end{cases}
\]

To make the function $f(u)$ continuous,

\[
u_1 = \frac{a}{C_1 + C_2} \quad \text{and} \quad u_2 = \frac{a + C_3}{C_1 + C_2}.
\]

The stable state is the solution of

\[
\begin{align*}
-f(u) - v &= 0, \\
\epsilon(u,v)(ku - v) &= 0.
\end{align*}
\]

(5.4)

The initial condition is chosen to be a small perturbation around the stable state. A wave is generated at the beginning, and propagating from the left hand side to the right hand side. If nothing is done to the wave during the propagation, the wave will eventually disappear from the right side. If the wave is cut by half, then a spiral wave is generated. The spiral wave will eventually break up into a lot of smaller ones interacting with each other. Each has a relatively fixed center, see Figure 5.1. The
Figure 5.1. Fitz-Hugh Nagumo kinetics: The red area represents high concentration. (a) A initial wave is generated at the left-hand side. (b) The wave propagates toward right. (c) At certain stage, remove one third of the wave. (d) the left wave becomes a spiral wave. (e) The spiral wave continues to break into smaller spiral waves.

wave breaks up due to the lateral instability. From the last picture in Figure 5.1, one can see that the wave front does not propagate at the same speed. The wave breaks up at the part which has slower speed.

5.2 Schnakenberg Equations

This section describes how a reaction-diffusion model generates patterns. These patterns are used to guide the bone formation (chondrogenesis) in the limb growth simulation.

The dimensionless Schnakenberg kinetics is given by

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \gamma(a - u + u^2v) + \nabla^2 u = \gamma f(u, v) + \nabla^2 u, \\
\frac{\partial v}{\partial t} &= \gamma(b - u^2v) + d\nabla^2 u = \gamma g(u, v) + d\nabla^2 v.
\end{align*}
\] (5.5)

1. $\gamma^\frac{1}{2}$ is proportional to the linear size of the special domain in one dimension.

In two dimensions $\gamma$ is proportional to the area.
2. $\gamma$ represents the relative strength of the reaction terms. This means, for example, that an increase in $\gamma$ may represent an increase in activity of some rate-limiting step in the reaction sequence.

3. An increase in $\gamma$ can also be thought of as equivalent to a decrease in the diffusion coefficient ratio $d$.

A reaction diffusion system exhibits diffusion-driven instability or Turing instability if the homogeneous steady is stable to small perturbations in the absence of diffusion but unstable to small spatial perturbations when diffusion is present.

Whether or not the system is capable of generating Turing-type spatial patterns crucially depends on the reaction kinetics $f$ and $g$, and the value of $\gamma$ and $d$.

5.2.1 Stability in absence of diffusion

With no spatial variation $u$ and $v$ satisfy

$$\begin{align*}
\frac{\partial u}{\partial t} &= \gamma f(u, v), \\
\frac{\partial v}{\partial t} &= \gamma g(u, v).
\end{align*}$$

(5.6)

Linearizing about the steady state $(u_0, v_0)$, we set

$$w = \begin{pmatrix} u - u_0 \\ v - v_0 \end{pmatrix}$$

and Schnakenberg equations become, for $|W|$ small,

$$w_t = \gamma Aw,$$

(5.7)

where

$$A = \begin{pmatrix} f_u & f_v \\ g_u & g_v \end{pmatrix}_{u_0,v_0},$$

(5.8)
We look for solutions in the forms

\[ w \propto e^{\lambda t} \]

where \( \lambda \) is the eigenvalue. The steady state \( w = 0 \) is linearly stable if \( \text{Re}\lambda < 0 \) since in this case the perturbation \( w \to 0 \) as \( t \to \infty \).

\[
|\gamma A - \lambda I| = \begin{vmatrix}
\gamma f_u - \lambda & \gamma f_v \\
\gamma g_u & \gamma g_v - \lambda
\end{vmatrix} = 0
\]

\[ \Rightarrow \lambda^2 - \gamma(f_u + g_v)\lambda + \gamma^2(f_u g_v - f_v g_u) = 0, \]

so

\[ \lambda_1, \lambda_2 = \frac{1}{2} \gamma \{ (f_u + g_v) \pm \sqrt{(f_u + g_v)^2 - 4(f_u g_v - f_v g_u)} \}^{1/2}. \]

Linear stability, that is \( \text{Re}\lambda < 0 \), is guaranteed if

\[ \text{tr} A = f_u + g_v < 0, |A| = f_u g_v - f_v g_u > 0. \]

### 5.2.2 Instability when diffusion is present

Now consider the full Schnakenberg equations and again linearize about the steady state to get

\[ w_t = \gamma A w + D \nabla^2 w, \tag{5.9} \]

where

\[ D = \begin{pmatrix}
1 & 0 \\
0 & d
\end{pmatrix}. \]

To solve it we first define \( W(t) \) to be the time independent solution of the spatial eigenvalue problem defined by

\[ \nabla^2 W + k^2 W = 0, (n \cdot \nabla)W = 0 \text{ for } r \text{ on } \partial B, \]

where \( k \) is the eigenvalue, which is also called wavenumber.
Let $W_k(r)$ be the eigenfunction corresponding to the wavenumber $k$. Each eigenfunction $W(k)$ satisfies zero flux boundary conditions. Because the problem is linear we can now look for solution $w(r,t)$ in the form

$$w(r,t) = \sum_{\text{eigenvalue } k} c_k e^{\lambda t} W_k(r),$$

where the constants $c_k$ are determined by a Fourier expansion of the initial conditions in terms of $W_k(r)$. Substituting this form into the linearized Schnakenberg equations, we get, for each $k$,

$$\lambda W_k = \gamma A W_k + D \nabla^2 W_k = \gamma A W_k - D k^2 W_k.$$

We require nontrivial solutions for $W_k$ so the $\lambda$ are determined by the roots of the characteristic polynomial

$$|\lambda I - \gamma A + Dk^2| = 0.$$

$$\lambda^2 + \lambda[k^2(1 + d) - \gamma(f_u + g_v)] + h(k^2) = 0,$$

$$h(k^2) = dk^4 - \gamma(d^2 + g_v)k^2 + \gamma^2|a|.$$

For the steady state to be unstable to spatial disturbances we require Re$\lambda(k) > 0$ for some $k \neq 0$. If the stability in absence of diffusion is satisfied, from last section $(f_u + g_v) < 0$, so the only way Re$\lambda(k) > 0$ can be positive is if $h(k^2) < 0$ for some $k$. It’s equivalent to say the minimum $h_{\text{min}}$ must be negative

$$h_{\text{min}} = \gamma^2 |\gamma| - \frac{(df_u - g_v)^2}{4d},$$

$$k^2 = k_m^2 = \gamma \frac{df_u + g_v}{2d}.$$
Thus the condition that $h(k^2) < 0$ for some $k^2 \neq 0$ is

$$\frac{(df_u + g_v)^2}{4d} > |A|.$$ 

The range of $k$ which satisfies $h(k^2) < 0$ is

$$k_1^2 = \gamma \left\{ \frac{(df_u + g_v)^2}{2d} - \frac{(df_u + g_v)^2 - 4d|A|}{4d} \right\}^{1/2} < k^2$$

$$< \gamma \left\{ \frac{(df_u + g_v)^2}{2d} + \frac{(df_u + g_v)^2 - 4d|A|}{4d} \right\}^{1/2} = k_2^2. \quad (5.10)$$

Since I assume no cell activities happen in the frozen zone, the domain of Schnakenberg kinetics is defined to be the active zone only. It moves up with the active zone. Some solution analysis has been done in [62]. The following is the simulation result.

5.2.3 Two Dimensional Analysis and Simulations

It turns out $\gamma$ has great control over the pattern generated. Different $\gamma$ values correspond to different number of high concentration areas, see Figure 5.2. Dark area represents high concentration; White area corresponds to low concentration.

Table 5.1 shows the parameters I used for 2-d simulation. The width of the whole domain is 1.0, the height is 3.0. Finally the height of the active zone is 0.5. If $\lambda$ is chosen to be 60.0, then I will have the one-bone pattern, 350.0 for the two-bone pattern and 500.0 for the three-bone pattern.

5.2.4 Three Dimensional Analysis and Simulations

It is significantly more difficult to obtain desirable patterns in 3-d. I show the pattern that has been obtained so far, and explain the difficulty later.

Figure 5.3 shows the resulted pattern. Similar to the 2-d case, there are three different stages with 1, 2 and 3 digits respectively.
TABLE 5.1

PARAMETER VALUES FOR THE 2-D SCHNAKENBERG KINETICS.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.017</td>
</tr>
<tr>
<td>b</td>
<td>1.015</td>
</tr>
<tr>
<td>d</td>
<td>7.1</td>
</tr>
<tr>
<td>Domain width</td>
<td>1.0</td>
</tr>
<tr>
<td>Domain height</td>
<td>3.0</td>
</tr>
<tr>
<td>Active zone height</td>
<td>0.5</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>60.0 for one stripe</td>
</tr>
<tr>
<td></td>
<td>350.0 for two stripes</td>
</tr>
<tr>
<td></td>
<td>500.0 for three stripes</td>
</tr>
</tbody>
</table>

Figure 5.2. 2-d Schnakenberg Kinetics: $\gamma$ is a step function which takes values 60.0, 350.0 and 500.0. Different value corresponds to different number of digits, 60.0–1 digit, 350.0–2 digits and 500.0–3 digits.
Figure 5.3. 3-d Schnakenberg Kinetics: Besides applying different values for $\gamma$, I also need to vary the diffusion in $x$ and $y$ directions to get different number of digits. Compared with 2-d case, 3-d is much more sensitive to the parameters.

If I take the some cross-sections of this pattern, I will have three different patterns shown in Figure 5.4. Each cross-section has 1, 2 or 3 “dots” respectively.

Those “dots” are very easy to evolve into stripes, vertical ones (Figure 5.6) or horizontal ones (Figure 5.5). Numerically, stripes are more stable than dots.

To make the “dots” stable, one has to give the system more freedom by applying different diffusivities in different directions. By adjusting the diffusivities carefully during the simulation, I are able to keep the dots stable eventually. Equations (5.11) are the modified Schnakenberg Equations for 3-d simulation.

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \gamma(a - u + u^2v) + (d_{u,x} \frac{\partial^2 u}{\partial x^2} + d_{u,y} \frac{\partial^2 u}{\partial y^2}), \\
\frac{\partial v}{\partial t} &= \gamma(b - u^2v) + (d_{v,x} \frac{\partial^2 v}{\partial x^2} + d_{v,y} \frac{\partial^2 v}{\partial y^2}).
\end{align*}
\]

(5.11)

Table 5.2 shows the parameters I used for 3-d simulation:
Figure 5.4. Cross sections of the 3-d pattern: I take 3 different cross-sections of Figure 5.3, each cross-section has a spot-like pattern. The number of spots corresponds to the number of digits.

Figure 5.5. Horizontal stripe: Instead of getting a spot-like pattern, the system is more likely to evolve into a stripe-like pattern. This is an example of a horizontal stripe. Mathematically, the stripe-like pattern is more stable than the spot-like pattern.
Figure 5.6. Vertical stripes: Sometimes I have several stripes and all of them are vertical.

### TABLE 5.2

PARAMETER VALUES FOR THE 3-D SCHNAKENBERG KINETICS.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0.017</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>1.015</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain width</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain length</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Domain height</td>
<td>3.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active zone height</td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>One bone</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>300.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,x}$</td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,y}$</td>
<td>1.75</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,x}$</td>
<td>1.0×8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,y}$</td>
<td>2.0×8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Two bones</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>520.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,x}$</td>
<td>0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,y}$</td>
<td>1.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,x}$</td>
<td>0.9×8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,y}$</td>
<td>1.9×8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Three bones</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\gamma$</td>
<td>650.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,x}$</td>
<td>0.82</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{u,y}$</td>
<td>1.77</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,x}$</td>
<td>0.9×8.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$d_{v,y}$</td>
<td>1.9×8.5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.3 A More Realistic Model

My collaborators proposed a more realistic model [35], see Equation 5.12.

\[
\begin{align*}
\frac{\partial c_a}{\partial t} &= \gamma [(J_a^1 R_1 + J_a(c_a)\beta(c_a)R_2) - k_a c_a c_i] + \nabla^2 c_a, \\
\frac{\partial c_i}{\partial t} &= \gamma [J_i(c_a)\beta(c_a)R_2 - k_i c_a c_i] + d\nabla^2 c_i.
\end{align*}
\]

(5.12)

where \( c_a \) and \( c_i \) represent the activator (TGF\(_\beta\)) and inhibitor respectively. \( R_1 \) cells release TGF\(_\beta\) at rate \( J_a^1 \). \( R_2 \) cells release TGF\(_\beta\) at rate \( J_a(c_a)\beta(c_a) \) and the inhibitor at rate \( J_i(c_a)\beta(c_a) \). The term \( k_a c_a c_i \) and \( k_i c_a c_i \) assume that the inhibitor binds to the TGF\(_\beta\) forming a non-reactive complex.

To obtain a desirable pattern, I modify the system for stability consideration. I apply diffusivities in \( x \) and \( y \) directions so that I can prevent a “spot” pattern from becoming a “stripe” pattern.

\[
\begin{align*}
J_a(x) &= \frac{8.0x^2}{6.25 + x^2} \quad (5.13) \\
J_i(x) &= \frac{8.6x^2}{6.25 + x^2} \quad (5.14) \\
\beta(x) &= \frac{0.745146x}{x + 1.92248} \quad (5.15)
\end{align*}
\]

where \( c_{as} \) and \( c_{is} \) represent the stable state (state without diffusion). The two terms \( b_a(c_a - c_{as})^3 \) and \( b_i(c_i - c_{is})^3 \) help the system to obtain the Turing instability.

Figure 5.7 is the pattern obtained for 1, 2 and 3 digits. Table 5.3 are the parameter values used for this pattern. The diffusion coefficients in \( x \) and \( y \) directions are very important and difficult to adjust.
Figure 5.7. 3-d simulation for Equations (5.16): Similar to the 3-d modified Schnakenberg Equations (5.11), I apply different values for $\gamma$ and adjust diffusions in $x$ and $y$ directions accordingly to get expected patterns: 1, 2 and 3 digits.
### TABLE 5.3

PARAMETER VALUES FOR 1-2-3 PATTERN.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain Length</td>
<td>$2\pi$</td>
</tr>
<tr>
<td>Domain Width</td>
<td>$\frac{6\pi}{7}$</td>
</tr>
<tr>
<td>Domain Height</td>
<td>$8\pi$</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>4.34319</td>
</tr>
<tr>
<td>$J^i_0$</td>
<td>0.04</td>
</tr>
<tr>
<td>$k_u$</td>
<td>1.0</td>
</tr>
<tr>
<td>$k_i$</td>
<td>1.0</td>
</tr>
<tr>
<td>$R_1$</td>
<td>2.0</td>
</tr>
<tr>
<td>$R_2$</td>
<td>2.0</td>
</tr>
<tr>
<td>$d$</td>
<td>5.0</td>
</tr>
<tr>
<td>$b_u$</td>
<td>0.02</td>
</tr>
<tr>
<td>$b_i$</td>
<td>-0.6</td>
</tr>
<tr>
<td>$c_{as}$</td>
<td>1.32494</td>
</tr>
<tr>
<td>$c_{ia}$</td>
<td>0.86545</td>
</tr>
<tr>
<td>$\Delta x$</td>
<td>$\frac{\pi}{35}$</td>
</tr>
<tr>
<td>$\Delta t$</td>
<td>0.00002</td>
</tr>
<tr>
<td>$d_{ax} = d_{ix}$</td>
<td>1.0 if $t &lt; 30$ (1-spot), 0.15 if $t &lt; 60$ (2-spot), otherwise (3-spot).</td>
</tr>
<tr>
<td>$d_{ay} = d_{iy}$</td>
<td>0.15</td>
</tr>
<tr>
<td>velocity</td>
<td>$\frac{4\pi}{35}$ if $t &lt; 30$, $\frac{\pi}{35}$ otherwise.</td>
</tr>
</tbody>
</table>
Figure 5.8. An example that I can get 4-digit pattern when applying different value $\gamma$.

### TABLE 5.4

PARAMETER VALUES FOR 1-2-4 PATTERN.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_{ax} = d_{ix}$</td>
<td>1.0 if $t &lt; 60$ (1-spot), $\frac{1}{4}$ if $t &lt; 120$ (2-spot), $\frac{1}{11}$ otherwise (3-spot).</td>
</tr>
<tr>
<td>$d_{ay} = d_{iy}$</td>
<td>0.17</td>
</tr>
<tr>
<td>velocity</td>
<td>$\frac{2\pi}{36}$</td>
</tr>
</tbody>
</table>

If I use different diffusion coefficients, I will have a different pattern. Figure 5.7 is the pattern obtained for 1, 2 and 4 digits. See Table 5.4 for the changes made to the parameters.

5.4 Summary

Reaction-diffusion equations are capable of producing skeletal patterns and explaining how the skeleton forms inside limb. I successfully produce several patterns and use them for limb growth simulation.
In the next chapter, I will extend the classical Finite Difference Method (FDM) from regular grids to irregular grids. I call the extended method the directional FDM.
CHAPTER 6

FINITE DIFFERENCE METHOD ON A IRREGULAR GRID

When the avian limb growth simulation uses a regular domain, the method I use to solve the partial differential equations (PDEs) is the finite difference method (FDM). Eventually, I need to do the simulation in a more realistic irregular domain which is much more complicated. Solving PDEs in an irregular domain is more challenging. Since I cannot use a regular grid for the discretization. Current FDM can only be used on a regular grid (or lattice). There are methods, such as finite element method and finite volume method, which do not require a regular grid, but they are complicated to be implemented. Because of the simplicity of FDM (easy to understand and easy to implement), it is desirable to extend it to an irregular grid, for example a mesh resulting from triangulation. This chapter describes how to do this precisely.

6.1 Finite Difference Method

FDM is a commonly used numerical method for solving PDEs. One can find it in most numerical analysis books, see [74]. The basic idea of FDM is to discretize the solution domain using a grid of quadrilaterals (for 2-d) or cubes (for 3-d), and find a way to approximate the partial derivatives (first order, second order or even higher orders) of the unknown function. At each grid point, every term in the partial differential is replaced by a difference formula which may include values of
the unknown function at neighboring of grid points. By substituting the difference formulae into the PDE, a difference equation is obtained. Here I want to extend FDM to an irregular grid, particularly a triangular grid.

6.2 Irregular Grid–Discretization of Domain

The commonly used methods to approximate the derivatives are backward difference, forward difference and central difference. All these methods can only be implemented on a rectangular lattice. One of the major limitations of regular FDM is that rectangular lattice has to be used, and it may be very difficult to use a rectangular lattice to approximate an arbitrary shape, for example a disc. To approximate an arbitrary shape, the most common discretization is triangulation – a way to divide the domain into small triangles. The method I propose here is to extend the regular FDM to any irregular grid.

Figure 6.1 shows an example of triangulation. Triangulation is the division of a surface or plane polygon into a set of triangles, usually with the restriction that each triangle side is entirely shared by two adjacent triangles.
6.3 Approximation of the First Order Derivatives on a Irregular Grid

Let $G$ be a grid on certain domain $D$. Suppose $p_0$ is a grid point of $G$, and it has $n$ neighbors $p_{01}$, $p_{02}$, $p_{03}$, \ldots, $p_{0n}$, ordered counter clockwise, see Figure 6.2. Let denote the vector from $p_0$ to $p_{0i}$ by $\vec{v}_{0i}$ ($1 \leq i \leq n$). Throughout this chapter, I will denote $\frac{\partial u}{\partial x}(p_0)$ by $\frac{\partial u_0}{\partial x}$, $\frac{\partial u}{\partial x}(p_{0i})$ by $\frac{\partial u_{0i}}{\partial x}$, $p_0$’s $i$-th neighbor by $p_{0i}$ and $p_{0i}$’s $j$-th neighbor by $p_{0ij}$.

Suppose $u$ is an unknown function with respect to $x$ and $y$. The approximations of $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$ using DFDM are

\[
\frac{\partial u_0}{\partial x} = \sum_{i=1}^{n} B_{0i} (u_i - u_0) = \sum_{i=1}^{n} B_{0i} u_i - (\sum_{i=1}^{n} B_{0i}) u_0 \tag{6.1}
\]

\[
\frac{\partial u_0}{\partial y} = \sum_{i=1}^{n} A_{0i} (u_i - u_0) = \sum_{i=1}^{n} A_{0i} u_i - (\sum_{i=1}^{n} A_{0i}) u_0. \tag{6.2}
\]
and

\[ D_0 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (a_{0i}b_{0j} - a_{0j}b_{0i})^2, \]  
(6.3)

\[ A_{0i} = \frac{1}{D_0} \sum_{j=1}^{n} (a_{0j}b_{0i} - a_{0i}b_{0j})a_{0j}, \]  
(6.4)

\[ B_{0i} = \frac{1}{D_0} \sum_{j=1}^{n} (a_{0i}b_{0j} - a_{0j}b_{0i})b_{0j}. \]  
(6.5)

See Appendix B for details of the derivation.

6.4 Approximation of the Second Order Derivatives on Irregular Grid

There are two different ways to approximate the second order derivatives. The first way is to use the fact that the second order derivatives are the derivatives of the first order derivatives; The second way is to consider the Taylor’s series approximation.

6.4.1 Approach 1: Derivatives of First Order Derivatives

The second order derivatives of \( u(x, y) \) are the first order derivatives of \( \frac{\partial u}{\partial x} \) and \( \frac{\partial u}{\partial y} \). I can use the above method repeatedly to approximate the second order derivatives of \( u(x, y) \). Therefore, the first order derivatives of \( u(x, y) \) at \( p_0 \) depend on \( p_0 \) and its first neighbors; The second order derivatives of \( u(x, y) \) at \( p_0 \) depend on \( p_0 \), its first neighbors and its second neighbors. The formulae for the second order derivatives are as follows.

\[
\frac{\partial^2 u_0}{\partial x^2} = \sum_{i=1}^{n} \sum_{j=1}^{m_i} B_{0i}B_{0ij}u_{0ij} - \sum_{i=1}^{n} u_{0i}(B_{0i}(\sum_{j=1}^{m_i} B_{0ij} + \sum_{j=1}^{n} B_{0j})) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} B_{0i}B_{0j};
\]  
(6.6)

where \( m_i \) means the number of neighbors of \( u_{0i} \), \( u_{0ij} \) represents the \( j \)-th neighbor.
of $u_{0i}$, and $D_{0i}$, $A_{0i}$ and $B_{0i}$ are as above. Similarly,

$$\frac{\partial^2 u_0}{\partial y^2} = \sum_{i=1}^{n} \sum_{j=1}^{n_i} A_{0i} A_{0ij} u_{0ij} - \sum_{i=1}^{n} u_{0i} (A_{0i} (\sum_{j=1}^{n_i} A_{0ij} + \sum_{j=1}^{n} A_{0j})) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{0i} A_{0j}. \quad (6.7)$$

$$\frac{\partial^2 u_0}{\partial x \partial y} = \sum_{i=1}^{n} \sum_{j=1}^{n_i} A_{0i} B_{0ij} u_{0ij} - \sum_{i=1}^{n} u_{0i} (A_{0i} (\sum_{j=1}^{n_i} B_{0ij} + \sum_{j=1}^{n} B_{0j})) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{0i} B_{0j}. \quad (6.8)$$

Finally,

$$\frac{\partial^2 u_0}{\partial y \partial x} = \sum_{i=1}^{n} \sum_{j=1}^{n_i} B_{0i} A_{0ij} u_{0ij} - \sum_{i=1}^{n} u_{0i} (B_{0i} (\sum_{j=1}^{n_i} A_{0ij} + \sum_{j=1}^{n} B_{0j})) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} B_{0i} A_{0j}. \quad (6.9)$$

See Appendix C for the details of the derivation.

One can see that the formulae do not look very neat. Moreover, by doing it in this way, the second order derivatives have to depend on the second neighbors. The farther the neighbors, the bigger the discretization error. Actually I can develop a formula for the second order derivatives so that they depend on the point itself and its first neighbors only. Therefore, the bandwidth of the coefficient matrix is reduced.

6.4.2 Approach 2: Taylor’s Formula

In this section, again only the final formulae are given. See Appendix D for details of the derivation.
Let us introduce some notations first, 

\[ L_{0i} = 2((a_{0i})^2 - B_{0i} \sum_{j=1}^{n} (a_{0j})^3 - A_{0i} \sum_{j=1}^{n} (a_{0j})^2 b_{0j}), \quad (6.10) \]

\[ M_{0i} = 4(a_{0i} b_{0i} - B_{0i} \sum_{j=1}^{n} (a_{0j})^2 b_{0j} - A_{0i} \sum_{j=1}^{n} a_{0j} (b_{0j})^2), \quad (6.11) \]

\[ N_{0i} = 2((b_{0i})^2 - B_{0i} \sum_{j=1}^{n} a_{0j} (b_{0j})^2 - A_{0i} \sum_{j=1}^{n} (b_{0j})^3), \quad (6.12) \]

\[
\begin{pmatrix}
 a & b & c \\
 b & d & e \\
 c & e & f
\end{pmatrix}
= \begin{pmatrix}
 \sum_1^n (a_{0i})^4 & \sum_1^n 2(a_{0i})^3 b_{0i} & \sum_1^n (a_{0i} b_{0i})^2 \\
 \sum_1^n 2(a_{0i})^3 b_{0i} & \sum_1^n (2a_{0i} b_{0i})^2 & \sum_1^n 2a_{0i} (b_{0i})^3 \\
 \sum_1^n (a_{0i} b_{0i})^2 & \sum_1^n 2a_{0i} (b_{0i})^3 & \sum_1^n (b_{0i})^4
\end{pmatrix},
\]

and

\[
\begin{pmatrix}
 r \\
 s \\
 t
\end{pmatrix}
= \begin{pmatrix}
 2(\sum_1^n ((a_{0i})^2 - B_{0i} \sum_1^n (a_{0j})^3 - A_{0i} \sum_1^n (a_{0j})^2 b_{0j})(u_{0i} - u_{0})) \\
 4(\sum_1^n (a_{0i} b_{0i} - B_{0i} \sum_1^n (a_{0j})^2 b_{0j} - A_{0i} \sum_1^n a_{0j} (b_{0j})^2)(u_{0i} - u_{0})) \\
 2(\sum_1^n ((b_{0i})^2 - B_{0i} \sum_1^n a_{0j} (b_{0j})^2 - A_{0i} \sum_1^n (b_{0j})^3)(u_{0i} - u_{0}))
\end{pmatrix}.
\]

The formulae to approximate the second order derivatives are as follows.

\[
\frac{\partial^2 u_0}{\partial y^2} = \sum_{i=1}^{n}(C_{r,y^2} L_{0i} + C_{s,y^2} M_{0i} + C_{t,y^2} N_{0i})(u_{0i} - u_{0}). \quad (6.13)
\]

where

\[
C_{r,y^2} = \frac{b(ae - bc) - c(ad - b^2)}{(af - c^2)(ad - b^2) - (ae - bc)^2}, \quad (6.14)
\]

\[
C_{s,y^2} = -\frac{a(ae - bc)}{(af - c^2)(ad - b^2) - (ae - bc)^2}, \quad (6.15)
\]

\[
C_{t,y^2} = \frac{a(ad - b^2)}{(af - c^2)(ad - b^2) - (ae - bc)^2}. \quad (6.16)
\]

By symmetry, I can write down the solution for \(\frac{\partial^2}{\partial x^2}\) and \(\frac{\partial^2}{\partial x \partial y}\):

\[
\frac{\partial^2 u_0}{\partial x^2} = \sum_{i=1}^{n}(C_{r,y^2} L_{0i} + C_{s,y^2} M_{0i} + C_{t,y^2} N_{0i})(u_{0i} - u_{0}), \quad (6.17)
\]
where

\[ C_{t,x^2} = \frac{f(eb - fc) - c(ed - f^2)}{(ea - c^2)(ed - f^2) - (eb - fc)^2}, \quad (6.18) \]

\[ C_{s,x^2} = -\frac{e(eb - fc)}{(ea - c^2)(ed - f^2) - (eb - fc)^2}, \quad (6.19) \]

\[ C_{r,x^2} = \frac{c(ed - f^2)}{(ea - c^2)(ed - f^2) - (eb - fc)^2}. \quad (6.20) \]

\[ \frac{\partial^2 u_0}{\partial x \partial y} = \sum_1^n (C_{r,y} L_0 + C_{s,y} M_0 + C_{t,y} N_0) (u_{0i} - u_0), \quad (6.21) \]

where

\[ C_{r,xy} = \frac{c(ae - cb) - b(af - c^2)}{(ad - b^2)(af - c^2) - (ae - cb)^2}, \quad (6.22) \]

\[ C_{t,xy} = -\frac{a(ae - cb)}{(ad - b^2)(af - c^2) - (ae - cb)^2}, \quad (6.23) \]

\[ C_{s,xy} = \frac{a(af - c^2)}{(ad - b^2)(af - c^2) - (ae - cb)^2}. \quad (6.24) \]

### 6.5 Boundary Conditions

There are three major types of boundary conditions:

1. *Dirichlet Condition*
   \[ u = \phi \text{ on } \partial \Gamma, \]

2. *Neumann Condition*
   \[ \frac{\partial u}{\partial n} = \gamma \text{ on } \partial \Gamma, \]

3. *Cauchy Condition*
   \[ \frac{\partial u}{\partial n} + \alpha u = \beta \text{ on } \partial \Gamma, \]

where \( \Gamma \) is the boundary of the domain, and \( \phi, \gamma, \alpha \) and \( \beta \) are given functions. Dirichlet condition is easy to implement. I will focus on the last two.
Figure 6.3. An ghost point for Neumann boundary condition: Point \( p_4 \) doesn’t exist in the triangulation, add it for the boundary condition implementation. Function value at \( p_4 \) is defined according to the boundary conditions.

6.5.1 Neumann Boundary Condition

Suppose \( p_0 \) is a grid point on the boundary, and it has \( n \) neighbors \( p_1, p_2, p_3, \ldots, p_n \), ordered counter clockwise. Let the vectors from \( p_0 \) to \( p_i \) be denoted by \( \vec{v}_i \) (1 \( \leq i \leq n \)). For simplicity, assume the Neumann boundary condition is

\[
\frac{\partial u}{\partial n} = 0 \text{ on } \partial \Omega.
\]

I want to add a neighbor (let us call it a ghost point) to \( p_0 \), so that if I use the method above to calculate \( \frac{\partial u}{\partial n}|_{p_0} \), the result is 0.

Let us call \( p_0 \)'s new neighbor \( p_{n+1} \). \( \vec{v}_{n+1} \), the vector from \( p_0 \) to \( p_{n+1} \), is normal to the domain (pointing outward), see Figure 6.3. Denote:

\[
u_i = u(p_i), \ 0 \leq i \leq n + 1.
\]

\[
\vec{v}_i = \overrightarrow{p_0 p_i} = (a_i b_i), \ 1 \leq i \leq n + 1.
\]

Among all the notations, \( u_{n+1} \) is the only unknown.
The final formula is

\[ u_{n+1} = u_0 - \frac{\sum_{i=1}^{n} (a_{n+1}B_i + b_{n+1}A_i)(u_{0i} - u_0)}{a_{n+1}B_{n+1} + b_{n+1}A_{n+1}}, \]  
\text{(6.25)}

where

\[ D = \sum_{i=1}^{n+1} \sum_{j=i+1}^{n+1} (a_i b_j - a_j b_i)^2, \]  
\text{(6.26)}

\[ A_i = \frac{1}{D} \sum_{j=1}^{n+1} (a_j b_i - a_i b_j)a_j, \]  
\text{(6.27)}

\[ B_i = \frac{1}{D} \sum_{j=1}^{n+1} (a_i b_j - a_j b_i)b_j. \]  
\text{(6.28)}

See Appendix E for details of the derivation.

For the general Neumann boundary condition

\[ \frac{\partial u}{\partial n} = \phi \text{ on } \partial \Omega, \]

we have

\[ u(p_{n+1}) = u_0 + \frac{\phi - \sum_{i=1}^{n} (a_{n+1}B_i + b_{n+1}A_i)(u_{0i} - u_0)}{a_{n+1}B_{n+1} + b_{n+1}A_{n+1}}, \]  
\text{(6.29)}

6.5.2 Cauchy Boundary Condition

The Cauchy boundary condition is

\[ \frac{\partial u}{\partial n} + \alpha u = \beta \text{ on } \partial \Omega. \]

Similar to the Neumann boundary condition, I add a ghost grid point \( p_{n+1}(a_{n+1}, b_{n+1}) \) neighboring \( p_0 \). To guarantee the boundary condition, we must have:

\[ a_{n+1} \frac{\partial u}{\partial x} + b_{n+1} \frac{\partial u}{\partial y} + \alpha u = \beta. \]
Solving $u_{n+1}$,

$$u(p_{n+1}) = u_0 + \frac{(\beta - \alpha u_0) - \sum_{i=1}^{n}(a_{n+1}B_i + b_{n+1}A_i)(u_0 - u_i)}{a_{n+1}B_{n+1} + b_{n+1}A_{n+1}}.$$ (6.30)

See Appendix E for more details.

6.6 An Example: Poisson Equation

In this section, I will focus on a specific example and find out how to evaluate the coefficient matrix.

The Poisson equation is

$$u_{xx} + u_{yy} = f(x, y), \text{ in } G,$$ (6.31)

subject to the boundary condition

$$\frac{\partial u}{\partial n} = 0, \text{ on } \partial G.$$ (6.32)

The value of $u(x, y)$ on each grid point is unknown. Suppose the number of grid points are $n$, $u_i$ represents the value of $u(x, y)$ at the $i$th grid point $p_i$ ($1 \leq i \leq n$). To find out $u_i$, I construct a linear system $Mx = d$, where $x = (u_1 \ u_2 \ \cdots \ u_n)^T$, and then solve it. The question is how to evaluate the coefficient matrix $M$ and the vector $d$. The boundary condition is taken care of by adding a ghost neighbor to each boundary grid point as discussed above.
\[ d = \begin{pmatrix} f(p_1) \\ f(p_2) \\ \vdots \\ f(p_n) \end{pmatrix}, \]

\[ M = (m_{ij}). \] To evaluate \( m_{ij} \),

- \( m_{ij} = 0 \) for \( 1 \leq i, j \leq n \),

- For each \( 1 \leq i \leq n \),

\[ m_{ij} = m_{ij} + \text{coefficients before } u_j \text{ in Equations (3.17) and (3.18)}, \]

\( u_j \) means the value of unknown function at the \( j \)-th grid point. It might be represented by a different notation in Equations (3.17) and (3.18).

### 6.7 Error Analysis

To extend FDM to irregular grids, I use the Taylor’s expansion. For the first order derivatives, I take the first two terms of the Taylor’s expansion, so the discretization error \( d_{\text{error}} = \mathcal{O}(h^2) \), where \( h \) is the distance to the farthest neighbor. For the first order derivatives, the discretization error \( d_{\text{error}} = \mathcal{O}(h^2) \).

### 6.8 Summary

In this chapter, I show how to extend FDM to irregular grids. Specifically I give the formula to approximate the first and second order derivatives. Higher order derivatives can be approximated similarly.
7.1 Summary of Results

In the first part of this dissertation, I present a multiscale computational model for morphogenesis, specifically the avian limb growth simulation. The simulation results include 2-d and 3-d, regular domain and irregular domain cases.

**CompuCell3D** is the software I use for morphogenesis simulation. **CompuCell3D** has a multi-model framework. The genetic regulation is modeled by a combination of a rule-based state automaton and a set of PDEs; the generic cellular mechanisms include cell adhesion, haptotaxis and chemotaxis. The simulation includes cell differentiation, cell growth, mitosis, secretion of morphogens, chemotaxis and haptotaxis.

The reaction-diffusion mechanism is used to model TGF$\beta$ concentration. A reaction-diffusion system can describe the skeleton patterning in an avian limb. For the Schnakenberg kinetics, I have produced bone-like patterns for both 2-d and 3-d cases. I already know how to adjust the parameters to get the pattern I want. Another more realistic reaction-diffusion system is also studied. I produce similar patterns with it for 3-d limb growth simulations too. To solve PDEs on an irregular grid, an extended FDM has been developed. FDM is used to solve partial differential equations, and it usually can only be implemented on rectangular lattice. I have found a way to extend FDM to an irregular grid.
7.2 Future Work

One of our next biggest challenges is to find out how to generate a realistic limb shape. In Chapter 4, I have presented a method to deal with an irregular shape once it is generated and simulation results with irregular domains. The question is how to generate the irregular shape, i.e. how the irregular limb shape is produced. Limb growth is a complicated biological process. How a limb determines its shape is not completely clear to us yet. There are two ways to obtain the actual shape of the limb, either by simulation, or by experiment.

Dillon & Othmer [17] have proposed a model to simulate the limb shape formation. The model is

\[
\frac{\partial u}{\partial \tau} + (u \cdot \nabla)u = -\nabla p + R^{-1}(\nabla^2 u + \frac{1}{3}\nabla S) + F \tag{7.1}
\]

\[
\frac{\partial c}{\partial \tau} + Sc + u \cdot \nabla c = D\nabla^2 c + G(c) \tag{7.2}
\]

\[
\frac{\partial X}{\partial \tau} = u(X(\sigma, \tau), \tau) \tag{7.3}
\]

\[
\nabla \cdot u = S(c) \tag{7.4}
\]

\[
F(x, \tau) = \int f(\sigma, \tau)\delta(x - X(\sigma, \tau))d\sigma \tag{7.5}
\]

where \( u \) is fluid velocity, \( c = (c_1, c_2) \) represents two morphogens, \( S \) is the volume distributed source, \( X \) is the boundary of the limb and \( F \) is the force on the boundary. The Navier-Stokes equation is used to model the fluid dynamics inside the limb. They solve this system to simulate the movement of the limb boundary \( X \).

This model is still a preliminary 2-d model, and its generated shape is not completely predictable. As a starting point, I can explore this model further.

Another way to obtain the actual limb shape is by experiment. I can take pictures of early limb development, and then use them as the input for the limb shape. Two
consecutive shapes do not need to be very close to each other. I seek a way that the shape at previous stage can evolve into the one at the next stage smoothly.

Figure 7.1. Limb Shapes $S_1$ and $S_2$ at two different stages: when we map $S_1$ and $S_2$ to the interval $[0, 1]$, $p_1$ and $p_2$ are mapped to the same point, then we assume $p_1$ evolve to $p_2$.

Figure 7.1 shows a 2-d example of two consecutive shapes. Suppose at Stage $i$ ($i = 1, 2$), the length of the shape is $L_i$ (without the bottom). The map $f_i : S_i \rightarrow [0, 1]$ is one-to-one map from $S_i$ to $[0, 1]$, for example,

$$f_i(p_i) = \frac{y}{L_i}, \quad (7.6)$$

where $y$ is the length from the starting point to $p_i$. If $p_1$ and $p_2$ are mapped to the same point $p$ in $[0, 1]$, let $p_1$ evolve into $p_2$. I can control the speed that $p_1$ moves to $p_2$ so that development of the shape looks continuous.
Part II

ALGORITHMS FOR PROTEIN-PROTEIN INTERACTION INFERENCE
The second part of this dissertation is about protein-protein interaction network problem. We explore new algorithms to infer protein-protein interactions [34,39].

As the protein-protein interaction is intrinsic to most cellular processes, the ability to predict which proteins in the cell interact can aid significantly in identifying the function of newly discovered proteins, and in understanding the molecular networks they participate in.

A goal of contemporary proteome research is the elucidation of the structure, interactions and functions of the proteins that constitute cells and organisms. Genomics has already produced an incredible quantity of molecular interaction data, contributing to maps of specific cellular networks. Indeed, large-scale attempts have unraveled the complex web of protein interactions in organisms as diverse as *H. pylori* [67] and *S. cerevisiae* [24,36,41,42,44,75,82]. Most recently, attention focused on the first protein interaction maps of complex multicellular organisms such as *C. elegans* [87] and *D. melanogaster* [25].

Although large-scale experimental attempts to uncover the complex webs of protein interaction in various organisms are still in progress, theoretical considerations focus on the prediction of potential protein interactions. Pioneering methods drew on the observation that interacting protein domains tend to combine into a fusion protein [20,53]. Another approach focused on the observation that functionally
linked proteins tend to be either preserved or eliminated in evolution. Proteins having matching phylogenetic profiles strongly tend to be functionally linked [52, 66]. The domain architectures of the interacting proteins account for the basic structure of a protein and offer a framework for prediction models. Interaction domain pair profiles [89] assess the potential presence of a particular interaction by clustering protein domains, depending on sequence and connectivity similarities. Another approach estimates the maximum likelihood that domains interact [15, 40]. Further ideas include overrepresented domain signatures [78], domain combination [32], graph-theoretical methods [28], the conserved properties of the protein network [2], and other probabilistic approaches [29, 80].

Assuming that protein domains facilitate the interactions among proteins, we introduce a novel method for the inference of protein interactions, which we test in *S. cerevisiae*. Utilizing a maximum-specificity set cover procedure (MSSC), we calculate the probabilities of putative protein interactions on an interaction network of yeast proteins. Our algorithm outscores previous methods in terms of sensitivity and specificity or performance. The predicted web of protein interactions keep the modular scale-free topology of the initial network. Our predictions correlate significantly with elevated levels of co-expression of micro-array data. In order to refine our predictions, we focus on a set of highly clustered interactions for our analysis. We observe that the proteins which constitute the predicted interactions are strongly co-expressed. Since interactions which are embedded in a highly clustered neighborhood tend to have an elevated degree of quality we conclude that this clustering preprocessing is a crucial step to significantly enhance the specificity of our predictions. High local clustering of interacting proteins coincides with evolutionary conservation [91] and coexpression. It has been shown that highly connected proteins display a lower evolutionary distance to their orthologous counterparts than
sparsely connected ones [22]. We observe that such a set of preprocessed inter-
actions improves MSSC’s ability to deal with significantly flawed data. Thus, we 
conclude that the combination of our algorithm with clustered interaction data helps 
to eliminate false positive and negative interaction signals, allowing for high quality 
predictions.
9.1 Materials

Investigations of the spatial protein structure suggest that the fundamental unit of protein structure is a domain. Independent of neighboring sequences, this region of a polypeptide chain folds into a distinct structure and mediates the protein’s biological functionality. The majority of proteins contain only one domain [18], while sequences of multicellular eukaryotes appear as multi-domain proteins of up to 130 domains [51].

Figure 9.1 illustrates these assumptions. Our objective is to select domain pairs (pairs of geometrical shapes in the figure) that explain the known protein interaction network. This network is the training set of the algorithm. Using the selected domain-pairs, we predict protein-protein interactions in a testing set of proteins. In order to assess the quality of our predicted interactome, typically the interactions among the proteins in the testing set are known. Thus, we can count how many real interactions we predict, and how many false positives. For these assumptions to hold, it is important to start with a curated network where the false positives have been reduced.
Figure 9.1. The fundamental units of proteins (shaded areas) are the domains (geometrical figures), mediating a distinct structure and biological functionality. We assume that the underlying protein domain architectures facilitate the interactions among proteins, allowing us to design a novel method for the inference of protein interactions in *S. cerevisiae*.

9.1.1 Protein Interactions

The first comprehensive, albeit weakly overlapping protein interaction maps of *S. cerevisiae* have been provided with the yeast-two-hybrid method [42, 75]. Currently, there exists a variety of yeast specific protein interaction databases. Many of them, such as MINT [95], MIPS [60] and BIND [4], collect experimentally determined protein interactions. PREDICTOME [57] and STRING [83] collect functional links between proteins, derived from genome scale two hybrid sets, domain fusion events, phylogenetic history, and gene proximity. These databases lack an assessment of the data’s quality. In contrast, the GRID database, a compilation of BIND, MIPS and other datasets, as well as the DIP database [93], provide sets of manually curated protein-protein interactions in *S. cerevisiae*. The majority of DIP entries are obtained from combined, non-overlapping data mostly gathered by systematic two-hybrid analysis. Here, we use the DIP database (http://dip.doe-mbi.ucla.edu) which is the qualitatively best compilation of yeast protein interaction data. The current version contains 3,677 proteins involved in 11,249 interactions for which
there is domain information. DIP also provides a high quality core set of 2,609
yeast proteins that are involved in 6,355 interactions which have been found with
more than one different experimental method.

9.1.2 Protein Domains

For our analysis, we focused on domain data retrieved from the PFAM database,
a reliable collection of multiple sequence alignments of protein families and pro-
file hidden Markov models [7] (http://pfam.wustl.edu). The current version, 10.0,
contains 6,190 fully annotated PFAM-A families. PFAM-B provides additional
PRODOM-generated [13] alignments of sequence clusters in SWISSPROT and TrEMBL
[8] that are not modeled in PFAM-A. In order to elucidate the PFAM domain ar-
chitecture, we browsed swisspfam, a compilation of the domain structure of SWIS-
SPROT and TrEMBL proteins according to PFAM.

9.1.3 Microarray Data

Genes with similar expression profiles are likely to encode interacting proteins
[30]. We assess MSSC’s ability to predict pairs of potentially interacting yeast
proteins, by utilizing gene expression data of Eisen et al. [19]. This compilation of co-
expression patterns consists of 2,467 yeast genes whose co-expression patterns have
been investigated for 79 data points. Considering the strength of our predictions, we
expect that potentially interacting proteins show an elevated degree of coexpression.

9.1.4 Conserved Network Features

Almost all biological networks are characterized by a series of organizing prin-
ciples [6]. The most dramatic is their scale-free nature, indicating that the prob-
ability that a node has degree \( k \) follows a power law, \( P(k) \sim k^{-\gamma} \) [3, 5]. Indeed,
we find this inhomogeneity in protein-protein interaction networks of numerous or-
organisms [25, 44, 86]: While most nodes have a small degree, a few highly connected hubs hold the network together [5]. The hubs’ crucial role for the protein network’s integrity is further indicated by the observation that highly interacting proteins exhibit a significantly elevated propensity to be simultaneously lethal and conserved in evolution [44, 90, 92].

Another important feature of complex networks is their tendency to cluster. The clustering coefficient [88] of a node \( i \) is defined as

\[
C_i = \frac{2n_i}{k_i(k_i - 1)},
\]

where \( n_i \) denotes the number of links connecting the \( k_i \) neighbors of node \( i \) to each other. The network’s inherent modularity is reflected by the distribution of \( C \) as a function of the nodes’ degree \( k \). If \( C(k) \) follows \( C(k) \sim k^{-1} \), the network has a hierarchical architecture, indicating that sparsely connected nodes are part of highly clustered areas [69]. This topology allows communities and the scale-free topology to seamlessly coexist [69], suggesting that complex networks are best described as the accumulation of discernible, yet topologically overlapping, functional modules. Apparently, networks featuring such functional modules are observed in almost all types of biological systems [38, 69, 71, 77] where a small subset of hubs play the important role of linking the networks modules [26, 33, 48, 69, 76]. Utilizing available yeast protein interactions as a training set of the MSSC, we expect that the web emerging from the predicted interactions will preserve these network characteristics.

9.1.5 Assessment of Protein Interactions

Although the current results concerning the structure of protein interaction networks are impressive, the observed error in experimental methods for the determination of protein interactions jeopardizes the strength of the results. A recent
estimation of the rate of inaccurately determined yeast protein interaction data un-
covered a startling false negative rate of 90% while false positives show a 50% error
rate [84]. Despite these data inconsistencies, a network topology based approach [28]
uncovered a remarkable correlation between enhanced quality and network cluster-
ing around a given protein interaction. Considering an interaction network of $N$

\[
C_{vw} = - \log \frac{\min(|N(v)|, |N(w)|)}{\sum_{i=|N(v) \cap N(w)|} \binom{|N(v)|}{i} \binom{N-|N(v)|}{|N(w)|-i}},
\]

where $N(x)$ represents the neighborhood of a vertex $x$, reflects the probability that
an interaction between proteins $v$ and $w$ indeed exists. Given the number of immediate
neighbors around the considered proteins, $N(v)$ and $N(w)$, the hypergeometric
clustering coefficient increases with elevated overlap between the protein’s neighbor-
hoods. Provided that the neighborhoods are independent, the summation can be
interpreted as a $p$ value reflecting the probability of obtaining a number of mutual
neighbors between proteins $v$ and $w$ at or above the observed number by chance [28].
We excluded the interaction between $v$ and $w$ from the calculation, rendering $C_{vw}$
independent from direct experimental evidence of the considered edge.

In our study, we calculated the link specific clustering coefficients $C_{vw}$ for each
pair of observed nodes. By applying different cut-off values, we elucidated the cor-
responding interaction network, serving as the basis for further protein interaction
predictions. We expect that interaction webs exhibiting an elevated degree of clus-
tering will raise the quality of our predictions.

9.1.6 Quality Measures

The accuracies of our predictions are measured by specificity and sensitivity. The
specificity is defined as the ratio of the number of matched interactions between the
predicted set, \( P \), and the observed testing set, \( T \), over the total number of predicted interactions, \( Sp = \frac{|P \cap T|}{|P|} \). The sensitivity is defined as the ratio of the number of matched interactions, \( P \), over the total number of observed interactions, \( T \), in the testing set, \( Sn = \frac{|P \cap T|}{|T|} \). Thus, it is obvious these metrics are testing set dependent on the choice of the testing set as well as the probability threshold.

9.1.7 Orthologous Data

The InParanoid database [70] provides orthologous sequence information for \( S. \) cerevisiae and the complete protein sets of \( H. \) sapiens, \( D. \) melanogaster, \( M. \) musculus, \( C. \) elegans and \( A. \) thaliana. Utilizing all-versus-all BLASTP searches in protein sets of two species, sequence pairs with mutually best scores were selected as central ortholog pairs. Proteins of both species showing an elevated degree of homology were clustered around these central pairs, a procedure that forms orthologous groups. The quality of the clustering was then assessed by a standard bootstrap procedure. The central ortholog sequence pair that provides a confidence level of 100% was considered as the real orthologous relationship while proteins with a lower level of confidence were considered as their in-paralogs. In our study, we selected only the central ortholog sequence pairs of each group, resulting in 1,847 yeast proteins with orthologs in \( H. \) sapiens, 1,975 in \( A. \) thaliana, 1,795 in \( C. \) elegans, 2,350 in \( M. \) musculus, and 1,565 in \( D. \) melanogaster. We also compiled a list of 976 proteins that have an ortholog in all organisms under consideration.

9.2 Previous Prediction Methods

In order to have an estimate of the quality of our predictions, we compare the performance to previous methods. In the following, we will give a brief description of the most relevant algorithms that utilize protein interactions and their corresponding domain profiles to predict otherwise unknown protein interactions in \( S. \) cerevisiae.
9.2.1 Association Method (AM)

The association method \[78\] assigns an interaction probability

\[ P(d_m, d_n) = \frac{I_{mn}}{N_{mn}} \]  

(9.3)

to each domain pair \((d_m, d_n)\). \(I_{mn}\) is the number of interacting protein pairs that contain \((d_m, d_n)\), and \(N_{mn}\) is the total number of protein pairs that contain \((d_m, d_n)\).

9.2.2 Maximum Likelihood Estimation (MLE)

The maximum likelihood estimation method \[15\] assumes that two proteins interact if at least one pair of domains of the two proteins interact, and the interactions between different domain pairs are independent. Throughout this paper, we use the same assumptions.

Under the above assumption, for any protein pair \((P_i, P_j)\), the probability of a potential interaction is

\[ E(P_i, P_j) = 1 - \prod_{(d_m, d_n) \in (P_i, P_j)} (1 - P(d_m, d_n)), \]  

(9.4)

where \(P(d_m, d_m)\) denotes the probability that domain \(d_m\) interacts with domain \(d_n\).

So, the maximum likelihood is

\[ L = \prod (E(O_{ij} = 1)^{O_{ij}} (1 - E(O_{ij} = 1))^{1-O_{ij}}, \]  

(9.5)

where \(O_{ij} = 1\) if \(P_i\) and \(P_j\) interact; otherwise, \(O_{ij} = 0\). The likelihood \(L\) is a function of \(\theta(E(d_i, d_j), f_p, f_n)\), where \(E(d_i, d_j)\) represents the probability that domains \(d_i\) and \(d_j\) interact while \(f_p\) and \(f_n\) indicate fixed rates of false positive and negative interactions in the underlying network. The maximization of \(L\) by an expectation maximization algorithm \[15\] achieved 42.5% specificity and 77.6% sensitivity on a combined yeast protein interaction set compiled from \[42,75\].
CHAPTER 10

MAXIMUM SPECIFICITY SET COVER (MSSC)

We present a novel method to predict protein-protein interactions. Our method uses a set-cover approach by choosing some domain pairs to cover the given protein-protein interactions. We say that a domain pair covers a protein-protein interaction if the two interacting proteins contain the two domains respectively.

We define the protein interaction problem as the problem of finding a set of domain pairs to represent the given protein-protein interactions. Ideally, the set of domain pairs should give as few false positives as possible. False positives are the predicted protein-protein interactions not included in the input interaction network.

10.1 Set Cover Problem

We use different approaches to predict protein-protein interaction by using protein domain information. Our approaches are based on set cover algorithms. We transform the protein-protein interaction problem to a set cover problem. By solving the induced set cover problem, we can get an answer for the original protein-protein interaction problem. In this section, we introduce the classical set cover problem, and then generalize and transform it into some other different forms which we will use later.
Figure 10.1. $X$ is the set to cover, and $\mathcal{F} = \{S_i, 1 \leq i \leq t\}$ is a family of subsets of $X$ that can cover $X$.

10.1.1 Classical Set Cover

Suppose $X$ is a finite set and $\mathcal{F} = \{S_i, 1 \leq i \leq t\}$ is a family of subsets of $X$ that can cover $X$, i.e., $X = \bigcup_{S \in \mathcal{F}} S$, see Figure 10.1. The set-cover problem is to find a subset $\mathcal{C}$ of $\mathcal{F}$ to cover $X$,

$$X = \bigcup_{S \in \mathcal{C}} S,$$

and $\mathcal{C}$ is also required to satisfy certain conditions according to different specific problems.

For example, the minimum set-cover (MSC) problem is to find a $\mathcal{C}$ with the minimum cardinality $|\mathcal{C}|$ (the number of elements in $|\mathcal{C}|$), and the minimum exact set-cover (MESC) problem requires that $\sum_{S \in \mathcal{C}} |S|$ is minimized [12, 46].

MSC and MESC are NP-complete problems. It is very expensive to find the optimal solution for a NP-complete problem, so usually people use a less expensive greedy algorithm to find an approximation instead.

Algorithm 7 is a greedy algorithm [12] for MSC. Sets for $\mathcal{C}$ is selected recursively. At every step, $U$ is the part of $X$ which is not covered yet. We choose an element $S$ (a subset of $Y$) from $\mathcal{F}$ which maximizes the intersection with $U$.

Algorithm 8 is the corresponding greedy algorithm for MESC [46]. Compared with Algorithm 7, at each intermediate step, Algorithm 8 chooses a set based on
Algorithm 7 Greedy algorithm for MSC.

```
GREEDY_MSSC(Y, X, F)
    U ← X
    C ← ∅
    while U ≠ ∅
        do select an S ∈ F that maximizes |S ∩ U|
           U ← U − S
           C ← C ∪ {S}
    return C
```

Algorithm 8 Greedy algorithm for MESC.

```
GREEDY_MSSC(Y, X, F)
    U ← X
    E ← F
    C ← ∅
    while U ≠ ∅
        do select an S ∈ E with the minimum \(\frac{|S − U|}{|S \cap U|}\)
            (a tie is broken by |S ∩ U|)
            U ← U − S
            E ← E − {S}
            C ← C ∪ {S}
    return C
```

not only the relationship with the uncovered part \(U\), but also the relationship with the covered part \(X − U\). The set to select should intersect with \(X − U\) as few as possible, and interact with \(U\) as many as possible. The ratio between these two intersections is finally minimized.

10.1.2 Generalize Set Cover

We generalize the set-cover problem by enclosing \(X\) into a bigger set \(Y\) (Figure 10.2). Suppose \(Y\) is a finite set, \(X ⊆ Y\) and \(F\) is a family of subsets of \(Y\) that can cover \(X\), i.e., \(X ⊆ \bigcup_{S ∈ F} S\). The generalized set-cover problem is to find a subset \(C\) of \(F\) to cover \(X\),

\[
X ⊆ \bigcup_{S ∈ C} S
\]  

(10.2)
and $\mathcal{C}$ is also required to satisfy certain conditions according to different specific problems, as before.

With respect to the generalized set-cover setting, the MESC problem requires that $\sum_{S \in \mathcal{C}} |S|$ be minimized. This criterion implies that both the overlap of $\mathcal{C}$ with $X$ and the overlap of $\mathcal{C}$ with $Y - X$ are minimized. MSC maintains the same requirement for the generalized case, i.e. $|\mathcal{C}|$ is minimized.

Since $X$ is enclosed in $Y$, sometimes we need to pay special attention to those elements outside $X$ (in $Y - X$). There are two interesting cases: a subcover $\mathcal{C}$ with minimum

$$\sum_{S \in \mathcal{C}} |S - X|,$$  \hspace{1cm} (10.3)

and a subcover $\mathcal{C}$ with minimum

$$|\bigcup_{S \in \mathcal{C}} S - X|.$$  \hspace{1cm} (10.4)

If we look at Equation (10.3) and Equation (10.4) carefully, we can see that both cases considers how many elements are outside $X$. Moreover Equation (10.3) cares about the overlap outside $X$, where Equation (10.4) does not.
10.2 Transformation of Protein Network to Set Cover Problem

We believe that the protein-protein interaction problem is \( NP \)-hard, although we have not proved it yet. We solve the protein-protein interaction problem by transforming it into a set-cover problem. The experimentally known protein-protein interaction network can be modeled by a graph \( G = (P, E) \), where \( P \) is the set of proteins and \( E \) is the set of edges. The proteins are the vertices of \( G \). There is an edge between two proteins if and only if they interact with each other.

Figure 10.3 shows a typical protein pair. Protein \( P_1 \) contains 3 domains: \( d_1, d_2, \) and \( d_3 \), and Protein \( P_2 \) contains two: \( d_4 \) and \( d_5 \). 6 domain pairs are possibly responsible for the interaction \((P_1, P_2)\).

Figure 10.4 shows a typical domain pair. \( d_1 \) is contained by 3 proteins: \( P_1, P_2, \) and \( P_3 \), and Domain \( d_2 \) is contained by 4: \( P_4, P_5, P_6 \) and \( P_7 \). There are 12 protein pair combinations between \( d_1 \) and \( d_2 \). Some of the protein pairs interact (in the given network), and others do not.

Furthermore, consider the set of all the possible protein pairs from the given protein network. In Figure 10.5, only solid segments represent interacting protein
Figure 10.4. $d_1$ is contained by 3 proteins: $P_1$, $P_2$, and $P_3$, and Domain $d_2$ is contained by 4: $P_4$, $P_5$, $P_6$ and $P_7$. There are 12 protein protein pair combinations between $d_1$ and $d_2$. Some of the protein pairs interact (solid lines), and others don’t.

pairs, and the dashed segments represent those non-interacting protein pairs. We group those solid segments (interactions) together.

Let us go back to Figure 10.4, where symbolically we can use the domain pair $(d_1, d_2)$ to represent all the 12 protein pairs in Figure 10.4, as in Figure 10.6.

A set-cover problem is constructed from the protein interaction network $G$ by taking

$$Y = \{ \text{all protein pairs } (P_i, P_j) | P_i, P_j \in P \},$$

$$X = \{ \text{protein pairs } (P_i, P_j) | P_i \text{ interacts with } P_j \text{ in } G \},$$

and $\mathcal{F}$ to be the set of all domain pairs $(d_m, d_n)$, where $(d_m, d_n)$ is contained by at least one element of $X$, see Figure 10.7.

A domain pair $(d_m, d_n)$ is viewed as a subset of $Y$. Specifically, if a protein pair $(P_i, P_j)$ (an element in $X$) contains $(d_m, d_n)$, then $(P_i, P_j)$ belongs to the subset $(d_m, d_n)$.

Suppose we find a subset $\mathcal{C}$ of $\mathcal{F}$ to cover every element $(P_i, P_j)$ in $X$. An
Figure 10.5. Set of all protein pairs: A solid segment represents an interaction and a dashed segment represents a non-interacting pair. We further group those solid segments (interactions) together.

Figure 10.6. A domain pair \((d_1, d_2)\) is a subset of protein pairs (solid or dashed segments). Each \(S_i\) \((1 \leq i \leq t)\) represents a domain pair.
Figure 10.7. Transformation from the protein interaction problem to the set cover problem: The big set $Y$ is taken to be the set of all potential protein pairs, the subset $X$ is taken to be the set of the given protein-protein interactions and finally the family $\mathcal{F}$ is taken to be the set of all the domain pairs.

Every element in $\mathcal{C}$ corresponds to a domain pair $(d_m, d_n)$. If $(d_m, d_n)$ covers $(P_i, P_j)$, then the two proteins $P_i$ and $P_j$ contain $d_m$ and $d_n$ respectively; so $(d_m, d_n)$ can be used to represent the interaction between $P_i$ and $P_j$. Therefore, we also have a set of domain pairs to represent the protein network $G$.

Suppose there is a set $D$ of domain pairs to represent the network $G$. For every element $(P_i, P_j)$ in $X$, there is a domain pair $(d_m, d_n)$ from $D$ to represent the interaction between $P_i$ and $P_j$. Since $(d_m, d_n)$ can be viewed as an element in $\mathcal{F}$, the collection $\mathcal{C}$ of all the domain pairs from $D$ is a subset of $\mathcal{F}$, and $\mathcal{C}$ covers $X$.

In this transformation, the set of protein-protein interactions $G$ corresponds to the set $X$ that needs to be covered, and a domain pair corresponds to an element in $\mathcal{F}$ (a subset of $Y$).

10.3 MSSC Approach

There are many ways to choose domain pairs to represent the protein interaction network. AM simply uses all possible domain pairs to explain protein-protein interactions, i.e., it uses $\mathcal{F}$ to cover $X$, so AM predicts more additional interactions and the resulting specificity is very low [15]. It is not practical to test all the predictions
Algorithm 9 Greedy algorithm for MSSC.

```plaintext
GREEDY_MSSC(Y, X, F)
    U ← X
    E ← F
    C ← ∅
    while U ≠ ∅
        do select an S ∈ E with the minimum \( \frac{|S-X|}{|S \cap U|} \)
            (a tie is broken by \( |S \cap U| \))
            U ← U − S
            E ← E − {S}
            C ← C ∪ {S}
    return C
```

experimentally. We are interested in using a subset of domain pairs to represent the protein-protein interaction network, and we choose the subset so that both the specificity and sensitivity are maximized, assuming that the training and testing set are the same. This is a reasonable assumption whenever the training set is sufficiently representative of the real testing sets, as is confirmed in our simulations.

The MSSC problem is to find a subset \( C \) of \( F \) to cover \( X \) such that,

\[
m(C) := \sum_{S \in C} |S - X|,
\]

is minimized.

Comparing MSSC with MESC, we can see that MSSC allows the subcover \( C \) to cover the overlap with \( X \), but the overlap with \( Y - X \) (outside \( X \)) is minimized. **MSSC chooses a cover in this way to maximize the specificity because false positives are considered to appear only in \( Y - X \).**

Algorithm 9 is our greedy algorithm for MSSC. \( U \) represents the uncovered part of \( X \). \( E \) is the subset of \( F \) that has not been chosen by the algorithm. Algorithm 9 is an instance of weighted set cover problem, and each set is assigned with the weight \( \frac{|S-X|}{|S \cap U|} \).
Figure 10.8. The shaded area is already covered by C. U is the unshaded area in X. The candidate set S is divided into 4 parts a, b, c and d. MSSC chooses a set S with the minimum $\frac{b+c}{a}$ while MESC chooses one with the minimum $\frac{b+c+d}{a}$. The greedy algorithm for MSSC allows overlapping of subcover inside X. The overlap actually increases the interaction probability for a protein pair.

In this algorithm, at each step when a subset needs to be chosen, we choose the one whose ratio between the part outside X and the part inside U is minimized. Note that the difference between MSSC and MESC is that MESC chooses the subset minimizing $\frac{|S-U|}{|S\cap U|}$, instead of $\frac{|S-X|}{|S\cap U|}$; MSSC allows the overlapping with X (Figure 10.8).

The number of iterations of the while loop is bounded by $\min(|X|, |F|)$, and each single iteration takes $|X||F|$ time; so the time complexity of this greedy algorithm is $O(|X||F|\min(|X|, |F|))$. If we apply proper data structures, it can be realized in $O(\log |F| \sum_{S \in F} |S|)$ time. Specifically, first, maintain a bipartite graph between elements in Y and elements in F. If the former is contained by the latter, we add an edge between them, so there are $\sum_{S \in F} |S|$ edges. Second, store all elements in F into a heap ordered by $\frac{|S-X|}{|S\cap U|}$. When a subset S is selected, it is excluded from our problem. We update the bipartite graph and the heap accordingly. The bipartite graph will not be updated more than $\sum_{S \in F} |S|$ total. For a single S, the updating of the heap takes $|S| \log |F|$. Therefore, the total time is $O(\sum_{S \in F} |S| + \sum_{S \in F} |S| \log |F|)$, which is $O(\log |F| \sum_{S \in F} |S|)$. If |F| is very big, we use an array of $|X|^2$ instead of a heap to store F, and the resulting time will be $O(|X|^2 + \sum_{S \in F} |S|)$. 94
The above greedy algorithm is just an approximation, and the solution found by it has the following relationship with the optimal solution of MSSC.

**Theorem 10.3.1.** Suppose \( C_a \) is the approximation of MSSC found by the above greedy algorithm, and \( C_o \) is an optimal subcover for MSSC. Let \( k = \max_{S \in \mathcal{F}} |S| \). If \( m(C_o) = 0 \), then \( m(C_a) = 0 \); otherwise, we have

\[
\frac{m(C_a)}{m(C_o)} \leq [\ln(k - 1) + 1]. 
\]

(10.6)

**Proof.** If \( m(C_o) = 0 \), it means that all elements in \( C_o \) are subsets of \( X \). **GREEDY_MSSC** cannot choose a set that is not completely in \( X \), because at any given time when \( U \neq \emptyset \) there exists a set \( S \in C_o \cap \mathcal{E} \) such that \( \frac{|S - X|}{|S \cap U|} = 0 \), i.e., \( S \subseteq X \). Hence \( m(C_a) = 0 \). We assume \( m(C_o) \neq 0 \) from this point. Without loss of generality, we can also assume that \( |C_o| \leq |X| \). If \( |C_o| > |X| \), it means that at least one element \( S \) in \( C_o \) is redundant to cover \( X \), so we can remove \( S \) from \( C_o \), and the remaining set is still an optimal solution.

\( |C_o| \leq |X| \) implies that \( |m(C_o)| \leq k|X| \). Suppose

\[ m(C_o) = a|X|, \text{ for some value } a, 0 < a \leq k. \]

At a given time, assume that the minimum \( \frac{|S - X|}{|S \cap U|} \) is \( r \), where \( U \) is defined as in **GREEDY_MSSC**. For any \( Z \in C_o \cap \mathcal{E} \),

\[
\frac{|Z - X|}{|Z \cap U|} \geq r, 
\]

so

\[
\frac{|Z \cap U|}{|Z - X|} \leq \frac{1}{r}. 
\]

(10.7)
We have

\[
|U| = \left| \bigcup_{z \in \mathcal{C}_a} Z \cap U \right| \\
\leq \sum_{z \in \mathcal{C}_a} \frac{|Z - X|}{r}, \text{ by Equation (10.7)} \\
= \frac{m(\mathcal{C}_a)}{r} \\
= \frac{a|X|}{r}.
\]

Therefore, there are \(|X| - |U| \geq (1 - \frac{a}{r})|X|\) points of \(X\) that are already covered when \(S\) is the next set to be chosen, i.e., \textsc{Greedy-MSSC} cannot choose a set \(S\) with

\[
\frac{|S - X|}{|S \cap U|} \geq r
\]

until a fraction \((1 - \frac{a}{r})\) of \(X\) has been covered. Conversely, if \(x = \frac{|X - U|}{|X|} = 1 - \frac{a}{r}\) (the covered part of \(X\)), then \(r = \frac{a}{1 - x}\), and for the set \(S\) chosen by \textsc{Greedy-MSSC},

\[
f(x) := \frac{|S - X|}{|S \cap U|} = \frac{a}{1 - x}.
\]

\(x\) is increasing from 0 to 1. Every time a new subset \(S\) is chosen, \(x\) “jumps” to a new value, so \(f(x)\) is a step function of \(x\). Since \(|S \cap U| \geq 1\) and \(|S - X| \leq k - 1\), \(f(x) \leq k - 1\). Note that \(\frac{a}{1 - x} = k - 1\) if and only if \(x = 1 - \frac{a}{k - 1}\).

When \textsc{Greedy-MSSC} chooses a set \(S\), \(S\) covers \(|S \cap U| = |X| \Delta x\) more points of \(X\), where \(\Delta x = \frac{|S \cap U|}{|X|}\). The contribution of \(S\) to \(m(\mathcal{C}_a)\) is

\[
|S - X| = f(x)|S \cap U| = f(x)|X| \Delta x.
\]
Therefore,

\[ m(C_a) = \sum_{s \in C_a} |S - X| \]

\[ = \sum_{s \in C_a} f(x)|X|\Delta x \]

\[ = |X| \int_{0}^{1} f(x)dx, \quad f(x) \text{ is a step function} \]

\[ \leq |X|\left[ \int_{0}^{1-a/x} \frac{a}{1-x} dx + \int_{1-a/x}^{1} (k-1)dx \right] \]

\[ = |X|(a \ln(k-1) - a \ln a + a) \]

\[ \leq a|X|\ln(k-1) + 1 \]

\[ = m(C_a)[\ln(k-1) + 1]. \]

The theorem shows the relationship between the approximation by GREEDY_MSSC and an optimal solution. If \( k \) is small, the difference between them is small too. In this theorem, \( k \) is the maximum number of elements a subset can have, and it corresponds to the maximum number of protein pairs that contain a domain pair in the protein network.

When \( X = Y \), MSSC is reduced to MSC, which is well known to be \( NP \)-hard. In the case of MSC, a logarithmic approximation is the best known approximation.

10.4 Prediction

Once the domain pairs are chosen by MSSC, each pair is assigned the same interaction probability (Equation (9.3)) as in AM. The unchosen domain pairs are given an interaction probability 0. Equation (9.4) is used to calculate the interaction probability for each putative protein pair.
CHAPTER 11

RESULTS OF MSSC AND DISCUSSION

We use two sources of protein-protein interactions: one is the combined data set of Uetz et al. [82] and Ito et al. [42], which we call CombUI; the other is a complete protein-protein interaction set retrieved from the DIP database [93], which we simply name DIP. The combined Uetz and Ito is also used in [15]. We also study the interactions with $C_{vw} \geq 5$, which are embedded in highly clustered neighborhoods in DIP. This subset of DIP we call DIP-5. We also study the yeast interactions that are evolutionary conserved in human, mouse, worm, fly, arabidopsis, and a cross-section of proteins that have orthologs in the latter organisms. these interactions were obtained from Integr8 database.

11.1 Performance Against Other Methods

We compare the ability of MSSC to predict protein-protein interactions against AM and MLE using CombUI. The training set is equal to the testing set in order to compare against published results. Figure 11.1a shows that MSSC clearly outscores AM [78] as well as MLE [15] in both specificity and sensitivity. In Figure 11.1a, we use the exact data in [15] for MLE.

MSSC uses a different criterion than the Minimum Sect Cover (MSC). MSC chooses fewer domain pairs to cover the protein interaction network, but it actually covers more false positives. We observe that MSSC clearly outscores MSC (Fig-
Figure 11.1. (a) Using CombUI as both the training set and the testing set, we compare the performance of MSSC, AM [78] and MLE [15]. MSSC shows significantly higher specificity and sensitivity. The data for MLE is the original data in [15]. (b) Using DIP as both training set and testing set, we observe that the MSSC algorithm clearly outscores MSC, allowing us to conclude that the design of the MSSC is much more suitable for the appropriate detection of potential protein interactions. We carried out an error analysis by running each algorithm 15 times (inset), allowing us to conclude that the performance of MSSC is consistent.

Algorithm GREEDY_MSSC selects a set $S \in \mathcal{E}$ with the minimum $\frac{|S - X|}{|S \cap U|}$ (a tie is broken by $|S \cap U|$). If two sets have the same $\frac{|S - X|}{|S \cap U|}$ and $|S \cap U|$, GREEDY_MSSC chooses one randomly. The inset in Figure 11.1b shows the error bar for 15 different runs, suggesting that the performance of MSSC is consistent.

MESC chooses a subcover $\mathcal{C}$ with minimum $\sum_{S \in \mathcal{C}} |S|$. The overlap inside $X$ is reduced as much as possible, while MSSC allows the overlapping inside $X$. For the data files we are using, the difference between MSSC and MESC is very slight. We expect this difference to be more significant for networks of higher eukaryotes, where more redundancy in the protein interaction network should be present.

We have also implemented MLE ourselves. The result we got is slightly different. With the MLE implemented by us, Figure 11.2a shows that MSSC clearly outscores
Figure 11.2. With the MLE implemented by us: (a) Using *CombUI* as both the training set and the testing set, MSSC outscores AM and displays a wider range of sample points than MLE. (b) Using *DIP* as both training set and testing set, MSSC has the similar result with MLE. Both outscore AM.

AM [78] and is very close to MLE [15] in both specificity and sensitivity (The MLE we implemented has better result than the original MLE [15]). When we use *DIP* as both the training and the testing sets Figure 11.2b shows that MSSC and MLE have the similar specificity vs. sensitivity curve, except that MLE has fewer sample points at higher specificity, so MSSC outscores MLE by providing more prediction sets with higher specificity. The biggest difference between MSSC and MLE is their running times. For this test, it takes MSSC 73 seconds to finish, while for MLE 6 hours and 20 minutes. MLE is a recursive method, so it takes significantly long.

11.2 Robustness of MSSC

We take different percentages of the protein network as the training set and the network itself as the testing set and compare MSSC against AM and MSC. In Figure 11.3, six different training sets are used, 10%, 20%, 40%, 60%, 80% and 100% respectively. The result shows that MSSC consistently outscores AM and MSC when the specificity is high enough, regardless of the size of training set.

The same test is also performed on *DIP-5*. Figure 11.4 shows that with a
Figure 11.3. The testing set is the whole DIP set, and different training sets are tried: (a) 10%, (b) 20%, (c) 40%, (d) 60%, (e) 80% and (f) 100% of DIP. The result shows that MSSC is consistent and outscores AM and MSC. The error bars are obtained by performing 10 runs for randomly chosen training sets.

highly clustered interaction network, the corresponding percentages for specificity and sensitivity are higher.

11.3 Conservation of Network Characteristics

The underlying protein-protein interaction network has some unique statistical characteristics. The scale-free nature is exemplified by the presence of a power-law in the networks degree distribution. The presence of modularity is indicated by a power-law dependence of the clustering coefficient $C(k)$ from degree $k$. In Figure 11.5a, we focus on interactions that score above a certain probability cutoff, allowing us to observe that the power-law dependence of the degree distribution of the networks thus emerging remains untouched (inset). In order to support this qualitative observation quantitatively, we applied a two dimensional Kolmogorov Smirnov test. Comparing the degree distribution of the original yeast protein interaction network
Figure 11.4. The testing set is DIP-5, and different training sets are tried: (a) 10%, (b) 20%, (c) 40%, (d) 60%, (e) 80% and (f) 100% of DIP-5. The result shows that MSSC consistently outscores AM and MSC. The error bars are obtained by performing 10 runs for randomly chosen training sets.

with the predicted networks emerging from the application of different probability thresholds, we find small differences ranging from 0.17 (threshold $t = 0.4$) to 0.28 (threshold $t = 1.0$). Furthermore, we find that the corresponding P-values gradually decrease from $P_{t=0.4} = 0.27$ to $P_{t=1.0} = 9.6 \times 10^{-2}$, allowing us to conclude that the observed distributions are basically drawn from the same statistical sample. In the same way, the choice of the cutoff value does not seem to impair the emergence of modules as well. We still find the power-law dependence of the clustering coefficients of the networks emerging from the application of different thresholds $t$, observations that are strongly supported by KS-scores (KSS) ranging from $KSS_{t=0.4} = 0.17$ ($P_{t=0.4} = 0.99$) to $KSS_{t=1.0} = 0.24$ ($P_{t=1.0} = 0.88$).

A different assessment of the prediction quality is the tendency of interactions toward co-expression. Utilizing the initial protein interaction data of $S. \textit{cerevisiae}$ and a set of co-expression data [19], we observe a bell-shaped curve peaking around
Figure 11.5. To assess the quality of protein interaction predictions of MSSC, we determine the statistical characteristics of the networks which emerges from our predictions. (a) Utilizing DIP, we observe that the degree distribution follows a power-law (inset). Accounting for interactions which score above a certain probability, we find that the power-law dependence of the degree remains largely unchanged. Similarly, we observe that the power-law dependencies of the clustering coefficient $C(k)$ from the degree as exemplified by the experimental data does largely not depend on the choice of the cutoff value. (b) Utilizing a set of co-expression data, we observe a bell-shaped distribution curve. Accounting only for interactions that score above certain cutoffs we observe that proteins which participate in interactions scoring higher probability strongly tend to be co-expressed.

a zero co-expression coefficient. If there exists a correlation between the presence of an interaction between a pair of proteins and their co-expression, we expect a shift to higher expression coefficients. Figure 11.5b shows that higher probability cutoffs let the resulting networks exhibit an enrichment of co-expressed interacting proteins. Assuming that the observed distributions have roughly the same variance, we apply a Student’s t-test to uncover possibly different means of the predicted co-expression profiles. Applying different thresholds $t$, we find statistically significant t-scores (TTS) $TTS_{t=0.4} = 18.38$ ($P_{t=0.4} = 2.9 \times 10^{-72}$) to $TTS_{t=1.0} = 15.86$ ($P_{t=1.0} = 6.1 \times 10^{-53}$), confirming our observation that an elevated amount of interacting proteins scores higher expression coefficients.
11.4 Results with High Quality Interactions

Currently available sets of protein-protein interactions contain startling rates of false positives ($\sim 50\%$) and false negatives ($\sim 90\%$) [84]. Recently, the quality of a protein-protein interaction was observed to correlate well with the degree of clustering of its immediate networks neighborhood [28]. We assume that our prediction results can be significantly improved by focusing on such highly clustered links. Calculating the hypergeometric clustering coefficient for every link in the yeast interaction network, we elucidated only those interactions that score above a certain level of clustering. In order to assess the strength of interactions which are embedded in an increasingly clustered neighborhood, we calculated the corresponding specificity vs. sensitivity curves and found that the quality of predictions was significantly improved. Increasing clustering coefficient reduces the network and gives better results when comparing their corresponding sensitivity vs. specificity curves (Fig. 11.6a).

Encouraged by these results, we assume that the proteins which participate in the corresponding interactions will significantly be coexpressed. Indeed, we find that the distribution of the coexpression coefficients emerging from the predicted interacting proteins peaks around shifts toward higher values if the threshold is increased (Figure 11.6b (inset)). Since we predict interactions evaluated by an occurrence probability, we consider predicted pairs that score above $p > 0.9$. We observe that the initial trends of coexpression are enhanced. These results are further supported by highly significant $P$-values of a Students $t$-test which gradually decrease from $P_{t=1} = 3.3 \times 10^{-21}$ to $P_{t=15} = 9.26 \times 10^{-4}$, indicating that the limitation to clustered proteins which participate in clustered interactions indeed significantly elevate the quality of our predictions.
Figure 11.6. (a) Observing that interactions which are embedded in a highly clustered neighborhood exhibit a significantly higher degree of quality, we calculated the hypergeometric clustering coefficient $C_{vw}$ of each link in the original protein-protein interaction network of Yeast. A network composed of interactions that score above a certain threshold is the basis for our predictions, which outscores the original predictions in terms of specificity and sensitivity. (b) Compared to a comprehensive set of coexpression values of protein links (background), the quality of the underlying stripped protein interaction network increases with increased threshold (inset). Since we predict interactions evaluated by an occurrence probability, we only consider predicted pairs that score above $p > 0.9$. The quality of predicted pairs thus obtained is well reflected by an elevated tendency of coexpression, measurements that are supported by low P-values ($P < 10^{-3}$) of a Student’s test.
11.5 Results with Orthologous Interactions

Figure 11.7. (a) Proteins that have orthologs in \textit{H. sapiens}, \textit{M. musculus}, \textit{D. melanogaster}, \textit{C. elegans} and \textit{A. thaliana} are embedded in a highly clustered neighborhood. Elucidating the links between orthologs, we used subnetworks thus emerging for our predictions. We observe that the results thus obtained are comparable to the distributions we obtained with interactions that score above a certain clustering threshold (Fig. 11.6). In particular, we observe that the predictions with a subnetwork of proteins that have an ortholog in all organisms (cross-section) under consideration scores better than with organism specific ones. (b) Compared to a set of coexpression values of protein links (background), the quality of the underlying orthologous protein-protein interaction network increases (inset) by shifting to higher values of coexpression. Considering only predicted pairs that score above \( p > 0.9 \), the tendency of coexpression further enhances, further supported by low P-values of a Students t-test (\( P < 10^{-9} \)). For these tests \( R = T \).

The observation that highly clustered areas provide high reliability of the corresponding links returns in a more biological disguise. In a recent work, densely connected small sub-networks or motifs were found to exhibit a significantly higher degree of conservation than their sparsely connected counterparts [91]. Combining the latter aspect with the already uncovered correlation of high clustering and preferential coexpression, we expect that yeast interaction networks that consist of proteins that have an ortholog to a higher order organism are good candidates to improve our prediction results. Utilizing ortholog data of the higher eukaryotes \textit{H. sapiens}, \textit{D. melanogaster}, \textit{M. musculus}, \textit{C. elegans} and \textit{A. thaliana} we elucidated
the corresponding interaction network of yeast proteins. Applying our method to predict potential interactions from these networks we clearly observe that the predictions clearly outscore the respective ones utilizing the whole yeast interaction network. As a further improvement, we span an interaction network by proteins that have an ortholog in all organisms under consideration (cross-section).

Indeed, we find that this set of interactions which presents a core of the higher eukaryotes once again exceeds the results of the latter predictions (Fig. 11.7a). Utilizing yeast coexpression data, we find our initial assumption that clustered interactions show preferential coexpression to be confirmed. In all cases, we observe a clear trend toward coexpression of the predicted interactions (Fig. 11.7b, inset), tendencies that are enhanced by limiting to interactions that score above occurrence probability \( p > 0.9 \) (Fig. 11.7b). Supporting their significance, we observe gradually decreasing P-values of Student’s t-tests (\( M. \text{musculus}: P = 8.6 \times 10^{-18} \) to \( A. \text{thaliana}: P = 5.8 \times 10^{-10} \)). Once again, the set of proteins that have an ortholog in all organism under consideration indicates it’s special status providing predictions that show the strongest, significant shift toward coexpression (\( P 8.5 \times 10^{-10} \)).

11.6 Disjoint Training and Testing Sets

We also run tests with disjoint training and testing sets. We take about 80% of \( DIP \) or \( DIP-5 \) as the training set, and the testing set is the remainder. Figure 11.8 shows that the specificity/sensitivity curve is significantly shifted to higher values than the corresponding one obtained from the full protein interaction network.

11.7 Dissemination

We created a web portal for protein-protein interaction, see Figure 11.9. The portal provides a means for predicting and analyzing interactions using different predic-
Figure 11.8. Performance of MSSC in the test of disjoint training and testing sets. The training sets are 80\% of DIP and DIP − 5, and the testing sets are the remainder. The predicted interactions in DIP-5 exhibit a significantly higher degree of quality.

The portal allows the user to create or download protein interaction data sets, calculate clustering coefficients for protein pairs, predict interactions, and analyze the PPI program output. This consists of the ability to view and download a Sensitivity vs. Specificity plot, and search individual protein names to get a list of all of its interactors.

11.8 Discussion

Our results suggest that the quality of predicted protein interactions depends basically on two different aspects. First, the quality of our predictions is strongly enhanced if we pre-assess the quality of the underlying protein interactions by determining the degree of clustering of the interaction’s immediate neighborhood in the network. The observation that highly clustered links exhibit an elevated reliability is an important step toward the reliable prediction of potential interactions, since the considerable error-proneness of protein interaction data clearly influences the quality of results. The correlation between well clustered neighborhoods around
the considered links and their interaction quality is further supported by the significantly elevated degree of co-expressed proteins that participate in present and predicted interactions. This observation is not only a proof of concept, it also suggests that potential strategies for the determination of potential protein interactions have to focus on co-expressed areas of the underlying interaction network.

On the other hand, by design, our MSSC approach selects a set of domain pairs that both cover the experimental observations and that maximize the specificity in the training set. Our results indicate that there is a strong correlation between high specificity in high quality training sets and high specificity in realistic testing sets.

We showed a new way of integrating protein interaction and domain data with a quality assessment of the underlying web of interactions. A further improvement of the algorithm will focus on the systematic integration of such pre-assessed interaction data in order to ensure highly reliable predictions. Once large-scale protein
interaction sets of organisms other than *S. cerevisiae* are available, we expect that our algorithm will contribute significantly to the elucidation of complete organism-specific interactomes.
CHAPTER 12

OTHER SET COVER APPROACHES AND DOMAIN COMBINATIONS

12.1 Set Cover Allowing Overlap

MSSC is to find a subcover \( C \) with minimum \( \sum_{S \in C} |S - X| \). If two subsets \( S_1 \) and \( S_2 \) have an overlap outside \( X \), the overlap is counted twice. Consider the following example in Figure 12.1, where \( Y \) is the set of all the points, \( X \) is the set of points between the two solid circles, \( S_i \) \((1 \leq i \leq 8)\) is the set of points inside the only ellipse which \( S_i \) is in and \( \mathcal{F} = \{S_i, 1 \leq i \leq 8\} \).

MSSC chooses \( C = \{S_i, 1 \leq i \leq 4\} \), but actually \( C' = \{S_i, 5 \leq i \leq 8\} \) covers fewer additional elements outside \( X \). \( C \) covers 8 points outside \( X \), while \( C' \) covers only 3. The 3 additional elements covered by \( C' \) may be very important, but are missing from the given protein network due to the defect of the network.

\( C' \) has is the subcover with the minimum \( |\bigcup_{S \in C'} S - X| \), so the overlap of two subsets \( S_1 \) and \( S_2 \) outside \( X \) is counted only once, i.e. \( C' \) allows subsets overlap outside \( X \).

Due to the simplicity of the available protein interaction networks, the difference between these two set cover approaches is very slight.

12.2 Domain Combinations

Up to now, we assume that a protein-protein interaction is realized by the interactions between domain pairs, and the interaction between one domain pair is inde-
Figure 12.1. \( Y \): set of all the points; \( X \): set of points between the two solid circles; \( S_i \) (1 \( \leq \) \( i \) \( \leq \) 8): set of points inside the only ellipse which \( S_i \) is in; \( \mathcal{F} = \{ S_i, 1 \leq i \leq 8 \} \).

Evidence shows that some domains in a protein form a group to interact with another group of domains from another protein, i.e. a protein-protein interaction is realized by the interaction between two groups of domain pairs. The related work about domain combinations can be found in [32].

We can also add this feature into our set cover algorithm. We take a combination of certain number of domains inside a protein to be a new domain, and the rest of the algorithm remains the same. If a protein contains \( n \) domains, the number of domain combinations is \( 2^n - n - 1 \). Sometimes it is enough for us to consider the domain combinations with no more than \( k \) domains, where \( k \) ranges from 2 to \( n \). Suppose Protein \( P \) contains \( n \) domains \( d_1, d_2, \ldots, d_n \). Algorithm 10 is the algorithm for combining up to \( k \) domains. Figure 12.2 shows that domain combinations can improve the quality of the specificity vs. sensitivity curve.

According to the transformation from the protein interaction problem to the set cover problem, two domain combinations (a pairs of new domains) represent a subset of \( Y \), the set of all potential protein pairs. Let \( d_{12} \) be the combination of domain \( d_1 \) and \( d_2 \), \( d_{34} \) the combination of domain \( d_3 \) and \( d_4 \), then as subsets of \( Y \),
Algorithm 10 Algorithm for Domain Combination.

Sort the domains $d_1, d_2, \ldots, d_n$.

P− >domains.resize(0);

for i=1 to n do
    originalSize=P− >domains.size();
    for j=1 to originalSize
        if P− >domains[j] is combined of l domains and l is less than k
            P− >domains.push_back(P− >domains[j]+d[i]);
            record that P− >domains.end() is combination of (l+1) domains;
            // For example, put l+1 at the head of the domain.
        endif
    endfor
    P− >domains.push_back(d[i]);
endfor

Figure 12.2. Domain combination can improve the quality of specificity vs. sensitivity. DIP is used as training and testing sets.
\( (d_{12}, d_{34}), (d_1, d_2) \) and \( (d_3, d_4) \) satisfy:

\[
(d_{12}, d_{34}) = (d_1, d_3) \cap (d_1, d_4) \cap (d_2, d_3) \cap (d_2, d_4).
\]

Generally, if \( d_{i_1, i_2, ..., i_m} \) is the combination of domain \( d_{i_1}, d_{i_2}, ..., d_{i_m} \), then we have

\[
(d_{i_1, i_2, ..., i_m}, d_{j_1, j_2, ..., j_n}) = \bigcap_{1 \leq k \leq m, 1 \leq l \leq n} (d_{i_k}, d_{j_l}).
\]

If we consider the extreme case that all the domain combinations are included, then a protein itself as a combination of all its domains is included too. In the greedy algorithm for MSSC (Algorithm 9), as long as there are some uncovered elements (protein-protein interactions), Algorithm 9 will choose a subset which completely resides in \( X \) (any uncovered protein-protein interaction (viewed as a pair of domain combinations) is such a set). Therefore, the specificity vs. sensitivity curve will converge to a point (100%, 100%). Of course, this extreme case does not help us a lot in practice, because we cannot predict many additional protein-protein interactions. All the predictions are almost the given interactions. It tells us that increasing the domain combination depth \( k \) does give us a better specificity vs. sensitivity curve, but it is not always very useful.
CHAPTER 13

SUMMARY OF PROTEIN-PROTEIN INTERACTION AND FUTURE WORK

13.1 Summary of Results

The second part of this dissertation presents different approaches using the set cover technique to choose domain pairs to represent the given protein interaction network and predict new protein-protein interactions.

We focus on the maximum specificity set cover (MSSC). MSSC is to find a subcover $\mathcal{C}$ with minimum $\sum_{S \in \mathcal{C}} |S - X|$. It improves specificity because the false positive appear in $S - X$. We compare MSSC with the association method (AM) and the maximum likelihood method (MLE). MSSC outscores MLE and AM.

We also discuss another set cover approach: find a subcover $\mathcal{C}$ with minimum $|\bigcup_{S \in \mathcal{C}} S - X|$. For the available protein interaction networks, it has almost the exact same result with MSSC.

Using Domain combination is another technique which can be added into all our set cover approaches. It improves the specificity vs. sensitivity curve, but there are two limitations:

1. People are not sure whether a protein-protein interaction is caused by a pair of domains, or by a pair of combinations of domains, and how many domains the combinations should contain.

2. If the specificity is too high, it cannot provide us useful new predictions.
13.2 Future Work

13.2.1 Improvement of the Algorithms

Due to the difficulty of finding an optimal solution for a set cover problem, the algorithms we presented for MSSC is just a greedy algorithm. We showed the difference between the approximation found by a greedy algorithm and an optimal solution. Sometimes the difference may be quite big.

If the speed is not our primary concern, we can improve the quality by taking unions of elements in $F$ into account. If all the unions are considered, we can find an optimal solution, because any optimal solution itself is a union of elements in $F$.

13.2.2 Improvement of Domain Pair Interacting Probability

I use set cover approach to choose domain pairs to cover the given protein interaction network. After a domain pair is chosen, it is assigned an interacting probability according to Equation (9.3). $I_{mn}$ depends on the training set quality. It is possible that among those $N_m n$ protein pairs not all the interactions are given by the training set, so the actual interacting probability for a domain pair should be higher than that given by Equation (9.3). We need to find a better way to estimate the interacting probability for a domain pair.
APPENDIX A

METHOD OF LEAST SQUARES

Consider an over-determined linear system

\[ Mx = d, \]  

where \( M \) is an \( m \times n \) (\( m > n \)) matrix, \( d \) and \( r \) are a column vector. \( r = Mx - d \) is called the residual. The goal is to find the unknown vector \( x \) such that the sum of squares of the residuals \( r_i \) is minimal.

The sum of the squares of the residuals \( r_i \) is \( r^T r \). From Equation (A.1), we have

\[
\begin{align*}
r^T r &= (Mx - d)^T(Mx - d) \\
&= x^T M^T M x - x^T M^T d - d^T M x + d^T d \\
&= x^T M^T M x - 2(M^T d)^T x + d^T d.
\end{align*}
\]

(A.2)

To simplify the notation we define

\[
A := M^T M \quad b := M^T d \quad A \in R^{n \times n} \quad b \in R^n.
\]

(A.3)

When \( \text{rank}M = n \), the symmetric matrix \( A \) is positive definite.

The problem becomes

\[
\text{minimize } f(x) := r^T r = x^T A x - 2b^T x + d^T d.
\]

(A.4)
A necessary condition for a minimum of \( f(x) \) at the point \( x \) is the vanishing of its gradient \( \nabla f(x) \). Since

\[
\nabla f(x) = 2Ax - 2b = 2(Ax - b), \quad (A.5)
\]

after division by 2 we obtain the linear system of equations:

\[
Ax = b. \quad (A.6)
\]

According to the Gaussian principle, the classical treatment of Equation A.1 consists of the following simple solution steps:

1. \( A = M^T M \ b = M^T d \) (normal equations \( Ax = b \)),
2. \( A = L^T L \) (Cholesky’s decomposition),
3. \( L^T y = b \ Lx = y \) (forward/backward substitution),
4. \( r = Mx - d \) (computation of residuals).
APPENDIX B

APPROXIMATION OF THE FIRST ORDER DERIVATIVES

Let $G$ be a grid decomposition for a domain $D$. Suppose $p_0$ is a grid point and has $n$ neighbors $p_{01}, p_{02}, p_{03}, \ldots, p_{0n}$, ordered counter clockwise, see Figure 6.2. Let denote the vector from $p_0$ to $p_{0i}$ by $\vec{v}_{0i}$ ($1 \leq i \leq n$).

Suppose $u = u(x, y)$ is an unknown function. The directional derivative of $u$ along $\vec{v}_{0i}$ ($1 \leq i \leq n$) can be written as a linear combination of $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$.

$$\frac{\partial u}{\partial \vec{v}_{0i}}(p_0) = \frac{1}{|\vec{v}_{0i}|}(a_{0i} \frac{\partial u}{\partial x}(p_0) + b_{0i} \frac{\partial u}{\partial y}(p_0)),$$

where $\vec{v}_{0i} = (a_{0i}, b_{0i})$, and $|\vec{v}_{0i}|$ means the length of the vector $\vec{v}_{0i}$. Denote $\frac{\partial u}{\partial x}(p_0)$ by $\frac{\partial u_0}{\partial x}$ and $\frac{\partial u}{\partial y}(p_0)$ by $\frac{\partial u_0}{\partial y}$.

Since

$$\frac{\partial u}{\partial \vec{v}_{0i}}(p_0) \approx \frac{u(p_{0i}) - u(p_0)}{|\vec{v}_{0i}|},$$

we have

$$a_{0i} \frac{\partial u_0}{\partial x} + b_{0i} \frac{\partial u_0}{\partial y} \approx u(p_{0i}) - u(p_0).$$

We have two unknowns $\frac{\partial u_0}{\partial x}$, $\frac{\partial u_0}{\partial y}$ and $n$ equations, it is an over-determined system. We can use the method of least squares to solve it. See Appendix A for the details.

Here we have:
\[ A = \begin{pmatrix} a_{01} & a_{02} & \cdots & a_{0n} \\ b_{01} & b_{02} & \cdots & b_{0n} \end{pmatrix} \begin{pmatrix} a_{01} & b_{01} \\ a_{02} & b_{02} \\ \vdots & \vdots \\ a_{0n} & b_{0n} \end{pmatrix} = \begin{pmatrix} \sum_1^n (a_{0i})^2 & \sum_1^n a_{0i}b_{0i} \\ \sum_1^n a_{0i}b_{0i} & \sum_1^n (b_{0i})^2 \end{pmatrix}, \quad (B.4) \]

\[ b = \begin{pmatrix} a_{01} & a_{02} & \cdots & a_{0n} \\ b_{01} & b_{02} & \cdots & b_{0n} \end{pmatrix} \begin{pmatrix} u_{01} - u_0 \\ u_{02} - u_0 \\ \vdots \\ u_{0n} - u_0 \end{pmatrix} = \begin{pmatrix} \sum_1^n a_{0i}(u_{0i} - u_0) \\ \sum_1^n b_{0i}(u_{0i} - u_0) \end{pmatrix}. \quad (B.5) \]

Solving \( Ax = b \), we get

\[
\begin{cases}
\frac{\partial u}{\partial x} = \frac{\sum_1^n a_{0i}(u_{0i} - u_0) \sum_1^n (b_{0i})^2 - \sum_1^n b_{0i}(u_{0i} - u_0) \sum_1^n a_{0i}b_{0i}}{\sum_1^n (a_{0i})^2 \sum_1^n (b_{0i})^2 - (\sum_1^n a_{0i}b_{0i})^2}, \\
\frac{\partial u}{\partial y} = \frac{\sum_1^n b_{0i}(u_{0i} - u_0) \sum_1^n (a_{0i})^2 - \sum_1^n a_{0i}(u_{0i} - u_0) \sum_1^n a_{0i}b_{0i}}{\sum_1^n (a_{0i})^2 \sum_1^n (b_{0i})^2 - (\sum_1^n a_{0i}b_{0i})^2}.
\end{cases}
\quad (B.6)
\]

Simplifying the numerator of \( \frac{\partial u}{\partial x} \),

\[
\sum_1^n a_{0i}(u_{0i} - u_0) \sum_1^n (b_{0i})^2 - \sum_1^n b_{0i}(u_{0i} - u_0) \sum_1^n a_{0i}b_{0i} \\
= \sum_{i=1}^n (u_{0i} - u_0) \sum_{j=1}^n a_{0i}(b_{0j})^2 - \sum_{i=1}^n (u_{0i} - u_0) \sum_{j=1}^n b_{0i}a_{0j}b_{0j} \\
= \sum_{i=1}^n (u_{0i} - u_0) \sum_{j=1}^n (a_{0i}b_{0j} - a_{0j}b_{0i})b_{0j}. \quad (B.7)
\]

Similarly,

\[
\sum_1^n b_{0i}(u_{0i} - u_0) \sum_1^n (a_{0i})^2 - \sum_1^n a_{0i}(u_{0i} - u_0) \sum_1^n a_{0i}b_{0i} \\
= \sum_{i=1}^n (u_{0i} - u_0) \sum_{j=1}^n (a_{0j}b_{0i} - a_{0i}b_{0j})a_{0j}. \quad (B.8)
\]

Simplifying the denominator of \( \frac{\partial u}{\partial x} \),

120
\[
\sum_{i=1}^{n} (a_{0i})^2 \sum_{j=1}^{n} (b_{0j})^2 - (\sum_{i=1}^{n} a_{0i}b_{0i})^2 \\
= \sum_{i=1}^{n} (a_{0i})^2 \sum_{j=1}^{n} (b_{0j})^2 - \sum_{i=1}^{n} a_{0i}b_{0i} \sum_{j=1}^{n} a_{0j}b_{0j} \\
= \sum_{i=1}^{n} \sum_{j=1}^{n} ((a_{0i})^2(b_{0j})^2 - a_{0i}b_{0i}a_{0j}b_{0j}) \\
= \sum_{i=1}^{n} \sum_{j=1}^{n} ((a_{0i})^2(b_{0j})^2 - 2a_{0i}b_{0i}a_{0j}b_{0j} + (a_{0j})^2(b_{0i})^2) \\
= \sum_{i=1}^{n} \sum_{j=i+1}^{n} (a_{0i}b_{0j} - a_{0j}b_{0i})^2. \quad (B.9)
\]

Let us introduce some notations:

\[
D_0 = \sum_{i=1}^{n} \sum_{j=i+1}^{n} (a_{0i}b_{0j} - a_{0j}b_{0i})^2, \quad (B.10)
\]

\[
A_{0i} = \frac{1}{D_0} \sum_{j=1}^{n} (a_{0j}b_{0i} - a_{0i}b_{0j})a_{0j}, \quad (B.11)
\]

\[
B_{0i} = \frac{1}{D_0} \sum_{j=1}^{n} (a_{0i}b_{0j} - a_{0j}b_{0i})b_{0j}. \quad (B.12)
\]

Finally, we have

\[
\frac{\partial u_0}{\partial x} = \sum_{i=1}^{n} B_{0i}(u_{0i} - u_0) = \sum_{i=1}^{n} B_{0i}u_{0i} - \sum_{i=1}^{n} B_{0i}u_0, \quad (B.13)
\]

\[
\frac{\partial u_0}{\partial y} = \sum_{i=1}^{n} A_{0i}(u_{0i} - u_0) = \sum_{i=1}^{n} A_{0i}u_{0i} - \sum_{i=1}^{n} A_{0i}u_0. \quad (B.14)
\]

One can clearly see the coefficients before \(u_0\) and \(u_{0i}\) \((1 \leq i \leq n)\).
APPENDIX C

APPROXIMATION OF THE SECOND DERIVATIVES I

The second order derivatives of $u(x, y)$ are the first order derivatives of $\frac{\partial u}{\partial x}$ and $\frac{\partial u}{\partial y}$. We can use the technique in Appendix B twice to approximate the second order derivatives of $u(x, y)$. The first order derivatives of $u(x, y)$ at $p_0$ depend on $p_0$ and its first neighbors; The second order derivatives of $u(x, y)$ at $p_0$ depend on $p_0$, its first neighbors and its second neighbors. The formula for the second order derivatives are as follows.

\[
\begin{align*}
\frac{\partial^2 u}{\partial x^2} &= \sum_{i=1}^{n} B_{0i}(\frac{\partial u_{0i}}{\partial x} - \frac{\partial u_0}{\partial x}), \\
\frac{\partial^2 u}{\partial y^2} &= \sum_{i=1}^{n} A_{0i}(\frac{\partial u_{0i}}{\partial y} - \frac{\partial u_0}{\partial y}), \\
\frac{\partial^2 u}{\partial x \partial y} &= \sum_{i=1}^{n} A_{0i}(\frac{\partial u_{0i}}{\partial x} - \frac{\partial u_0}{\partial x}), \\
\frac{\partial^2 u}{\partial y \partial x} &= \sum_{i=1}^{n} B_{0i}(\frac{\partial u_{0i}}{\partial y} - \frac{\partial u_0}{\partial y}),
\end{align*}
\]

(C.1)

where $\frac{\partial u_{0i}}{\partial x} = \frac{\partial u}{\partial x}(p_{0i})$ and $\frac{\partial u_{0i}}{\partial x} = \frac{\partial u}{\partial x}(p_0)$.

\[
\begin{align*}
\frac{\partial^2 u_0}{\partial x^2} &= \sum_{i=1}^{n} B_{0i}(\frac{\partial u_{0i}}{\partial x} - \frac{\partial u_0}{\partial x}) \\
&= \sum_{i=1}^{n} B_{0i}(\sum_{j=1}^{m_i} B_{0ij}(u_{0i}^{ij} - u_{0i}) - \sum_{j=1}^{n} B_{0j}(u_{0j} - u_0)) \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m_i} B_{0i} B_{0ij} u_{0ij} - \sum_{i=1}^{n} u_{0i}(B_{0i}(\sum_{j=1}^{m_i} B_{0ij} + \sum_{j=1}^{n} B_{0j})) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} B_{0i} B_{0j},
\end{align*}
\]

(C.2)
where \( m_i \) means the number of neighbors of \( u_{0i} \), and \( u_{0ij} \) represents the \( j \)-th neighbor of \( u_{0i} \).

Similarly,

\[
\frac{\partial^2 u_0}{\partial y^2} = \sum_{i=1}^{n} A_{0i} \left( \frac{\partial u_{0i}}{\partial y} - \frac{\partial u_0}{\partial y} \right)
= \sum_{i=1}^{n} A_{0i} \left( \sum_{j=1}^{m_i} A_{0ij} (u_{0j}^{ij} - u_{0i}) - \sum_{j=1}^{n} A_{0j} (u_{0j} - u_0) \right)
= \sum_{i=1}^{n} \sum_{j=1}^{m_i} A_{0i} A_{0ij} u_{0j}^{ij} - \sum_{i=1}^{n} u_{0i} \left( A_{0i} \left( \sum_{j=1}^{m_i} A_{0ij} + \sum_{j=1}^{n} A_{0j} \right) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{0i} A_{0j} \right), \quad (C.3)
\]

\[
\frac{\partial^2 u_0}{\partial x \partial y} = \sum_{i=1}^{n} A_{0i} \left( \frac{\partial u_{0i}}{\partial x} - \frac{\partial u_0}{\partial x} \right)
= \sum_{i=1}^{n} A_{0i} \left( \sum_{j=1}^{m_i} B_{0ij}^{ij} (u_{0j}^{ij} - u_{0i}) - \sum_{j=1}^{n} B_{0j} (u_{0j} - u_0) \right)
= \sum_{i=1}^{n} \sum_{j=1}^{m_i} A_{0i} B_{0ij} u_{0j}^{ij} - \sum_{i=1}^{n} u_{0i} \left( A_{0i} \left( \sum_{j=1}^{m_i} B_{0ij} + \sum_{j=1}^{n} B_{0j} \right) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} A_{0i} B_{0j} \right), \quad (C.4)
\]

and

\[
\frac{\partial^2 u_0}{\partial y \partial x} = \sum_{i=1}^{n} B_{0i} \left( \frac{\partial u_{0i}}{\partial y} - \frac{\partial u_0}{\partial y} \right)
= \sum_{i=1}^{n} B_{0i} \left( \sum_{j=1}^{m_i} A_{0ij}^{ij} (u_{0j}^{ij} - u_{0i}) - \sum_{j=1}^{n} A_{0j} (u_{0j} - u_0) \right)
= \sum_{i=1}^{n} \sum_{j=1}^{m_i} B_{0i} A_{0ij}^{ij} u_{0j}^{ij} - \sum_{i=1}^{n} u_{0i} \left( B_{0i} \left( \sum_{j=1}^{m_i} A_{0ij}^{ij} + \sum_{j=1}^{n} B_{0j} \right) + u_0 \sum_{i=1}^{n} \sum_{j=1}^{n} B_{0i} A_{0j} \right). \quad (C.5)
\]

One can see that the formula is really nasty. Moreover, by doing it in this way,
the second order derivatives have to depend on the second neighbors. The farther the neighbors, the bigger the discretization error. Actually we can develop a formula for the second order derivatives so that they depend on the point itself and its first neighbors only. Therefore, the bandwidth of the coefficient matrix is reduced. I will talk about how in Appendix D.
APPENDIX D

APPROXIMATION THE SECOND DERIVATIVES II

I present another technique to approximate the second derivatives here. The Taylor’s formula for \( u(x, y) \) at the point \((a, b)\) is

\[
\begin{align*}
    u(a + h, b + k) &= u(a, b) + (hu_x + ku_y)|_{(a, b)} + \\
                     &\quad \frac{1}{2!}(h^2u_{xx} + 2hku_{xy} + k^2u_{yy})|_{(a, b)} + \cdots. \\
\end{align*}
\]

Therefore, for \( i = 1, 2, \cdots, n, \)

\[
    u_{0i} - u_0 \approx a_{0i} \frac{\partial u_0}{\partial x} + b_{0i} \frac{\partial u_0}{\partial y} + \frac{1}{2}((a_{0i})^2 \frac{\partial^2 u_0}{\partial x^2} + 2a_{0i}b_{0i} \frac{\partial^2 u_0}{\partial x \partial y} + (b_{0i})^2 \frac{\partial^2 u_0}{\partial y^2}).
\]

In another form,

\[
    (a_{0i})^2 \frac{\partial^2 u_0}{\partial x^2} + 2a_{0i}b_{0i} \frac{\partial^2 u_0}{\partial x \partial y} + (b_{0i})^2 \frac{\partial^2 u_0}{\partial y^2} \approx 2(u_{0i} - u_0 - (a_{0i} \frac{\partial u_0}{\partial x} + b_{0i} \frac{\partial u_0}{\partial y})).
\]

We already have the formula for \( \frac{\partial u_0}{\partial x} \) and \( \frac{\partial u_0}{\partial y} \). Taking them as given and solving another over-determined system \( Ax = b \), where
\[ A = \begin{pmatrix} (a_{01})^2 & (a_{02})^2 & \cdots & (a_{on})^2 \\ 2a_{01}b_{01} & 2a_{02}b_{02} & \cdots & 2a_{on}b_{on} \\ (b_{01})^2 & (b_{02})^2 & \cdots & (b_{on})^2 \end{pmatrix} \begin{pmatrix} (a_{01})^2 & 2a_{01}b_{01} & (b_{01})^2 \\ (a_{02})^2 & 2a_{02}b_{02} & (b_{02})^2 \\ \vdots & \vdots & \vdots \\ (a_{on})^2 & 2a_{on}b_{on} & (b_{on})^2 \end{pmatrix} \]

\[ = \begin{pmatrix} \sum_1^n(a_{0i})^4 & \sum_1^n 2(a_{0i})^3b_{0i} & \sum_1^n(a_{0i}b_{0i})^2 \\ \sum_1^n 2(a_{0i})^3b_{0i} & \sum_1^n(2a_{0i}b_{0i})^2 & \sum_1^n 2a_{0i}(b_{0i})^3 \\ \sum_1^n(a_{0i}b_{0i})^2 & \sum_1^n 2a_{0i}(b_{0i})^3 & \sum_1^n(b_{0i})^4 \end{pmatrix} \quad \text{(D.4)} \]

and

\[ b = \begin{pmatrix} (a_{01})^2 & (a_{02})^2 & \cdots & (a_{on})^2 \\ 2a_{01}b_{01} & 2a_{02}b_{02} & \cdots & 2a_{on}b_{on} \\ (b_{01})^2 & (b_{02})^2 & \cdots & (b_{on})^2 \end{pmatrix} \begin{pmatrix} 2(u_{01} - u_0 - (a_{01}\frac{\partial u_0}{\partial x} + b_{01}\frac{\partial u_0}{\partial y})) \\ 2(u_{02} - u_0 - (a_{02}\frac{\partial u_0}{\partial x} + b_{02}\frac{\partial u_0}{\partial y})) \\ \vdots \\ 2(u_{on} - u_0 - (a_{on}\frac{\partial u_0}{\partial x} + b_{on}\frac{\partial u_0}{\partial y})) \end{pmatrix} \]

\[ = \begin{pmatrix} \sum_1^n(a_{0i})^2(u_{0i} - u_0 - (a_{0i}\frac{\partial u_0}{\partial x} + b_{0i}\frac{\partial u_0}{\partial y})) \\ 4\sum_1^n a_{0i}b_{0i}(u_{0i} - u_0 - (a_{0i}\frac{\partial u_0}{\partial x} + b_{0i}\frac{\partial u_0}{\partial y})) \\ 2\sum_1^n(b_{0i})^2(u_{0i} - u_0 - (a_{0i}\frac{\partial u_0}{\partial x} + b_{0i}\frac{\partial u_0}{\partial y})) \end{pmatrix} \]

\[ = \begin{pmatrix} 2(\sum_1^n(a_{0i})^2 - B_{0i}\sum_1^n(a_{0j})^3 - A_{0i}\sum_1^n(a_{0j})^2b_{0j})(u_{0i} - u_0) \\ 4(\sum_1^n(a_{0i}b_{0i} - B_{0i}\sum_1^n(a_{0j})^2b_{0j} - A_{0i}\sum_1^n a_{0j}(b_{0j})^2)(u_{0i} - u_0)) \\ 2(\sum_1^n(b_{0i})^2 - B_{0i}\sum_1^n a_{0j}(b_{0j})^2 - A_{0i}\sum_1^n (b_{0j})^3)(u_{0i} - u_0) \end{pmatrix} \quad \text{(D.6)} \]
Let us introduce some notations,

\[ L_0i = 2((a_0i)^2 - B_0i \sum_1^n (a_{0j})^3 - A_0i \sum_1^n (a_{0j})^2b_{0j}), \quad (D.7) \]

\[ M_0i = 4(a_0ib_{0i} - B_0i \sum_1^n (a_{0j})^2b_{0j} - A_0i \sum_1^n a_{0j}(b_{0j})^2), \quad (D.8) \]

\[ N_0i = 2((b_{0i})^2 - B_0i \sum_1^n a_{0j}(b_{0j})^2 - A_0i \sum_1^n (b_{0j})^3). \quad (D.9) \]

Therefore,

\[ b = \begin{pmatrix} \sum_1^n L_0i(u_{0i} - u_0) \\ \sum_1^n M_0i(u_{0i} - u_0) \\ \sum_1^n N_0i(u_{0i} - u_0) \end{pmatrix}. \quad (D.10) \]

Solve \( Ax = b \), we can get \( \frac{\partial^2 u_0}{\partial x^2} \), \( \frac{\partial^2 u_0}{\partial x \partial y} \) and \( \frac{\partial^2 u_0}{\partial y^2} \). Since each term in \( A \) and \( b \) have is complicate, it will make the process of solving the system complicated too. We can simplify \( A \) and \( b \) by using a single letter to represent each term. Let us use the following simple forms for both \( A \) and \( b \).

\[ A = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}, \quad (D.11) \]

and

\[ b = \begin{pmatrix} r \\ s \\ t \end{pmatrix}. \quad (D.12) \]
\[
\begin{pmatrix}
  a & b & c & r \\
  b & d & e & s \\
  c & e & f & t \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  a & b & c & r \\
  0 & ad - b^2 & ae - bc & as - br \\
  0 & ae - bc & af - c^2 & at - cr \\
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  a & b & c & r \\
  0 & ad - b^2 & ae - bc & as - br \\
  0 & (af - c^2)(ad - b^2) & (at - cr)(ad - b^2) & (as - br)(ae - bc) \\
\end{pmatrix}.
\]

Hence,
\[
\frac{\partial^2 u_0}{\partial y^2} = \frac{(at - cr)(ad - b^2) - (as - br)(ae - bc)}{(af - c^2)(ad - b^2) - (ae - bc)^2}.
\]

Introducing
\[
C_{r,y^2} = \frac{b(ae - bc) - c(ad - b^2)}{(af - c^2)(ad - b^2) - (ae - bc)^2}, \quad (D.13)
\]
\[
C_{s,y^2} = -\frac{a(ae - bc)}{(af - c^2)(ad - b^2) - (ae - bc)^2}, \quad (D.14)
\]
\[
C_{t,y^2} = \frac{a(ad - b^2)}{(af - c^2)(ad - b^2) - (ae - bc)^2}. \quad (D.15)
\]

So,
\[
\frac{\partial^2 u_0}{\partial y^2} = C_{r,y^2}r + C_{s,y^2}s + C_{t,y^2}t
\]
\[
= C_{r,y^2} \sum_{i=1}^{n} L_0(u_{0i} - u_0) + C_{s,y^2} \sum_{i=1}^{n} M_0(u_{0i} - u_0) + C_{t,y^2} \sum_{i=1}^{n} N_0(u_{0i} - u_0)
\]
\[
= \sum_{i=1}^{n} (C_{r,y^2}L_0 + C_{s,y^2}M_0 + C_{t,y^2}N_0)(u_{0i} - u_0). \quad (D.16)
\]
By symmetry, we can write down the solution for \( \frac{\partial^2 u_0}{\partial x^2} \) and \( \frac{\partial^2 u_0}{\partial x \partial y} \).

\[
\frac{\partial^2 u_0}{\partial x^2} = C_{t,x}t + C_{s,x}s + C_{r,x}r
\]

\[
= C_{t,x} \sum_{i=1}^{n} N_{0i}(u_{0i} - u_0) + C_{s,y} \sum_{i=1}^{n} M_{0i}(u_{0i} - u_0) + C_{t,y} \sum_{i=1}^{n} L_{0i}(u_{0i} - u_0)
\]

\[
= \sum_{i=1}^{n} (C_{r,y}L_{0i} + C_{s,y}M_{0i} + C_{t,y}N_{0i})(u_{0i} - u_0),
\] (D.17)

where

\[
C_{t,x} = \frac{f(eb - fc) - c(ef - f^2)}{(ea - c^2)(ef - f^2) - (eb - fc)^2},
\] (D.18)

\[
C_{s,x} = -\frac{e(eb - fc)}{(ea - c^2)(ef - f^2) - (eb - fc)^2},
\] (D.19)

\[
C_{r,x} = \frac{e(ef - c^2)}{(ea - c^2)(ef - f^2) - (eb - fc)^2}.
\] (D.20)

\[
\frac{\partial^2 u_0}{\partial x \partial y} = C_{r,xy}r + C_{t,xy}t + C_{s,xy}s
\]

\[
= C_{t,x} \sum_{i=1}^{n} L_{0i}(u_{0i} - u_0) + C_{s,y} \sum_{i=1}^{n} N_{0i}(u_{0i} - u_0) + C_{t,y} \sum_{i=1}^{n} M_{0i}(u_{0i} - u_0)
\]

\[
= \sum_{i=1}^{n} (C_{r,y}L_{0i} + C_{s,y}M_{0i} + C_{t,y}N_{0i})(u_{0i} - u_0),
\] (D.21)

where

\[
C_{r,xy} = \frac{c(af - c^2) - b(af - c^2)}{(ad - b^2)(af - c^2) - (ae - cb)^2},
\] (D.22)

\[
C_{t,xy} = -\frac{a(af - c^2)}{(ad - b^2)(af - c^2) - (ae - cb)^2},
\] (D.23)

\[
C_{s,xy} = \frac{a(af - c^2)}{(ad - b^2)(af - c^2) - (ae - cb)^2}.
\] (D.24)
APPENDIX E

VALUE AT THE GHOST POINT FOR NEUMANN BOUNDARY CONDITION

Suppose $p_0$ is a grid point on the boundary of a domain $D$, and has $n$ neighbors $p_1, p_2, p_3, \ldots, p_n$, ordered counter clockwise. Let denote the vector from $p_0$ to $p_i$ by $\vec{v}_i$ ($1 \leq i \leq n$). For simplicity, assume the Neumann boundary condition is

$$\frac{\partial u}{\partial n} = 0 \text{ on } \partial\Omega.$$  \hspace{1cm} (E.1)

I want to add a neighbor to $p_0$, so that if I use DFDM to calculate $\frac{\partial u}{\partial n}|_{p_0}$, the result is 0.

Let call $p_0$’s new neighbor $p_{n+1}$. $\vec{v}_{n+1}$, the vector from $p_0$ to $p_{n+1}$, is normal to the domain boundary (pointing outward), see Figure 6.3. To simplify the notation we define

$$u_i = u(p_i), \; 0 \leq i \leq n + 1.$$  \hspace{1cm} (E.2)

$$\vec{v}_i = p_0p_i = (a_i, b_i), \; 1 \leq i \leq n + 1.$$  \hspace{1cm} (E.3)

Among all the notations, $u_{n+1}$ is the only unknown.

The over-determined linear system is

$$\begin{cases}
a_1 \frac{\partial u_0}{\partial x} + b_1 \frac{\partial u_0}{\partial y} = u_1 - u_0, \\
a_2 \frac{\partial u_0}{\partial x} + b_2 \frac{\partial u_0}{\partial y} = u_2 - u_0, \\
\vdots \\
a_{n+1} \frac{\partial u_0}{\partial x} + b_{n+1} \frac{\partial u_0}{\partial y} = u_{n+1} - u_0.
\end{cases}$$  \hspace{1cm} (E.4)
Therefore,

\[
A = \begin{pmatrix}
    a_1 & a_2 & \cdots & a_{n+1} \\
    b_1 & b_2 & \cdots & b_{n+1} \\
    & & \cdots & \\
    & & & a_{n+1} & b_{n+1}
\end{pmatrix}
= \begin{pmatrix}
    a_1 & b_1 \\
    a_2 & b_2 \\
    \vdots & \vdots \\
    a_{n+1} & b_{n+1}
\end{pmatrix}
= \begin{pmatrix}
    \sum_{i=1}^{n+1} a_i^2 & \sum_{i=1}^{n+1} a_ib_i \\
    \sum_{i=1}^{n+1} a_ib_i & \sum_{i=1}^{n+1} b_i^2
\end{pmatrix},
\]

(E.5)

\[
b = \begin{pmatrix}
    a_1 & a_2 & \cdots & a_{n+1} \\
    b_1 & b_2 & \cdots & b_{n+1} \\
    & & \cdots & \\
    & & & u_{n+1} - u_0
\end{pmatrix}
\begin{pmatrix}
    u_1 - u_0 \\
    u_2 - u_0 \\
    \vdots \\
    u_{n+1} - u_0
\end{pmatrix}
= \begin{pmatrix}
    \sum_{i=1}^{n+1} a_i(u_i - u_0) \\
    \sum_{i=1}^{n+1} b_i(u_i - u_0)
\end{pmatrix}.
\]

(E.6)

Solving \(Ax = b\), we have

\[
\begin{align*}
\frac{\partial u_0}{\partial x} &= \frac{\sum_{i=1}^{n+1} a_i(u_i - u_0) \sum_{j=1}^{n+1} b_j^2 - \sum_{i=1}^{n+1} b_i(u_i - u_0) \sum_{j=1}^{n+1} a_j b_j}{\sum_{i=1}^{n+1} a_i^2 \sum_{j=1}^{n+1} b_j^2 - (\sum_{i=1}^{n+1} a_i b_i)^2}, \\
\frac{\partial u_0}{\partial y} &= \frac{\sum_{i=1}^{n+1} a_i(u_i - u_0) \sum_{j=1}^{n+1} a_j b_j - \sum_{i=1}^{n+1} b_i(u_i - u_0) \sum_{j=1}^{n+1} a_j a_j}{(\sum_{i=1}^{n+1} a_i b_i)^2 - \sum_{i=1}^{n+1} a_i^2 \sum_{j=1}^{n+1} b_j^2}.
\end{align*}
\]

(E.7)

If we use the similar notations \(D\), \(A_i\) and \(B_i\) as previous \(D_0\), \(A_0\) and \(B_0\),

\[
D = \sum_{i=1}^{n+1} \sum_{j=i+1}^{n+1} (a_i b_j - a_j b_i)^2,
\]

(E.8)

\[
A_i = \frac{1}{D} \sum_{j=1}^{n+1} (a_j b_i - a_i b_j) a_j,
\]

(E.9)

\[
B_i = \frac{1}{D} \sum_{j=1}^{n+1} (a_i b_j - a_j b_i) b_j,
\]

(E.10)

we get

\[
\begin{align*}
\frac{\partial u_0}{\partial x} &= \sum_{i=1}^{n+1} B_i(u_i - u_0), \\
\frac{\partial u_0}{\partial y} &= \sum_{i=1}^{n+1} A_i(u_i - u_0).
\end{align*}
\]

(E.11)

To guarantee \(\frac{\partial u}{\partial n}|_{p_0} = 0\) \((\frac{\partial u}{\partial n}|_{p_0} = a_{n+1} \frac{\partial u_0}{\partial x} + b_{n+1} \frac{\partial u_0}{\partial y})\), the following must be
satisfied

\[ a_{n+1} \sum_{i=1}^{n+1} B_i (u_i - u_0) + b_{n+1} \sum_{i=1}^{n+1} A_i (u_i - u_0) = 0. \]  \hspace{1cm} (E.12)

Therefore,

\[ u_{n+1} = u_0 - \frac{\sum_{i=1}^{n} (a_{n+1}B_i + b_{n+1}A_i)(u^i_0 - u_0)}{a_{n+1}B_{n+1} + b_{n+1}A_{n+1}}. \]  \hspace{1cm} (E.13)
BIBLIOGRAPHY


