MODELLING A TWO-SPECIES INTERACTION IN A HIGH INTERTIDAL
ALGAL COMMUNITY

by

Nickolas Grabovac

B.Sc., University of British Columbia, 1995

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
(Department of Botany)

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

May 1997

© Nickolas Grabovac, 1997
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of **BOTANY**

The University of British Columbia
Vancouver, Canada

Date **May 15/97**
ABSTRACT

This study is an approach to simulating the population dynamics and species interactions in an intertidal algal community. The community being modelled is located in Barkley Sound, Vancouver Island and consists of three dominant perennial macroalgae, *Mazzaella cornicopiae, Fucus gardneri,* and *Pelvetiopsis limitata,* as well as some ephemeral algae and invertebrates such as barnacles and limpets. Interactions between the two species *Mazzaella* and *Fucus* were simulated in order to predict their population dynamics over a two-year period. The empirical basis for the simulation models was provided by the field experiments and observations of previous researchers. In total, eight different models were created, six of them being cellular automata (CA) models and the remaining two being birth-death difference equation models. These models were tested and validated using an independent set of field data. In general, all of the models were very poor predictors of *Fucus* density. Two of the models were excellent predictors of *Mazzaella* percent cover: a stochastic nonseasonal CA model and a *Mazzaella* birth-death model. The study suggests that simple, few-species models are able to predict the dynamics of some species, even when such species interact with many other species in the community. In addition, this study serves as an example of how many of the problems associated with constructing mathematical models of ecological systems, especially CA models, can be overcome and how a greater understanding of the system of interest can be gleaned from their results.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td>Table of Contents</td>
<td>iii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>List of Figures</td>
<td>viii</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>xiv</td>
</tr>
<tr>
<td><strong>CHAPTER 1: Thesis introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>General Introduction</td>
<td>1</td>
</tr>
<tr>
<td>The empirical basis for a CA model of a high intertidal community</td>
<td>3</td>
</tr>
<tr>
<td>Organization of the thesis</td>
<td>5</td>
</tr>
<tr>
<td><strong>CHAPTER 2: Cellular automata models of a two-species interaction in a high intertidal community</strong></td>
<td>8</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>8</td>
</tr>
<tr>
<td>METHODS</td>
<td>10</td>
</tr>
<tr>
<td>Types of cellular automata models constructed</td>
<td>10</td>
</tr>
<tr>
<td>Parameterization of the cellular automata models: The DIPS</td>
<td>11</td>
</tr>
<tr>
<td>Deterministic cellular automata</td>
<td>16</td>
</tr>
<tr>
<td>Stochastic cellular automata</td>
<td>18</td>
</tr>
<tr>
<td>Zero neighbour distance cellular automata</td>
<td>21</td>
</tr>
<tr>
<td>Cellular automata model implementation</td>
<td>22</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 2.1. The different cellular automata models constructed. ................................................. 40

Table 2.2. Deterministic, nonseasonal CA model state transition rules for Mazzaella increase and mortality and Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 40

Table 2.3. Deterministic, seasonal CA model state transition rules for Mazzaella increase and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 41

Table 2.4. Deterministic, seasonal CA model state transition rules for Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 41

Table 2.5. The season of the interval between each pair of field sampling dates (season of transition) used for parameterizing the seasonal CA models. ................................................................. 42

Table 2.6. Stochastic, nonseasonal CA model state transition rules (conditional probability functions) for Mazzaella increase and mortality and Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 42

Table 2.7. Stochastic, seasonal CA model state transition rules (conditional probability functions) for Mazzaella increase and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 43

Table 2.8. Stochastic, seasonal CA model state transition rules (conditional probability functions) for Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. \( p \) is the probability of the given state transition. .................................................................................................................. 43

Table 2.9. Zero Neighbour Distance, nonseasonal CA model state transition probabilities for Mazzaella increase and mortality and Fucus recruitment and mortality. ......................................................... 44

Table 2.10. Zero Neighbour Distance, seasonal CA model state transition probabilities for Mazzaella increase and mortality. ............................................................................................................. 44
Table 2.11. Zero Neighbour Distance, seasonal CA model state transition probabilities for *Fucus* recruitment and mortality. .................................................................................. 45

Table 2.12. Model predictive bias (calculated using the $W$ statistic) for all CA models. (Final refers to model predictions for the sampling date 24 months after the initial sample, and Previous refers to model predictions for the sampling date 22 months after the initial sample). .................................................................................. 46

Table 2.13. Comparison of CA model predictions to a naive zero-change prediction (calculated using Theil's inequality coefficient, $U^2$) for all CA models. (Final refers to model predictions for the sampling date 24 months after the initial sample, and Previous refers to model predictions for the sampling date 22 months after the initial sample). .................................................................................. 46

Table 3.1. Parameters for the two Birth-Death (BD) models. $\beta$ is the probability of recruitment and $\delta$ is the probability of mortality. .................................................................................. 91

Table 3.2. Birth-Death model predictive bias, calculated using the $W$ statistic, for 22 months after initial sampling and 24 months after initial sampling. ........................................ 91

Table 3.3. Birth-Death model predictions as compared with a naive zero-change prediction, calculated using Theil's inequality coefficient ($U^2$), for 22 months after initial sampling and 24 months after initial sampling. A $U^2 = 0$ indicates perfect predictions, $U^2 < 1$ indicates model predictions are better than a naive zero-change prediction, $U^2 > 1$ indicates the predictive power of the model is worse than a naive zero-change prediction. .................................................................................. 91
LIST OF FIGURES

Figure 1.1. Location of Kim's (1995) study sites (from Kim 1995)........................................6

Figure 1.2. A diagrammatic representation of the interactions among the major component organisms in Kim's (1995) community. PEL and FUC in the upper diagram indicate Pelvetiopsis and Fucus at the recruitment stage, while those in the lower indicate the fucoids after recruitment stage. Small letters are the abbreviations for types of interaction: (co) for competition, (fa) for facilitation, (gr) for grazing, (ov) for overgrowth, (pr) for preemption, (pt) for protection, (re) for refuge, (sh) for shading (Adapted from Kim 1995). ........................................7

Figure 2.1. The DIPS (Direct Identification of Probabilistic Synchronous Cellular Automata) algorithm (adapted from Adamatzky 1994).................................47

Figure 2.2. Seasonal recruitment probabilities for Fucus gardneri ........................................48

Figure 2.3. Seasonal mortality probabilities for Fucus gardneri ........................................49

Figure 2.4. Seasonal increase probabilities for Mazzaella cornucopiae ....................................50

Figure 2.5. Seasonal mortality probabilities for Mazzaella cornucopiae ....................................51

Figure 2.6. Transition Probability Vector θ_{M,Empty→Mazzaella}, the probability of Mazzaella increase for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = 0.0042 + 0.091(# Mazzaella neighbours), r^2 = 0.98. ......................................................52

Figure 2.7. Transition Probability Vector θ_{F,Empty→Mazzaella}, the probability of Mazzaella increase for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = 0.098 - 0.0030(# Fucus neighbours), r^2 = 0.16. ...........................................53
Figure 2.8. Transition Probability Vector $\theta_{M,Mazzaella\rightarrow Empty}$, the probability of Mazzaella mortality for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.61 - 0.073(\# Mazzaella$ neighbours), $r^2 = 0.99$.

Figure 2.9. Transition Probability Vector $\theta_{F,Mazzaella\rightarrow Empty}$, the probability of Mazzaella mortality for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.12 + 0.0063(\# Fucus$ neighbours), $r^2 = 0.62$.

Figure 2.10. Transition Probability Vector $\theta_{M,Empty\rightarrow Fucus}$, the probability of Fucus recruitment for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.049 + 0.00027e^{(# Mazzaella$ neighbours}), $r^2 = 0.97$.

Figure 2.11. Transition Probability Vector $\theta_{F,Empty\rightarrow Fucus}$, the probability of Fucus recruitment for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.012 + 0.0036e^{(# Fucus$ neighbours}), $r^2 = 0.88$.

Figure 2.12. Transition Probability Vector $\theta_{M,Fucus\rightarrow Empty}$, the probability of Fucus mortality for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.33 - 0.033(\# Mazzaella$ neighbours), $r^2 = 0.68$.

Figure 2.13. Transition Probability Vector $\theta_{F,Fucus\rightarrow Empty}$, the probability of Fucus mortality for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.20 + 0.0025(\# Fucus$ neighbours), $r^2 = 0.008$.

Figure 2.14. Percent cover of Mazzaella predicted by the six different CA models and the field data from the final sampling dates of the three control test quadrats, C-2 to C-4. The $x$ denotes no field data available for the final sampling date for that quadrat.

Figure 2.15. Percent cover of Mazzaella predicted by the six different CA models and the field data from the final sampling dates of the three Mazzaella-removal test quadrats, I-2 to I-4.
Figure 2.16. Percent cover of Mazzaella predicted by the six different CA models and the field data from the final sampling dates of the three Fucus-removal test quadrats, F-2 to F-4. ......................................................... 62

Figure 2.17. Percent cover of Mazzaella predicted by the six different CA models and the field data from the final sampling dates of the three Pelvetiopsis-removal test quadrats, P-2 to P-4. .................................................................. 63

Figure 2.18. Percent cover of Mazzaella predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three control test quadrats, C-2 to C-4. ............................................................... 64

Figure 2.19. Percent cover of Mazzaella predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three Mazzaella-removal test quadrats, I-2 to I-4. ............................................................... 65

Figure 2.20. Percent cover of Mazzaella predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three Fucus-removal test quadrats, F-2 to F-4. ............................................................... 66

Figure 2.21. Percent cover of Mazzaella predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three Pelvetiopsis-removal test quadrats, P-2 to P-4. ............................................................... 67

Figure 2.22. Density of Fucus predicted by the six different CA models and the field data from the final sampling dates of the three control test quadrats, C-2 to C-4. The x denotes no field data available for the final sampling date for that quadrat. .............................................................................................................. 68

Figure 2.23. Density of Fucus predicted by the six different CA models and the field data from the final sampling dates of the three Mazzaella-removal test quadrats, I-2 to I-4. ............................................................... 69

Figure 2.24. Density of Fucus predicted by the six different CA models and the field data from the final sampling dates of the three Fucus-removal test quadrats, F-2 to F-4. ............................................................... 70

Figure 2.25. Density of Fucus predicted by the six different CA models and the field data from the final sampling dates of the three Pelvetiopsis-removal test quadrats, P-2 to P-4. ............................................................... 71

Figure 2.26. Density of Fucus predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three control test quadrats, C-2 to C-4. ............................................................... 72
Figure 2.27. Density of Fucus predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three Mazzaella-removal test quadrats, I-2 to I-4. .......................................................... 73

Figure 2.28. Density of Fucus predicted by the six different CA models and the field data from samples just prior to the final sampling dates of the three Fucus-removal test quadrats, F-2 to F-4. .......................................................... 74

Figure 2.29. Density of Fucus predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three Pelvetiopsis-removal test quadrats, P-2 to P-4. .......................................................... 75

Figure 2.30. Observed vs. Predicted plot of Mazzaella percent cover for the deterministic CA models. EF = modelling efficiency.  
A. Deterministic, nonseasonal CA, 24 months after initial sampling, EF = 0.69.  
B. Deterministic, nonseasonal CA, 22 months after initial sampling, EF = 0.86.  
C. Deterministic, seasonal CA, 24 months after initial sampling, EF = 0.67.  
D. Deterministic, seasonal CA, 22 months after initial sampling, EF = 0.87. ....... 76

Figure 2.31. Observed vs. Predicted plot of Mazzaella percent cover for the stochastic CA models. EF = modelling efficiency.  
A. Stochastic, nonseasonal CA, 24 months after initial sampling, EF = 0.37.  
B. Stochastic, nonseasonal CA, 22 months after initial sampling, EF = 0.92.  
C. Stochastic, seasonal CA, 24 months after initial sampling, EF = -0.19.  
D. Stochastic, seasonal CA, 22 months after initial sampling, EF = 0.70. .......... 77

Figure 2.32. Observed vs. Predicted plot of Mazzaella percent cover for the zero neighbour distance (ZND) CA models. EF = modelling efficiency.  
A. ZND, nonseasonal CA, 24 months after initial sampling, EF = 0.10.  
B. ZND, nonseasonal CA, 22 months after initial sampling, EF = 0.63.  
C. ZND, seasonal CA, 24 months after initial sampling, EF = 0.37.  
D. ZND, seasonal CA, 22 months after initial sampling, EF = 0.53. .......... 78

Figure 2.33. Observed vs. Predicted plot of Fucus density for the deterministic CA models. EF = modelling efficiency.  
A. Deterministic, nonseasonal CA, 24 months after initial sampling, EF = -0.94.  
B. Deterministic, nonseasonal CA, 22 months after initial sampling, EF = -0.79.  
C. Deterministic, seasonal CA, 24 months after initial sampling, EF = -0.74.  
D. Deterministic, seasonal CA, 22 months after initial sampling, EF = -0.77. ...... 79
Figure 2.34. Observed vs. Predicted plot of *Fucus* density for the stochastic CA models. EF = modelling efficiency.
A. Stochastic, nonseasonal CA, 24 months after initial sampling, EF = -0.88.
B. Stochastic, nonseasonal CA, 22 months after initial sampling, EF = -0.59.
C. Stochastic, seasonal CA, 24 months after initial sampling, EF = -0.72.
D. Stochastic, seasonal CA, 22 months after initial sampling, EF = -0.36.

Figure 2.35. Observed vs. Predicted plot of *Fucus* density for the zero neighbour distance (ZND) CA models. EF = modelling efficiency.
A. ZND, nonseasonal CA, 24 months after initial sampling, EF = -1.38.
B. ZND, nonseasonal CA, 22 months after initial sampling, EF = -0.13.
C. ZND, seasonal CA, 24 months after initial sampling, EF = -1.46.
D. ZND, seasonal CA, 22 months after initial sampling, EF = -0.14.

Figure 3.1. Percent cover of *Mazzaella* predicted by the *Mazzaella* Birth-Death model and the field data from the final sampling dates of the 12 test quadrats. The x denotes no field data available for the final sampling date for that quadrat. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella*-removal quadrats, F-2 to F-4 are *Fucus*-removal quadrats and P-2 to P-4 are *Pelvetiopsis*-removal quadrats.

Figure 3.2. Percent cover of *Mazzaella* predicted by the *Mazzaella* Birth-Death model and the field data from sampling dates just prior to the final sampling dates of the 12 test quadrats. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.

Figure 3.3. Density of *Fucus* predicted by the *Fucus* Birth-Death model and the field data from the final sampling dates of the 12 test quadrats. The x denotes no field data available for the final sampling date for that quadrat. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.

Figure 3.4. Density of *Fucus* predicted by the *Fucus* Birth-Death model and the field data from sampling dates just prior to the final sampling dates of the 12 test quadrats. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.

Figure 3.5. Observed vs. predicted plot of *Mazzaella* percent cover, 24 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = 0.46 (EF = modelling efficiency).
Figure 3.6. Observed vs. predicted plot of *Mazzaella* percent cover, 22 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = 0.92 (EF = modelling efficiency). 97

Figure 3.7. Observed vs. predicted plot of *Fucus* density, 24 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = -2.2. (EF = modelling efficiency). 98

Figure 3.8. Observed vs. predicted plot of *Fucus* density, 22 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = -0.085. (EF = modelling efficiency). 99
ACKNOWLEDGMENTS

I would like to thank my teacher, supervisor and long time mentor, Dr. Robert E. DeWreede, for his support, encouragement, good humour, open mind and interest in mathematics. This study was funded by a Natural Science and Engineering Research Council of Canada PGS A scholarship. My thanks also go to my committee members, Drs. Gary Bradfield and Paul G. Harrison, for their critical suggestions and editorial comments on this thesis. I am grateful to Mark J. Kusec, for providing me with the software tools I needed to complete this project, as well as his friendship, helpful suggestions and assistance with debugging. I thank my colleagues in Dr. DeWreede’s lab, Ricardo Scrosati, Kristen Drewes, Russell Markel, Tania Thenu, Andrea Sussman and Leonard Dyck, for stimulating discussions, help and friendship.

I am very grateful for the love and support of my wife, Andrea Grabovac, without whom I probably never would have gotten my act together. Her help with organization and patience during some of the harrier days, even when she was tired and post-call, allowed me to complete this undertaking in a reasonable amount of time. I would like to thank my parents, Drago and Jelena Grabovac as well as my parents-in-law, Dagmar and Joseph Kalousek, for their understanding, encouragement, wisdom, and many delicious, hot meals during my study at UBC.
CHAPTER 1

THESIS INTRODUCTION

General Introduction

In 1948, von Neumann proposed that the survival, reproduction and evolution of complex forms of organization could be explained, in principle, by the interactions of many identical and very simple primitives (i.e., fundamental, low-level building blocks) and their resulting dynamics (von Neumann 1966). This ambitious goal gave rise to the theory of cellular automata. Since that time, cellular automata theory has experienced long periods of dormancy, punctuated by periods of increased interest and new developments which closely followed increases in the power of computer hardware (Hogeweg 1988).

Informally, cellular automata (also called tessellation automata) are discrete dynamical systems which are represented by a large tessellation (i.e., grid or lattice) of identical finite-state automata (called cells). There exists a large body of literature on the theory of finite-state automata (also known as finite-state machines) (see Bavel 1983 for an introduction). For the purposes of this thesis, the individual cells of the cellular automata (CA) are represented by a discrete number (usually small) of finite states which evolve in discrete time steps. The state of any given cell in the future is a function of its current state and the states of the surrounding cells in a defined neighbourhood. Common neighbourhoods used in the study of CA are the four nearest neighbours (von
Neumann neighbourhood) and the eight neighbours (Moore neighbourhood) on a square lattice. The function which determines the state of a given cell based on its neighbourhood configuration is often termed the next-state function, while the individual neighbourhood configurations which give rise to particular states are called the rules of the CA. CA models are discrete in space, state and time, and thus represent an alternative to more traditional differential equation approaches (which are continuous) to modelling.

Cellular automata have become popular again in the last decade, especially as a way of modelling physical systems (Vichniac 1984). CA as a modelling approach have also recently received a great deal of interest by biologists (Ermentrout and Edelstein-Keshet 1993, Hogeweg 1988 for reviews). Some examples of applications of CA to biological problems are modelling DNA sequences (Burks and Farmer 1984), as spatially explicit population models for a wide variety of populations in ecology (grass and clover populations, Schwinning and Parsons, 1996; wild daffodil populations, Barkham and Hance 1982; competition between annual and perennial plant species, Crawley and May 1987; metapopulation dynamics of annual plants with a seedbank, Perry and Gonzalez-Andujar 1993; effect of spatial patterns on interspecific competition, Silvertown et al. 1992) and as models of resource dynamics in plant populations (Colosanti and Grime 1993). What all of these models have in common is the use of CA to answer questions about how population dynamics are altered when various attributes of the environment are manipulated. This is one type of investigation for which CA models are ideally suited (Durrett and Levin 1993).
CA models have many advantages over other modeling approaches. First, CA are explicitly spatial and as such, they are ideally suited to modeling the spatial dynamics of organisms. Indeed, in systems where the interactions between individuals are local and there is spatial heterogeneity in species distributions and/or environmental conditions, models which are able to explicitly incorporate space are considered to be a necessity (Caswell and John 1992, Judson 1994). Secondly, CA models are constructed from the “bottom-up” because they explicitly incorporate the behaviours and states of individuals in the system. It is the interaction of these individuals which then determine the dynamics of the system. This is useful because it allows the modeler to reproduce in quantitative detail the results of an experimental manipulation (Ermentrout and Edelstein-Keshet 1993).

The empirical basis for a CA model of a high intertidal community

From 1991 to 1994, Kim (1996 and 1995) experimentally studied two rocky high intertidal communities at Prasiola Point and Nudibranch Point, located in Barkley Sound on the west coast of Vancouver Island, British Columbia, Canada ($48^049'$ N, $125^010'$ W) near the Bamfield Marine Station (Fig. 1.1). The community he studied was composed of three dominant perennial macroalgae: *Fucus distichus* L., *Pelvetiopsis limitata* Gard (Phaeophyta), and *Mazzaella cornucopiae* Hommersand (Rhodophyta), as well as some ephemeral algae. In addition, invertebrates such as barnacles, limpets, and snails were also present. Kim examined many biotic and abiotic factors affecting the high intertidal
community such as competition, facilitation, herbivory, and disturbance, and proposed a complex set of interactions to explain the dynamics of the community (Fig. 1.2). Such a conceptual model is an excellent starting point for developing mathematical and computer models to generate testable hypotheses and predictions. In particular, because Kim collected detailed, small-scale, spatial data on his community, his work is an ideal empirical foundation for the construction and quantitative testing of a CA model. In addition, Kim’s experimental results stressed that local, spatial interactions between members of the community were very important in determining the dynamics of the system (Kim, 1995), and this provided further incentive for using CA to model this system.

The main objective of this thesis was to use the experimental results from Kim’s work to construct a series of CA models of Kim’s high intertidal community in order to simulate the population dynamics of *Mazzaella cornucopiae* and *Fucus gardneri*. This goal was important for two reasons. Firstly, the ability to accurately simulate population dynamics is one of the cornerstones of applied ecology and is important for the management and conservation of populations. Secondly, the use of CA models as simulation tools has never before been attempted in ecology. Thus, using CA models in this way is a genuinely new modelling approach with much potential, and if successful in this study, could prove to be a useful tool for the simulation of other populations.
Organization of the Thesis

This thesis consists of 5 chapters: an introductory chapter, two research chapters, a concluding chapter and an appendix.

In Chapter 2, the construction of six different CA models and their predictions are described. The different model predictions are tested against field data and are compared and ranked in terms of predictive ability.

Chapter 3 examines the results from a simple birth-death model as an alternative to the CA models of chapter 2. Predictions from the birth-death model are also tested against field data.

Chapter 4 discusses the approach of constructing multiple models of a system as multiple competing hypotheses and ranks all the models created to determine the overall best model. In addition, the implications of constructing CA models as simulation tools in ecology are discussed, as well as some of the problems with the CA approach and how they were resolved in this thesis.

The appendix contains the source code for one of the CA models, as an example of how such models are actually implemented.
Figure 1.1. Location of Kim's (1995) study site. B = Prasiola Point.
Figure 1.2. A diagrammatic representation of the interactions among the major component organisms in Kim's (1995) community. PEL and FUC in the upper diagram indicate *Pelvetiopsis* and *Fucus* at the recruitment stage, while those in the lower indicate the fucoids after recruitment stage. Small letters are the abbreviations for types of interaction: (co) for competition, (fa) for facilitation, (gr) for grazing, (ov) for overgrowth, (pr) for preemption, (pt) for protection, (re) for refuge, (sh) for shading. (Adapted from Kim 1995).
CHAPTER 2

CELLULAR AUTOMATA MODELS OF A TWO-SPECIES INTERACTION IN A HIGH INTERTIDAL COMMUNITY

INTRODUCTION

There has been an explosion of interest in recent years in the use of cellular automata models in ecology (Ermentrout and Edelstein-Keshet 1993). This is largely the result of frustration with more traditional modelling approaches which assumed a homogenous environment, and thus did not account for spatial heterogeneity. In addition, the availability of relatively cheap, yet computationally powerful, computers has allowed the cellular automata approach to become practical (Judson 1994, DeAngelis and Gross 1992, Huston et al. 1988), and often much simpler and more intuitive for non-mathematically minded ecologists.

The majority of cellular automata (CA) models of ecological systems to date are what can be called 'what-if' models because they attempt to answer general questions about the types of global behaviours that can be expected given certain local interaction rules between the organisms in the model. Such models are excellent for specifying limitations on the possible causes for the patterns observed in nature, but they are not able to tell us whether or not the rules specified in the model are indeed the particular mechanisms which are acting in nature. The goal of this approach is to eventually arrive at general ecological laws which can be used to explain the distribution and abundance of organisms
through time. Although many such models have been constructed, we are still very far
from the goal.

A different approach is to use CA models to simulate a system as exactly as possible.
In this case, we trade the generality of the "what-if" approach for the concreteness of a
real system existing in nature. The goal here is to be able to quantitatively predict the
distribution and abundance of the organisms in the particular system of interest at
particular times. Such models can be extremely useful tools for the management of
specific populations and are useful for answering questions about the effects of specific
changes in a particular system on the organisms inhabiting that system. To date, this
approach has hardly been explored at all in ecology, possibly due to the preponderance of
"what-if" CA models. This is unfortunate since CA are ideal models for simulation in
ecology, as well as other fields such as physics where they are being used with increasing
regularity (Vichniac 1984).

The purpose of this study was to use the cellular automata approach to construct
simulation models of Kim's high intertidal community in order to predict the population
dynamics of Mazzaella cornucopiae and Fucus gardneri, the two dominant macrophytes
in the community. Several different CA models were constructed to provide a set of
competing models from which the best model (i.e., the one which most accurately
predicted the dynamics of Mazzaella and Ficus) could be chosen.
METHODS

Types of Cellular Automata Models Constructed

Six different cellular automata (CA) models were constructed in an attempt to predict the population dynamics of Mazzaella and Fucus (Table 2.1). These models were divided into three categories: 1) deterministic, 2) stochastic and 3) zero-neighbour distance (ZND) models. Each of these three categories of CA model was further divided into a seasonal and a nonseasonal model. The deterministic models had transition rules which occurred with a probability of 1.0, thus making the behaviour of this class of models completely non-random. The stochastic models, on the other hand, had a probability associated with each transition rule in the model, thus introducing a random component to this class of models. The ZND models were also stochastic (i.e., had probabilistic transition rules) but the neighbourhood size of each cell in the model was zero. In other words, in the ZND models, the behaviour of each cell was not dependent on the state of the cells surrounding it. Thus, a ZND model is a CA model without local interactions present. Nonseasonal models were constructed in such a way that the transition rules did not depend on the season, whereas seasonal models had transition rules which changed depending on what season it was. All of the CA models had three possible states for each cell: Mazzaella, Fucus and empty.

Transition rules and testing for each of the six CA models were estimated from 16 permanent quadrats of field data, sampled bimonthly over a two-year period by Kim (1995). These data consisted of 20 by 20 cm quadrats which were divided into 400 one cm² cells. During each sampling date, Kim recorded whether one or more of the species
Mazzaella cornucopiae, Fucus gardneri and Pelvetiopsis limitata were present in each cell of each of the 16 permanent quadrats. These quadrats thus formed maps of the three dominant macrophytes in the community every two months for a period of two years.

In order to have independent data sets for both parameterization and testing, it was necessary to split the data pool into two smaller subsets of data. The first subset was used for estimating the transition rules for the various CA models and consisted of four of the 16 permanent quadrats. Each of these four parameterization quadrats represented a different treatment in Kim's (1995) field work: a control quadrat (no treatment applied), a Mazzaella removal quadrat, a Fucus removal quadrat and a Pelvetiopsis removal quadrat. The second subset was used for testing the CA models and consisted of 12 of the 16 permanent quadrats. These 12 quadrats consisted of three quadrats of each of the four treatments described above.

**Parameterization of the Cellular Automata Models: The DIPS algorithm**

All of the transition rules for the different CA models were estimated from the four parameterization quadrats using the DIPS (Direct Identification of Probabilistic Synchronous Cellular Automata) algorithm, developed by Adamatzky (1992, 1994) (Fig. 2.1). DIPS treats each quadrat map as a snapshot of the global state of a CA model with unknown rules. Each cell of the parameterization quadrats is treated as if it is a cell in a stochastic CA.

DIPS begins by examining the 'state' of a given cell for the first sampling date of a given quadrat. In this case, the 'state' of the quadrat cell can be Mazzaella, Fucus, or
empty (bare rock). In the case where a quadrat cell contained more than one species, the extra species was removed by consulting a random number table. If the number from the table was between zero and four, inclusive, then species A would be the 'state' of that cell, otherwise species B would be the 'state' of that cell. DIPS then looks at the neighbourhood of the cell of interest (i.e., examines the states of all of the cells adjacent to the cell of interest). Because the CA models being constructed have three states: *Mazzaella*, *Fucus* and empty (bare rock), there are 45 possible neighbourhood configurations of these three states, where a neighbourhood is defined as every cell adjacent to the cell of interest (i.e., the 8 cells surrounding a given cell). Each neighbourhood configuration is identified by the number of *Mazzaella* and *Fucus* cells present in it, ignoring the exact spatial ordering of each of the cells around the given cell. Examples of neighbourhood configurations are three *Mazzaella* and one *Fucus*, five *Mazzaella* and zero *Fucus* and one *Mazzaella* and four *Fucus*. Once DIPS identifies the neighbourhood configuration of a given cell, it then increments a counter representing the number of times that particular configuration has been encountered. DIPS uses the value of the counter to create a Neighbourhood Configuration Matrix, $N$, where each element of the matrix corresponds to the number of occurrences of each neighbourhood configuration in the quadrat being examined.

For the CA being constructed here, $N$ is a 9x9 matrix with the rows representing the number of *Mazzaella* in the neighbourhood and the columns representing the number of *Fucus* in the neighbourhood (equation (2.1)).
number of *Fucus* neighbours

\[
N = \begin{bmatrix}
  n_{0,0} & n_{0,1} & \cdots & \cdots & n_{0,8} \\
  n_{1,0} & n_{1,1} & \cdots & \cdots & n_{1,7} & 0 \\
  \vdots & \vdots & \ddots & & \vdots & \vdots & n_{2,6} & 0 & 0 \\
  & \ddots & n_{3,5} & 0 & 0 & 0 & 0 \\
  & & n_{4,4} & 0 & 0 & 0 & 0 & 0 \\
  \vdots & \vdots & & n_{5,3} & 0 & 0 & 0 & 0 & 0 \\
  & \vdots & \vdots & \vdots & n_{6,2} & 0 & 0 & 0 & 0 & 0 \\
  & \vdots & \vdots & \vdots & \vdots & n_{7,1} & 0 & 0 & 0 & 0 & 0 \\
  n_{8,0} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

where \( N \) is the Neighbourhood Configuration Matrix and the \( n_{r,c} \) represent the number of occurrences of the neighbourhood configuration with \( r \) *Mazzaella* neighbours and \( c \) *Fucus* neighbours.

This matrix is used later to calculate the Transition Probability Matrices (see below). DIPS then takes the next sampling date for the same quadrat and examines the 'state' of the same cell. In this way, DIPS determines the transition which is taking place during the time interval from sample \( t \) to sample \( t+1 \). After identifying the transition, DIPS increments another counter, representing the number of times that particular state transition has occurred with a particular neighbourhood configuration. DIPS uses this counter to create a series of State Transition Matrices, \( T_i \) (one for each possible state transition), where the elements of the matrices correspond to the number of occurrences of a given transition for a particular neighbourhood configuration.

In the case of the CA models being constructed here, there were four State Transition Matrices, one for each of the following transitions: 1) *Mazzaella* to empty, 2) empty to
Mazzaella, 3) Fucus to empty and 4) empty to Fucus\(^1\). Each of these State Transition Matrices were 9x9 matrices, with the rows representing the number of Mazzaella cells in the neighbourhood and the columns representing the number of Fucus cells in the neighbourhood (equation (2.2)).

\[
T_i = \begin{bmatrix}
 t_{0,0} & t_{0,1} & \cdots & \cdots & t_{0,8} \\
 t_{1,0} & t_{1,1} & \cdots & \cdots & t_{1,7} & 0 \\
 \vdots & \vdots & \ddots & \ddots & t_{2,6} & 0 & 0 \\
 \vdots & \vdots & \ddots & t_{3.5} & 0 & 0 & 0 \\
 t_{4,4} & 0 & 0 & 0 & 0 \\
 t_{5,3} & 0 & 0 & 0 & 0 & 0 \\
 t_{6.2} & 0 & 0 & 0 & 0 & 0 \\
 t_{7.1} & 0 & 0 & 0 & 0 & 0 & 0 \\
 t_{8.0} & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

where \(T_i\) is the State Transition Matrix for transition \(i\) and the \(t_{r,c}\) represent the number of occurrences of transition \(i\) with \(r\) Mazzaella neighbours and \(c\) Fucus neighbours.

This process is repeated, with DIPS moving from one cell to another in the quadrat map, until it reaches the last cell for that particular quadrat. At this point, DIPS estimates the Transition Probability Matrices, \(P_i\) (one for each possible transition), the elements of which represent the probability of a given transition occurring for a particular neighbourhood configuration. Each element in the \(P_i\) is calculated by dividing the corresponding element in \(T_i\) with the corresponding element in \(N\) (i.e., by dividing the

\(^1\) The state transitions Mazzaella to Fucus and Fucus to Mazzaella were not included in the CA models because these transitions occurred so rarely in the data set that they were omitted in order to simplify the models. The transition from a given state to itself, such as Mazzaella to Mazzaella, Fucus to Fucus and Empty to Empty, were not estimated, since they represent no-change conditions which are easily incorporated into the model during coding (with no loss of information) and thus did not require explicit estimation of their probabilities.
number of times a given transition was encountered by the number of times its particular neighbourhood configuration was encountered). Thus for the CA models being constructed here, the $P_i$ consist of four 9x9 matrices, with the rows and columns the same as for the $T_j$ (equation (2.3)).

$$
P_i = \begin{bmatrix}
    t_{0,0} & t_{0,1} & \cdots & \cdots & t_{0,8} \\
    n_{0,0} & n_{0,1} & \cdots & \cdots & n_{0,8} \\
    t_{1,0} & t_{1,1} & \cdots & \cdots & t_{1,7} \\
    n_{1,0} & n_{1,1} & \cdots & \cdots & n_{1,7} \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    t_{3,5} & \cdots & \cdots & t_{5,3} & \cdots \\
    n_{3,5} & \cdots & \cdots & n_{5,3} & \cdots \\
    t_{4,4} & \cdots & \cdots & t_{6,2} & \cdots \\
    n_{4,4} & \cdots & \cdots & n_{6,2} & \cdots \\
    \vdots & \vdots & \ddots & \vdots & \vdots \\
    t_{7,1} & \cdots & \cdots & t_{8,0} & \cdots \\
    n_{7,1} & \cdots & \cdots & n_{8,0} & \cdots 
\end{bmatrix}
$$

where $P_i$ is the Transition Probability Matrix for transition $i$ and the $p_{r,c}$ represent the probability of transition $i$ with $r$ Mazzaella neighbours and $c$ Fucus neighbours.

DIPS then takes the next pair of samples and repeats the entire process, checking to see if the transition probabilities satisfy a Cauchy criterion (i.e., that each transition probability converges to a stable value). If this criterion is satisfied, the algorithm stops.
Otherwise, the neighbourhood radius is increased by one and the entire process is repeated again. DIPS continues to increase the size of the neighbourhood radius until either the transition probabilities converge, or the size of the neighbourhood is too large to fit within the quadrat.

Because the quadrat maps did not contain enough data to obtain reliable estimates of transition probabilities from a single pair of samples (see Chapter 4), a modified form of DIPS was used. In the modified form of DIPS, all the sample pairs were enumerated for neighbourhood configurations and state transitions before calculating the transition probabilities, thus pooling the entire two-year data set. In addition, a neighbourhood radius of one was used, even though only 60% of the transition probabilities had converged at this radius.

In order to use DIPS to estimate the transition rules in the CA models being constructed, a computer program was written in C++ (Borland, v4.5) to implement the DIPS algorithm. Data for the four parameterization quadrats were entered into a spreadsheet in Microsoft Excel (v 5.0) and then converted to a text file. Another program was made to read the quadrat maps from the text file and send the data to the DIPS program. All output from DIPS was then saved in another text file to be analyzed later.

**Deterministic Cellular Automata**

In order to construct the deterministic CA models, the Transition Probability Matrices output from DIPS were examined and approximated by the deterministic rules shown in
Table 2.2. These rules were used as the state transition rules for the nonseasonal deterministic CA model. In order to construct the seasonal model, the DIPS output was pooled by season to produce seasonal probabilities of recruitment and mortality for *Fucus* (Figs. 2.2 and 2.3) and seasonal probabilities of increase and mortality for *Mazzaella* (Figs. 2.4 and 2.5). Thus, if a transition from empty to *Mazzaella* occurred during the summer, for example, it would be lumped together with all of the other summer empty to *Mazzaella* transitions in order to determine the probability of *Mazzaella* increase in the summer. Because the ‘transitions’ in community structure measured by Kim in the field occurred during the time intervals between successive sampling dates, the transitions in the CA models were timed to occur in the same time intervals. Thus, for the seasonal deterministic CA model (and also for all of the other seasonal CA models), it was necessary to assign a season to each of the time intervals between each pair of successive sampling dates. When both successive sampling dates occurred in the same season, the season assigned to the interval between the sampling dates was simply the season during which both field samples were taken. When the successive sampling dates occurred during different seasons, a method was required in order to assign a season to the interval between the pair of successive sampling dates. This method is shown in Table 2.5. Thus, over a two-year period, with each time interval in the model representing two months of real time, there were 4 spring, 3 summer, 3 fall and 2 winter seasons for each of the 12 test quadrats (Table 2.5).
The seasonal transition probabilities were incorporated into the seasonal deterministic CA model by approximating the probabilities using a deterministic rule. The rules used in the seasonal model are shown in Tables 2.3 and 2.4.

**Stochastic Cellular Automata**

The raw output from DIPS was not used directly for the transition probabilities of the stochastic CA models, since this would have required that each model have 180 parameters (four transitions, each with 45 different transition probabilities corresponding to the different neighbourhood configurations). To simplify the models the transition probabilities were pooled by reducing the number of neighbourhood configurations from 45 to 9 for each given transition, using equations (2.4) to (2.9):

\[
\eta_M = N z
\]  \hspace{1cm} (2.4)

where \( \eta_M \) is the Neighbourhood Configuration Vector for the different neighbourhood configurations of \textit{Mazzaella} (i.e., pooled across all \textit{Fucus} configurations) and \( z = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \).

\[
\eta_F = N^t z
\]  \hspace{1cm} (2.5)

where \( \eta_F \) is the Neighbourhood Configuration Vector for the different neighbourhood configurations of \textit{Fucus} (i.e., pooled across all \textit{Mazzaella} configurations) and \( N^t \) is the transpose of \( N \).
\[ \tau_M = \mathbf{T}_i z \]  

(2.6)

where \( \tau_M \) is the State Transition Vector for the different neighbourhood configurations of Mazzaella (i.e., pooled across all Fucus configurations).

\[ \tau_F = \mathbf{T}'_i z \]  

(2.7)

where \( \tau_F \) is the State Transition Vector for the different neighbourhood configurations of Fucus (i.e., pooled across all Mazzaella configurations) and \( \mathbf{T}'_i \) is the transpose of \( \mathbf{T}_i \).

\[ \theta_M = \begin{bmatrix} \tau_{M,0} / \eta_{M,0} & \tau_{M,1} / \eta_{M,1} & \ldots & \tau_{M,8} / \eta_{M,8} \end{bmatrix} \]  

(2.8)

where \( \theta_M \) is the Transition Probability Vector for the different neighbourhood configurations of Mazzaella for transition \( i \).

\[ \theta_F = \begin{bmatrix} \tau_{F,0} / \eta_{F,0} & \tau_{F,1} / \eta_{F,1} & \ldots & \tau_{F,8} / \eta_{F,8} \end{bmatrix} \]  

(2.9)

where \( \theta_F \) is the Transition Probability Vector for the different neighbourhood configurations of Fucus for transition \( i \).

Regressions were performed on the Transition Probability Vectors to give conditional probability functions, in order to simplify the construction of the stochastic CA even further, and also to allow for easier modification of the models. The Transition Probability Vectors for each stochastic CA model transition, and their corresponding conditional probability functions are shown in Figures 2.6 to 2.13.
An examination of Figures 2.6 and 2.7 revealed that the probability of *Mazzaella* increase was dependent on the number of *Mazzaella* neighbours in its neighbourhood, and that it appeared to be largely independent of the number of *Fucus* neighbours. For this reason, only the conditional probability function for *Mazzaella* increase for a given neighbourhood configuration of *Mazzaella* was used in the model. Although the probability of *Mazzaella* mortality was dependent on both the numbers of *Mazzaella* and *Fucus* neighbours in its neighbourhood, the dependence of *Mazzaella* mortality on *Mazzaella* largely outweighed its dependence on *Fucus* (Figs. 2.8 and 2.9). Thus, only the conditional probability function for *Mazzaella* mortality for a given neighbourhood configuration of *Mazzaella* was used in the model.

There was a strong relationship between the number of *Mazzaella* neighbours and the probability of recruitment for *Fucus* (Fig. 2.10), resulting in a range of recruitment probabilities from 0.05 to 0.8. Although there was a relationship between the number of *Fucus* neighbours and the probability of *Fucus* recruitment (Fig. 2.11), the range of recruitment probabilities was only from 0.02 to 0.09. The number of *Mazzaella* neighbours thus accounted for much greater magnitude changes in *Fucus* recruitment probabilities than the number of *Fucus* neighbours. For this reason, only the conditional probability function for *Fucus* recruitment for a given neighbourhood configuration of *Mazzaella* was used in the stochastic CA models. Although the probability of *Fucus* mortality was dependent on both the numbers of *Mazzaella* and *Fucus* neighbours in its neighbourhood, the dependence of *Fucus* mortality on *Mazzaella* largely outweighed its dependence on *Fucus* (Figs. 2.12 and 2.13). For this reason, only the conditional
probability function for *Fucus* increase for a given neighbourhood configuration of *Mazzaella* was used in the model.

The nonseasonal stochastic CA model was test run 100 times for 12 iterations (equivalent to 2 years in the real world) and the model results were compared to the final sample for the control quadrat C-2 from the model testing data set. The conditional probability functions for the model were then modified slightly (usually by changing the value of the y-intercept in the regression equation) in order to produce a better fit to the C-2 field results. These modified conditional probability functions were then used in the final versions of the stochastic CA models and are shown in Tables 2.6 to 2.8.

To construct the seasonal stochastic CA model, the same deterministic rules as for the deterministic seasonal CA model were used, with respect to the behaviour of each species in the different seasons. The seasonal stochastic transition rules were thus a combination of deterministic and probabilistic rules and are shown in Tables 2.7 and 2.8.

**Zero-Neighbour Distance Cellular Automata**

In order to estimate the probability of recruitment or increase for a given species, the number of recruitment or increase events over all of the samples in the four parameterization quadrats for that species were counted and divided by the total number of possible recruitment or increase events (i.e., the DIPS output was pooled across all possible neighbourhood configurations so as to reduce the Transition Probability Matrices to scalars representing the total probability of a given state transition). The total number of possible recruitment events was estimated by the total number of empty
sites in which recruitment was able to occur (i.e., by counting every empty cell for each sampling date for each quadrat). The probability of mortality for the two species was determined similarly, by counting all the mortality events for a given species and dividing by the total number of possible mortality events. In this case, the total number of possible mortality events was simply the total number of the species in question in all of the samples of all of the quadrats. Table 2.9 shows the estimated recruitment and mortality probabilities for Fucus and the increase and mortality probabilities for Mazzaella which were used in the nonseasonal ZND CA model.

The raw output from DIPS for seasonal probabilities of increase and mortality for Mazzaella and seasonal probabilities of recruitment and mortality for Fucus were used as the stochastic seasonal transition rules for the seasonal ZND CA model (Tables 2.10 and 2.11).

**Cellular Automata Model Implementation**

The six different CA models discussed above were implemented by writing a computer program for each model in Turbo C (Borland, v4.5) (see the Appendix for the source code of the nonseasonal Zero-Neighbour Distance CA model).

**Cellular Automata Model Testing and Validation**

Because transition probabilities were estimated from each successive pair of samples taken from the four parameterization quadrats, and the interval between successive field samples was approximately two months (Kim 1995), each time step (iteration) in the CA models represented two months in the real world. Data from the four parameterization
quadrats were pooled over the two year sampling period in order to estimate the transition rules in the models (see DIPS algorithm above). Thus, any monthly dynamics present in the parameters were destroyed since the parameters were calculated as averaged transition probabilities over a two year period. As a result, predictions from the CA models were only possible every 12 iterations (two years in the real world).

In order to generate predictions from the different CA models, each model was run for 12 iterations. The deterministic CA models were only run once, since additional runs could only produce identical results. The stochastic CA models and the ZND CA models were run 100 times each and the average of the 100 runs used for testing and validation. Model predictions for each CA model were obtained for each of the 12 test quadrats by initializing the initial states of every cell in the CA models with the corresponding one cm$^2$ cell of the initial sampling date of each of the 12 test quadrats. Model predictions for a given test quadrat were then paired with that test quadrat for testing purposes. In order to mimic the various treatments performed by Kim (1995) in 9 of the 12 test quadrats, the CA models were coded so that at each iteration of the model, the appropriate treatment was also performed in the model. The model results were compared with both the last sampling date of the test quadrats (which occurred 24 months after the initial sampling) and the sampling date just prior to the last one (which occurred 22 months after the initial sampling), for each test quadrat. The reason for this was twofold: 1) data from the last sampling date for test quadrat C-4 was incomplete and therefore could not be used to test model predictions, and 2) nearly all densities and percent covers for both Fucus and Mazzaella from the last sampling date tended to be markedly lower than the
22 month sample, much more so than at any other time observed in the two year sampling history.

Comparison of the CA model results to the field data from the test quadrats was performed in several ways. First, the model predictions were plotted next to the actual field data for visual inspection of the model results. In addition, observed vs. predicted plots were made and model efficiencies, EF, (equation 2.10) (Mayer and Butler 1993) were calculated for all the CA models.

\[
EF = 1 - \frac{\sum (y_i - \hat{y}_i)^2}{\sum (y_i - \bar{y})^2}
\]  

(2.10)

where the numerator represents the sum of squares about the line of perfect fit \( (y = \hat{y}) \) and the denominator represents the corrected sum of squares. An EF of 1.0 indicates that the model predictions are identical to the observed results. An EF less than zero indicates a very poor fit between model predictions and observed results (Mayer and Butler 1993).

Models were then tested for systematic errors in prediction by calculating model predictive bias \( W \) (Power 1993) using the following formula:

\[
W = \frac{\sqrt{m \bar{e}}}{s}
\]  

(2.11)

where \( m \) is the number of quadrats in the model testing data set, \( s \) is the sample standard deviation of the data values in the model testing data set and \( \bar{e} \) is the model bias, where
and \( e_t = M_t - S_t \), where \( e_t \) is the predictive error at time \( t \), calculated as the difference between the model prediction at time \( t \) and the actual field data value at time \( t \). Because the models were being tested using a single data point from each test quadrat, the predictive bias was calculated across the 12 different quadrats rather than over 12 different times.

In order to test the predictive power of the models (by comparing with a naive zero change prediction), and to facilitate comparison between the different CA models, Theil's inequality coefficient (Power 1993) (equation (2.13)) was also calculated for the six models.

\[
U^2 = \frac{\sum_{t=1}^{m} (M_{n+t} - S_{n+t})^2}{\sum_{t=1}^{m} S_{n+t}^2}
\]  

(2.13)

where \( U^2 \) is Theil's inequality coefficient. If the model predicts perfectly, then \( U = 0 \). If \( U < 1 \) then the model predicts better than a naive zero change prediction, and if \( U > 1 \) then the model predicts more poorly than a naive zero change prediction. As with predictive bias, rather than testing over time, the CA models were tested across quadrats. Thus, a naive zero change prediction assumes that the percent cover and density of Mazzaella and Fucus, respectively, will be the same in every quadrat. The different CA models were then ranked according to their \( U \) values.
RESULTS

CA model results for *Mazzaella* 24 months after initial sampling

The percent covers of *Mazzaella* predicted by the CA models and compared with the field results, 24 months after the initial sampling date, are shown in Figures 2.14 to 2.17. All of the CA models overestimated the percent cover of *Mazzaella* in control quadrat C-3, all the *Fucus*-removal quadrats (F-2 to F-4) and all of the *Pelvetiopsis*-removal quadrats (P2 to P4) (Figs. 2.14, 2.16 and 2.17). Model predictions in these quadrats tended to differ from the field results by 10 percent cover or more. Only the Zero-Neighbour Distance models predicted greater that zero *Mazzaella* percent cover for the *Mazzaella*-removal quadrats (I-2 to I-4) (Fig. 2.15). No comparisons between model predictions and field data were possible for the C-4 control quadrat because there were no field data for the final sampling date (24 months after the initial sampling) for that quadrat; however, a great deal of variation occurred between the CA model predictions for that quadrat and sampling date (Fig 2.15). Model predictions for this quadrat ranged from zero percent cover to almost 10 percent cover.

CA model results for *Mazzaella* 22 months after initial sampling

The percent covers of *Mazzaella* predicted by the CA models and compared with the field results, 22 months after the initial sampling date, are shown in Figures 2.18 to 2.21. In general, model predictions for *Mazzaella* percent cover for the sampling date 22
months after the initial sampling were closer to the field results than for the 24 month sampling date. No systematic overestimation or underestimation was present, except for the *Pelvetiopsis*-removal quadrat, P-3, in which all the models overestimated the percent cover of *Mazzaella* (Fig. 2.21). Only the ZND and stochastic seasonal models predicted greater than zero percent cover of *Mazzaella* for the *Mazzaella*-removal quadrats (I-2 to I-4) (Fig. 2.19).

**CA model results for *Fucus* 24 months after initial sampling**

There was high variability between different CA model predictions of *Fucus* density for the 24 month sampling date, especially for control quadrat C-4 (Figs. 2.22 to 2.25). In general, the models did not seem to predict *Fucus* densities well - differences in density between the model results and field data were often greater than 10 plants per 400 cm$^2$. *Fucus* densities were consistently underestimated in all of the *Mazzaella*-removal quadrats (I-2 to I-4) and in the *Pelvetiopsis*-removal quadrat P-2 (Figs. 2.27 and 2.29). On the other hand, *Fucus* densities in the *Fucus*-removal quadrats F-2 and F-4 were consistently overestimated (Fig. 2.24). Only the nondeterministic models (i.e., the stochastic nonseasonal and seasonal, and the seasonal and nonseasonal ZND models) predicted greater than zero densities of *Fucus* for the *Fucus*-removal quadrats (F-2 to F-4) (Fig. 2.24). As with *Mazzaella* percent cover results for the same sampling date, no comparisons between *Fucus* CA model results and field data were possible for the C-4 control quadrat due to a lack of field data.
CA model results for *Fucus* 22 months after initial sampling

In contrast to the *Mazzaella* percent cover results, the *Fucus* density predictions did not seem to improve in the 22 month sampling date as compared with the 24 month sampling date results (Figs. 2.22 to 2.29). *Fucus* densities in the *Mazzaella*-removal quadrats, 1-2 and 1-4, as well as all *Fucus*-removal quadrats (F-2 to F-4) were consistently overestimated by the various CA models (Figs. 2.27 and 2.28). As with the model predictions for the 24 month sampling date, only the nondeterministic models predicted greater than zero *Fucus* densities for the *Fucus*-removal quadrats (F-2 to F-4) (Fig. 2.28).

**CA model results for *Mazzaella*: observed vs. predicted plots**

The observed vs. predicted plots of *Mazzaella* percent cover showed very clearly which CA models predominantly overestimated the percent cover of *Mazzaella*, since most of the points in the plots in which overestimation occurred fell below the line of perfect fit. Very little difference in predictive ability was observed between the deterministic, nonseasonal and the deterministic, seasonal CA models for the sampling date 24 months after the initial sampling (Fig. 2.30, A and C; EF = 0.69 and 0.67 for the nonseasonal and seasonal deterministic CA, respectively), as well as between the same two models for the 22 month sampling date (Fig. 2.30, B and D; EF = 0.86 and 0.87 for the nonseasonal and seasonal deterministic CA, respectively). For all of the deterministic model results, the predictions of *Mazzaella* percent cover in the 22 month sampling date (EF = 0.86 and 0.87) were much better than for the 24 month sampling date (EF = 0.69 and 0.67). Comparison between the 22 and 24 month sampling date stochastic model
results were similar in that the 22 month sampling date predictions (Fig. 2.31, B and D; EF = 0.92, 0.70 for the nonseasonal and seasonal stochastic CA, respectively) were much better than the 24 month predictions (Fig. 2.31, A and C; EF = 0.37, -0.19 for the nonseasonal and seasonal stochastic CA, respectively). For both sampling dates, the nonseasonal stochastic CA model predicted substantially better than the seasonal stochastic model. In fact, examination of the observed vs. predicted plots for these models revealed that the seasonal model tended to overestimate Mazzaella percent cover more than the nonseasonal model (Fig. 2.31 A to D). These observations were confirmed by the large difference in model efficiency between the two models.

Zero-Neighbourhood Distance predictions of Mazzaella percent cover for both the 22 and 24 month sampling date seemed to bear no relationship with the actual field results, since the points in the observed vs. predicted plots of these models did not cluster around the line of perfect fit at all (Fig. 2.32, A to D). The predictive abilities for this entire class of models thus seemed quite poor, with the 24 month sampling date predictions being worse than the 22 month predictions, as reflected in the model efficiencies for these models. In addition, the seasonal ZND model seemed to predict better than its nonseasonal counterpart (EF = 0.10 and 0.37 for the 24 month nonseasonal and seasonal ZND, respectively; EF = 0.63 and 0.53 for the 22 month nonseasonal and seasonal ZND, respectively).

CA model results for Fucus: observed vs. predicted plots

An examination of the observed vs. predicted plots of Fucus density revealed that all the CA models were very poor predictors of Fucus density. The plots of the model
predictions from the 24 month sampling date showed very clearly that the deterministic models consistently underestimated *Fucus* density, since all the points fell above the line of perfect fit (Fig. 2.33, A and C). This was reflected in the poor model efficiencies of these two models (EF = -0.94 and -0.74 for the nonseasonal and seasonal deterministic CA, respectively). Plots for the 22 month sampling date predictions of the same models did not show any consistent underestimation of *Fucus* density; however, the scatter of points seemed to bear no relationship to the line of perfect fit, indicating that these models also had no predictive power for the 22 month sampling date (Fig 2.33, B and D). Again, this was clearly reflected in the poor model efficiencies for these models (EF = -0.79 and -0.77 for the nonseasonal and seasonal deterministic CA, respectively).

Results for the stochastic CA models, for both the 22 and 24 month sampling dates, were also extremely poor. Examination of the observed vs. predicted plots for these models revealed very little relationship between the scatter of points and the line of perfect fit (Fig. 2.34, A to D). Again, this was reflected in the poor model efficiencies for these models (EF = -0.88 and -0.72 for the 24 month predictions of the nonseasonal and seasonal stochastic CA, respectively; EF = -0.59 and -0.36 for the 22 month predictions of the nonseasonal and seasonal stochastic CA, respectively). The ZND models were similarly poor predictors of *Fucus* density at both sampling dates (Fig. 2.35, A to D) and also had correspondingly low model efficiencies (EF = -1.38 and -1.46 for the 24 month predictions of the nonseasonal and seasonal ZND CA, respectively; EF = -0.13 and -0.14 for the 22 month predictions of the nonseasonal and seasonal ZND CA, respectively).
CA model validation results

Table 2.12 shows the $W$ statistic for predictive bias for all of the CA models. Several of the models were statistically significantly biased in their predictions, thus making them unacceptable for use on statistical grounds. The models which were found to be statistically significantly biased predictors of Mazzaella percent cover were the stochastic seasonal model (24 month predictions) and the nonseasonal ZND CA (24 month predictions). Both of these models showed large overestimation of Mazzaella percent covers in the observed vs. predicted plots (Figs. 2.31 C and 2.32 A). Models which were unacceptable predictors of Fucus included all of the CA models when predicting the density of Fucus 24 months after the initial sampling date (with the exception of the stochastic CA models), all of which showed marked underestimation of Fucus densities in the observed vs. predicted plots (Figs. 2.33 A, C and 2.35 A and C). Both stochastic models were found to be statistically significantly biased in their predictions of Fucus density for the 22 month sampling date. The remaining models did not show any statistically significant predictive bias (Table 2.12).

Although many of the CA models were poor predictors of Mazzaella percent cover and Fucus density, all of them predicted better than a naive zero change prediction (Table 2.13), including the CA model predictions of Fucus density (which tended to give very poor predictions). Since Theil's inequality coefficient can be used to rank competing models in terms of predictive power (the best models are those with the lowest $U^2$ values), these results indicate that the greatest predictive power was obtained by the stochastic, nonseasonal CA when predicting the percent cover of Mazzaella 22 months
after the initial sampling date ($U^2 = 0.159$). This model also had the highest model efficiency of all the CA models (EF = 0.92) (Fig. 2.31 B). The best predictor of *Mazzaella* percent cover 24 months after the initial sampling date was the deterministic, nonseasonal CA ($U^2 = 0.356$). In terms of predicting *Fucus* density 24 months after the initial sampling date, the best model was the stochastic, seasonal CA ($U^2 = 0.595$), one of only two CA models which were found to be statistically unbiased in its predictions of *Fucus* density for that sampling date (Table 2.12). The best predictor of *Fucus* density 22 months after the initial sampling date was the nonseasonal ZND CA ($U^2 = 0.668$), which also had the highest model efficiency for predicting *Fucus* densities for that sampling date (EF = -0.13) (Fig. 2.35 B). When considering all of the CA models together, $U^2$ values were lowest when predicting percent covers of *Mazzaella* for the 22 month sampling date, followed by predictions of *Mazzaella* percent cover for the 24 month sampling date, predictions of *Fucus* density for the 24 month sampling date, and finally, predictions of *Fucus* density for the 22 month sampling date.

**DISCUSSION**

It was found that the predictive abilities of the models depended greatly on whether the model results were compared with field data from the last sampling date (24 months after the initial sampling) or from the sampling date just prior to the last sampling date (22 months after the initial sampling). In general, for all the CA models, predictions of *Mazzaella* percent cover were much better when compared with the 22 month samples, than when compared with the 24 month samples. A similar, though not nearly as
dramatic, difference in predictive ability was also found for the CA models when predicting *Fucus* densities. An examination of the parameterization data set revealed that there was a marked difference in the percent covers of *Mazzaella* and densities of *Fucus* in the final sampling date.

There are many possible explanations to account for the poorer predictive abilities of the CA models for the 24 month sampling date percent covers of *Mazzaella* and densities of *Fucus*. First, it may be that climatic conditions at the field site caused the percent cover of *Mazzaella* to change in a way impossible for the CA models to predict, since the models only take into account abundances of *Mazzaella* and *Fucus* and the interactions between the two species. Any factors extrinsic to these would be unpredictable by the models. The experiences of other researchers in the Pacific Northwest tends to support this possibility, since many noticed unusual changes in similar intertidal systems during the same time period, perhaps due to an el Niño event (DeWreede, *pers. comm.*). In addition, it has been well documented that during the summer months, *Mazzaella* blades growing in the upper limit of its habitat suffer from bleaching, which leads to loss of tissue by the end of the summer (Scrosati and Kim, *pers. comm.*). Anything which would enhance this process, such as an unusually hot day with little cloud cover during a low tide, could cause an unseasonably large decline in the *Mazzaella* population. Such a decline would be impossible to predict simply by observing percent covers of *Mazzaella* and densities of *Fucus*, thus resulting in poor predictions by the models.

Another possible reason for the difference in predictive ability between the 24 and 22 month samples could be that other biotic factors, such as herbivory, caused the
unseasonable changes in *Mazzaella* percent cover during the fall of 1993, when the last sampling date occurred. It is known that snails of the genus *Littorina* feed preferentially on *Mazzaella* (Kim and DeWreede 1996). If these snails were more abundant than usual, they may have caused an unseasonable decline in the *Mazzaella* population; combinations of these abiotic and biotic factors could also have the same effect. Since none of these possible factors were explicitly included in the CA models, the models would be unable to predict the consequent changes in *Mazzaella* percent cover, thus leading to poorer predictive ability in the 24 month samples than in the 22 month samples.

An important point about the usefulness and applicability of such models (and other models in general) for predicting population dynamics is that if the dynamics of the population of interest are sensitive to factors which have not been explicitly included in the models, then there will be times when the model predictions do not mimic closely the dynamics in the field. In general, it is difficult in ecological systems to determine all the factors to which a population is sensitive and to include those in predictive models. Indeed, doing so often leads to enormous, intractable models which make poor predictions (Brooks and Tobias 1996, Stockle 1992). Although such extrinsic factors are included implicitly in the CA models constructed here, since these effects do impact the densities of *Fucus* and percent cover of *Mazzaella* and thus are present in the parameterization data set, they are too abstractly represented in the models to provide good predictive power when the factors behave outside their normal or average ranges. Thus, a balance must often be struck between including enough factors in the model and
keeping the model simple enough to be usable. In general, the proper balance depends on both the system being modelled and the purpose of the model. Alternatively, if the parameterization data set was not representative of normal or average conditions in the field, the resultant model parameters could result in poor predictions under normal or average conditions. Without a long term data set spanning several years (which was unavailable for this community) it would not be possible to determine whether the models were parameterized with average or non-average data.

When the CA model predictions were examined individually, it was found that a few of the models were good predictors of Mazzaella percent cover for the 22 month sampling date: the deterministic CA, both nonseasonal and seasonal, and the nonseasonal stochastic CA. No model was able to satisfactorily predict Fucus densities. This suggests that Mazzaella dynamics can be explained by the interaction between Mazzaella and Fucus alone. In fact, for the stochastic CA models, the dynamics were explicitly dependent only on Mazzaella individuals, since the conditional probability functions used to parameterize these models only considered the different neighbourhood configurations of Mazzaella. This was due to the fact that, although Fucus individuals did affect both Mazzaella and Fucus, the effect of the different neighbourhood configurations of Fucus were overpowered, in general, by the effects of the different neighbourhood configurations of Mazzaella. Thus, it may be that Mazzaella population dynamics are even simpler than suggested by the two-species CA models and may be explicitly dependent only on Mazzaella individuals. This suggests that, even though Mazzaella is highly connected to other
species in the community, and was found to interact directly with *Fucus*, *Pelvetiopsis*, and littorine snails (Kim 1995), its population dynamics may be predominantly governed by one or two simple interactions.

*Fucus*, on the other hand was very poorly predicted by the models. This suggests either that interactions between *Fucus* and *Mazzaella* have little or nothing to do with *Fucus* dynamics, or that additional species and interactions, such as with *Pelvetiopsis* and/or invertebrates, are needed to predict the dynamics. It is unlikely that *Mazzaella* does not play a role in *Fucus* dynamics, since parameterizing the CA models revealed exactly the opposite: that *Mazzaella* plays an important role in *Fucus* dynamics, often even more important that *Fucus* itself. Thus, it seems most likely that *Fucus* dynamics can only be predicted by a more complex model, one that incorporates additional interactions with other species in the community and/or additional abiotic factors.

As a group, the zero neighbour distance (ZND) CA models predicted both *Mazzaella* and *Fucus* very poorly. Since this class of model was constructed to serve as a kind of control (namely a CA model lacking local interactions), its poor predictive ability, although not a definitive test, would suggest that local interactions are indeed important between *Mazzaella* and *Fucus*.

Although many of the CA model predictions were quite poor, especially for *Fucus*, all of the models made predictions that were better than assuming no change in the populations of *Mazzaella* and *Fucus*. Since the comparisons of the model predictions to a zero change prediction were performed across space rather than
across time, these results suggest that population densities and percent covers of *Fucus* and *Mazzaella* change enough over space that even poor predictions of a population change are better than predicting no change at all.

When the CA models were compared with one another, it was found that the best predictor of *Mazzaella* percent cover for the 22 month sampling date was the stochastic, nonseasonal CA. For the 24 month sampling date, the deterministic, nonseasonal CA was the best predictor of *Mazzaella* dynamics. Given that *Mazzaella* tends to have seasonal variations in abundance (Kim 1995, Kim and Scrosati, *pers. comm.*), this result seems somewhat surprising. Although seasonal effects are observed in the field, their incorporation into the models resulted in a decline in predictive ability, rather than an increase. The reason for this may be that the way in which seasonal effects were incorporated into the models were too great a simplification of the field data to allow for good predictions, and the deterministic seasonal rules used were indeed a very abstract representation of the effects of season on *Fucus* and *Mazzaella*. Another possibility is that this is another case where the model required to simulate the system of interest is simpler than field results would initially suggest. If *Mazzaella* dynamics can be accurately predicted by a model without incorporating all of the interactions that it has with other species (as was the case with both of these models), then it does not seem unreasonable that some abiotic factors, such as seasonality, can also be safely ignored.

None of the CA models was adequate for predicting *Fucus* dynamics, either due to extremely poor modelling efficiencies, statistically significant predictive bias, or
both. The fact that the best CA models for predicting Mazzaella dynamics were poor at predicting Fucus is not that unusual. It is the exception, rather than the rule, that a given model is good at predicting everything that the modeller sets out to model (Law and Kelton 1991).

In general, the approach taken to construct and parameterize the CA models, by using the DIPS algorithm on a subset of available field data, is a viable method for the construction of simulation-style cellular automata models in ecology. Since CA are generally only used as “what-if” models, this study should serve to expand their usefulness to ecologists, as well as researchers in other sciences. To date, the only other CA used as a simulation tool was as a model for simulating the growth of NH$_4$Br crystals (Richards et al. 1990). Although Ermentrout and Keshet (1993) stated that one of the advantages of CA models was their ability to quantitatively reproduce ‘real-world’ experiments, this advantage had remained largely unexploited until the present study. Attempting to reproduce experimental results (such as clearings in the field, or competition experiments) with a CA constructed either from field data (as was done here) or from an understanding of the system to be modelled, could be a very powerful approach in determining the importance of various mechanisms for producing observed patterns and behaviours in the system of interest. As an example from this study, the ability of the nonseasonal stochastic CA model, which only explicitly included Mazzaella and Fucus, to accurately predict the 22 month percent cover of Mazzaella under the effects of a Pelvetiopsis-removal treatment indicates that Pelvetiopsis is relatively unimportant in determining
Mazzaella percent cover. This result is certainly not obvious from examining the interaction web of Kim’s community (Fig. 1.2). Another advantage to such an approach is that it forces the modeller to cast the parameters of the model in a form that can actually be measured in the field or lab (unlike many Lotka-Volterra style models, as well as others). This serves to make the model much more concrete and aids greatly in the interpretation of model predictions, by reducing model vagueness and ambiguity. Thus, a simulation approach to the use of cellular automata models in ecology provides a number of additional benefits to both the modeller and experimentalist, over the conventional “what-if” CA approach.
Table 2.1. The different cellular automata models constructed.

<table>
<thead>
<tr>
<th>Model Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic, nonseasonal</td>
</tr>
<tr>
<td>Deterministic, seasonal</td>
</tr>
<tr>
<td>Stochastic, nonseasonal</td>
</tr>
<tr>
<td>Stochastic, seasonal</td>
</tr>
<tr>
<td>Zero-Neighbour Distance, nonseasonal</td>
</tr>
<tr>
<td>Zero-Neighbour Distance, seasonal</td>
</tr>
</tbody>
</table>

Table 2.2. Deterministic, nonseasonal CA model state transition rules for *Mazzaella* increase and mortality and *Fucus* recruitment and mortality, based on the number of *Mazzaella* and *Fucus* neighbours. $p$ is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Species</th>
<th>State Transition Rule Increase / Recruitment</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Mazzaella</em></td>
<td>$p = 1.0$ if no. <em>Fucus</em> neigh. $&lt; 4$ AND no. <em>Mazzaella</em> neigh. $\geq 5$</td>
<td>$p = 1.0$ if no. <em>Fucus</em> neigh. $\geq 3$ OR no. <em>Mazzaella</em> neigh. $\leq 2$</td>
</tr>
<tr>
<td></td>
<td>0.0 otherwise</td>
<td>0.0 otherwise</td>
</tr>
<tr>
<td><em>Fucus</em></td>
<td>$p = 1.0$ if no. <em>Mazzaella</em> neigh. $\geq 4$</td>
<td>$p = 1.0$ if no. <em>Mazzaella</em> neigh. $\leq 3$</td>
</tr>
<tr>
<td></td>
<td>0.0 otherwise</td>
<td>0.0 otherwise</td>
</tr>
</tbody>
</table>
Table 2.3. Deterministic, seasonal CA model state transition rules for *Mazzaella* increase* and mortality, based on the number of *Mazzaella* and *Fucus* neighbours. *p* is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Rule Increase</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Fucus \text{ neigh.} &lt; 4 \text{ AND} \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td><em>p</em> = 0.0</td>
</tr>
<tr>
<td>Summer</td>
<td>[ p = 0.0 ]</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Fucus \text{ neigh.} \geq 3 \text{ OR} \ 0.0 \text{ otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>Fall</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Fucus \text{ neigh.} &lt; 4 \text{ AND} \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td><em>p</em> = 0.0</td>
</tr>
<tr>
<td>Winter</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Fucus \text{ neigh.} &lt; 4 \text{ AND} \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td><em>p</em> = 0.0</td>
</tr>
</tbody>
</table>

*An increase occurs when an empty cell at time *t* becomes occupied by *Mazzaella* at time *t+1*.

Table 2.4. Deterministic, seasonal CA model state transition rules for *Fucus* recruitment and mortality, based on the number of *Mazzaella* and *Fucus* neighbours. *p* is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Rule Recruitment</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \geq 4 \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \leq 3 \ 0.0 \text{ otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>Summer</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \geq 4 \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \leq 3 \ 0.0 \text{ otherwise} \end{cases} ]</td>
</tr>
<tr>
<td>Fall</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \geq 4 \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td><em>p</em> = 0.0</td>
</tr>
<tr>
<td>Winter</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \geq 4 \ 0.0 \text{ otherwise} \end{cases} ]</td>
<td>[ p = \begin{cases} 1.0 \text{ if no. } Mazzaella \text{ neigh.} \leq 3 \ 0.0 \text{ otherwise} \end{cases} ]</td>
</tr>
</tbody>
</table>
Table 2.5. The season of the interval between each pair of field sampling dates (season of transition) used for parameterizing the seasonal CA models.

<table>
<thead>
<tr>
<th>Sample Pair</th>
<th>Sampling Dates</th>
<th>Season of Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Aug 10, Oct 23</td>
<td>Fall</td>
</tr>
<tr>
<td>2</td>
<td>Oct 23, Nov 23</td>
<td>Fall</td>
</tr>
<tr>
<td>3</td>
<td>Nov 23, Feb 15</td>
<td>Winter</td>
</tr>
<tr>
<td>4</td>
<td>Feb 15, Apr 16</td>
<td>Spring</td>
</tr>
<tr>
<td>5</td>
<td>Apr 16, Jun 1</td>
<td>Spring</td>
</tr>
<tr>
<td>6</td>
<td>Jun 1, Jul 30</td>
<td>Summer</td>
</tr>
<tr>
<td>7</td>
<td>Jul 30, Oct 24</td>
<td>Summer</td>
</tr>
<tr>
<td>8</td>
<td>Oct 24, Dec 10</td>
<td>Fall</td>
</tr>
<tr>
<td>9</td>
<td>Dec 10, Feb 5</td>
<td>Winter</td>
</tr>
<tr>
<td>10</td>
<td>Feb 5, Apr 23</td>
<td>Spring</td>
</tr>
<tr>
<td>11</td>
<td>Apr 23, Jun 3</td>
<td>Spring</td>
</tr>
<tr>
<td>12</td>
<td>Jun 3, Oct 15</td>
<td>Summer</td>
</tr>
</tbody>
</table>

Table 2.6. Stochastic, nonseasonal CA model state transition rules (conditional probability functions) for Mazzaella increase and mortality and Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. $p$ is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Species</th>
<th>State Transition Rule Increase / Recruitment</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mazzaella</td>
<td>$p = 0.004 + 0.091 \times \text{(# Mazzaella neigh.)}$</td>
<td>$p = 0.37 - 0.073 \times \text{(# Mazzaella neigh.)}$</td>
</tr>
<tr>
<td>Fucus</td>
<td>$p = 0.049 + 0.00027e^{(# Mazzaella neigh.)}$</td>
<td>$p = 0.43 - 0.033 \times \text{(# Mazzaella neigh.)}$</td>
</tr>
</tbody>
</table>
**Table 2.7.** Stochastic, seasonal CA model state transition rules (conditional probability functions) for Mazzaella increase and mortality, based on the number of Mazzaella and Fucus neighbours. $p$ is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Rule Increase</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>$p = 0.004 + 0.091 , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.0$</td>
</tr>
<tr>
<td>Summer</td>
<td>$p = 0.0$</td>
<td>$p = 0.37 - 0.073 , (# \text{Mazzaella neigh.})$</td>
</tr>
<tr>
<td>Fall</td>
<td>$p = 0.004 + 0.091 , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.0$</td>
</tr>
<tr>
<td>Winter</td>
<td>$p = 0.004 + 0.091 , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.0$</td>
</tr>
</tbody>
</table>

**Table 2.8.** Stochastic, seasonal CA model state transition rules (conditional probability functions) for Fucus recruitment and mortality, based on the number of Mazzaella and Fucus neighbours. $p$ is the probability of the given state transition.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Rule Recruitment</th>
<th>State Transition Rule Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>$p = 0.049 + 0.00027e , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.43 - 0.033 , (# \text{Mazzaella neigh.})$</td>
</tr>
<tr>
<td>Summer</td>
<td>$p = 0.049 + 0.00027e , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.43 - 0.033 , (# \text{Mazzaella neigh.})$</td>
</tr>
<tr>
<td>Fall</td>
<td>$p = 0.049 + 0.00027e , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.0$</td>
</tr>
<tr>
<td>Winter</td>
<td>$p = 0.049 + 0.00027e , (# \text{Mazzaella neigh.})$</td>
<td>$p = 0.43 - 0.033 , (# \text{Mazzaella neigh.})$</td>
</tr>
</tbody>
</table>
Table 2.9. Zero Neighbour Distance, nonseasonal CA model state transition probabilities for *Mazzaella* increase and mortality and *Fucus* recruitment and mortality.

<table>
<thead>
<tr>
<th>Species</th>
<th>State Transition Probability Increase / Recruitment</th>
<th>State Transition Probability Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Mazzaella</em></td>
<td>0.185</td>
<td>0.171</td>
</tr>
<tr>
<td><em>Fucus</em></td>
<td>0.0194</td>
<td>0.250</td>
</tr>
</tbody>
</table>

Table 2.10. Zero Neighbour Distance, seasonal CA model state transition probabilities for *Mazzaella* increase and mortality.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Probability Increase</th>
<th>State Transition Probability Mortality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring</td>
<td>0.212</td>
<td>0.0619</td>
</tr>
<tr>
<td>Summer</td>
<td>0.0536</td>
<td>0.226</td>
</tr>
<tr>
<td>Fall</td>
<td>0.175</td>
<td>0.0980</td>
</tr>
<tr>
<td>Winter</td>
<td>0.312</td>
<td>0.0817</td>
</tr>
</tbody>
</table>
Table 2.11. Zero Neighbour Distance, seasonal CA model state transition probabilities for *Fucus* recruitment and mortality.

<table>
<thead>
<tr>
<th>Season</th>
<th>State Transition Probability</th>
<th>State Transition Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Recruitment</td>
<td>Mortality</td>
</tr>
<tr>
<td>Spring</td>
<td>0.0173</td>
<td>0.247</td>
</tr>
<tr>
<td>Summer</td>
<td>0.0182</td>
<td>0.324</td>
</tr>
<tr>
<td>Fall</td>
<td>0.0234</td>
<td>0.154</td>
</tr>
<tr>
<td>Winter</td>
<td>0.0182</td>
<td>0.255</td>
</tr>
</tbody>
</table>
Table 2.12. Model predictive bias (calculated using the $W$ statistic) for all CA models. (Final refers to model predictions for the sampling date 24 months after the initial sample, and Previous refers to model predictions for the sampling date 22 months after the initial sample).

<table>
<thead>
<tr>
<th>Model</th>
<th>Mazzaella Final</th>
<th>Mazzaella Previous</th>
<th>Fucus Final</th>
<th>Fucus Previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic, nonseasonal</td>
<td>1.01</td>
<td>-0.742</td>
<td>-3.44*</td>
<td>-1.57</td>
</tr>
<tr>
<td>Deterministic, seasonal</td>
<td>1.06</td>
<td>-0.660</td>
<td>-3.23*</td>
<td>-1.38</td>
</tr>
<tr>
<td>Stochastic, nonseasonal</td>
<td>1.78</td>
<td>-0.145</td>
<td>0.335</td>
<td>2.60*</td>
</tr>
<tr>
<td>Stochastic, seasonal</td>
<td>2.68*</td>
<td>1.45</td>
<td>-0.253</td>
<td>2.13*</td>
</tr>
<tr>
<td>ZND, nonseasonal</td>
<td>2.09*</td>
<td>0.569</td>
<td>-2.40*</td>
<td>-0.211</td>
</tr>
<tr>
<td>ZND, seasonal</td>
<td>1.28</td>
<td>1.26</td>
<td>-2.85*</td>
<td>-0.384</td>
</tr>
</tbody>
</table>

* statistically significant predictive bias present, $\alpha = 0.05$.

Table 2.13. Comparison of CA model predictions to a naive zero-change prediction (calculated using Theil's inequality coefficient, $U^2$) for all CA models. (Final refers to model predictions for the sampling date 24 months after the initial sample, and Previous refers to model predictions for the sampling date 22 months after the initial sample).

<table>
<thead>
<tr>
<th>Model</th>
<th>Mazzaella Final</th>
<th>Mazzaella Previous</th>
<th>Fucus Final</th>
<th>Fucus Previous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic, nonseasonal</td>
<td>0.356</td>
<td>0.217</td>
<td>0.631</td>
<td>0.840</td>
</tr>
<tr>
<td>Deterministic, seasonal</td>
<td>0.367</td>
<td>0.208</td>
<td>0.598</td>
<td>0.837</td>
</tr>
<tr>
<td>Stochastic, nonseasonal</td>
<td>0.505</td>
<td>0.159</td>
<td>0.622</td>
<td>0.794</td>
</tr>
<tr>
<td>Stochastic, seasonal</td>
<td>0.694</td>
<td>0.318</td>
<td>0.595</td>
<td>0.734</td>
</tr>
<tr>
<td>ZND, nonseasonal</td>
<td>0.604</td>
<td>0.351</td>
<td>0.699</td>
<td>0.668</td>
</tr>
<tr>
<td>ZND, seasonal</td>
<td>0.506</td>
<td>0.394</td>
<td>0.711</td>
<td>0.671</td>
</tr>
</tbody>
</table>
neighbourhood radius, \( r = 0 \);
repeat \( r = r + 1 \); current sample, \( t = 0 \);
set the elements of the State Transition Matrix (for transition \( z \)), \( T_{ij,z} = 0 \), for every \( i \), for every \( j \), for every \( z \);
set the elements of the Transition Probability Matrix (for transition \( z \)), \( P_{ij,z} = 0 \), for every \( i \), for every \( j \), for every \( z \);
repeat
\( t = t + 1 \);
\( x = \) first cell of lattice
set the elements of the Neighbourhood Configuration Matrix, \( N_{ij} = 0 \), for every \( i \), for every \( j \);
repeat
get neighbourhood (of radius \( r \)) of cell \( x \)
get state of cell \( x \) at next time step (\( x' \))
i,\( j = \) current neighbourhood configuration
\( z = \) state transition from \( x \) to \( x' \)
\( N_{ij} = N_{ij} + 1 \);
\( T_{ij,z} = T_{ij,z} + 1 \);
\( x = \) next cell
until \( x = \) last cell of lattice
calculate the transition probability matrix: for every \( i \), for every \( j \), for every \( z \) \( P_{ij,z} = T_{ij,z} / B_{ij} \);
until (the transition probabilities converge or \( t >= n-1 \))
until (the transition probabilities converge or \( r >= n^{1/d} \))

Figure 2.1. The DIPS (Direct Identification of Probabilistic Synchronous Cellular Automata) algorithm (adapted from Adamatzky 1994).
Figure 2.2. Seasonal recruitment probabilities for *Fucus gardneri*.
Figure 2.3. Seasonal mortality probabilities for *Fucus gardneri*.
Figure 2.4. Seasonal increase probabilities for *Mazzaella cornucopiae*. 
Figure 2.5. Seasonal mortality probabilities for Mazzaella cornucopiae.
Figure 2.6. Transition Probability Vector $\theta_{\text{M,Empty} \rightarrow \text{Mazzaella}}$, the probability of Mazzaella increase for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.0042 + 0.091(\# \text{Mazzaella neighbours})$, $r^2 = 0.98$. 
Figure 2.7. Transition Probability Vector $\theta_{F, \text{Empty} \rightarrow \text{Mazzaella}}$, the probability of Mazzaella increase for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is $\text{probability} = 0.098 - 0.0030(\# \text{Fucus neighbours})$, $r^2 = 0.16$. 
Figure 2.8. Transition Probability Vector $\theta_{M Mazzaella\rightarrow Empty}$, the probability of Mazzaella mortality for the different neighbourhood configurations of Mazzaella. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.61 - 0.073$ (# Mazzaella neighbours), $r^2 = 0.99$. 
Figure 2.9. Transition Probability Vector $\theta_{\text{Mazzaella} \rightarrow \text{Empty}}$, the probability of Mazzaella mortality for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = 0.12 + 0.0063(# Fucus neighbours), $r^2 = 0.62$. 
Figure 2.10. Transition Probability Vector $\theta_{M,Empty\rightarrow Fucus}$, the probability of *Fucus* recruitment for the different neighbourhood configurations of *Mazzaella*. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability = $0.049 + 0.00027e^{(\# Mazzaella neighbours)}$, $r^2 = 0.97$. 
Figure 2.11. Transition Probability Vector $\theta_{F,\text{Empty} \rightarrow \text{Fucus}}$, the probability of Fucus recruitment for the different neighbourhood configurations of Fucus. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is $\text{probability} = 0.012 + 0.0036e^{(\# \text{Fucus neighbours})}$, $r^2 = 0.88$. 
Figure 2.12. Transition Probability Vector $\theta_{M,Fucus\rightarrow Empty}$, the probability of $Fucus$ mortality for the different neighbourhood configurations of $Mazzaella$. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is $\text{probability} = 0.33 - 0.033(# Mazzaella \text{ neighbours})$, $r^2 = 0.68$. 
Figure 2.13. Transition Probability Vector $\theta_{F,Fucus\rightarrow Empty}$, the probability of *Fucus* mortality for the different neighbourhood configurations of *Fucus*. The bars represent the raw output from DIPS and the line represents the regression equation for the DIPS output. The equation of the regression line is probability $= 0.20 + 0.0025(#\ Fucus\ neighbours)$, $r^2 = 0.008$. 
Figure 2.14. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from the final sampling dates of the three control test quadrats, C-2 to C-4. The x denotes no field data available for the final sampling date for that quadrat.
Figure 2.15. Percent cover of Mazzaella predicted by the six different CA models and the field data from the final sampling dates of the three Mazzaella removal test quadrats, I-2 to I-4.
Figure 2.16. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from the final sampling dates of the three *Fucus*-removal test quadrats, F-2 to F-4.
Figure 2.17. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from the final sampling dates of the three *Pelvetiopsis* removal test quadrats, P-2 to P-4.
Figure 2.18. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three control test quadrats, C-2 to C-4.
Figure 2.19. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three *Mazzaella*-removal test quadrats, I-2 to I-4.
Figure 2.20. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three *Fucus*-removal test quadrats, F-2 to F-4.
Figure 2.21. Percent cover of *Mazzaella* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three *Pelvetiopsis*-removal test quadrats, P-2 to P-4.
Figure 2.22. Density of *Fucus* predicted by the six different CA models and the field data from the final sampling dates of the three control test quadrats, C-2 to C-4. The x denotes no field data available for the final sampling date for that quadrat.
Figure 2.23. Density of *Fucus* predicted by the six different CA models and the field data from the final sampling dates of the three *Mazzaella*-removal test quadrats, I-2 to I-4.
Figure 2.24. Density of Fucus predicted by the six different CA models and the field data from the final sampling dates of the three Fucus-removal test quadrats, F-2 to F-4.
Figure 2.25. Density of *Fucus* predicted by the six different CA models and the field data from the final sampling dates of the three *Pelvetiopsis*-removal test quadrats, P-2 to P-4.
Figure 2.26. Density of *Fucus* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three control test quadrats, C-2 to C-4.
Figure 2.27. Density of *Fucus* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three *Mazzaella*-removal test quadrats, I-2 to I-4.
Figure 2.28. Density of *Fucus* predicted by the six different CA models and the field data from samples just prior to the final sampling dates of the three *Fucus*-removal test quadrats, F-2 to F-4.
Figure 2.29. Density of *Fucus* predicted by the six different CA models and the field data from sampling dates just prior to the final sampling dates of the three *Pelvetiopsis*–removal test quadrats, P-2 to P-4.
Figure 2.30. Observed vs. Predicted plot of *Mazzaella* percent cover for the deterministic CA models. EF = modelling efficiency.
A. Deterministic, nonseasonal CA, 24 months after initial sampling, EF = 0.69.
B. Deterministic, nonseasonal CA, 22 months after initial sampling, EF = 0.86.
C. Deterministic, seasonal CA, 24 months after initial sampling, EF = 0.67.
D. Deterministic, seasonal CA, 22 months after initial sampling, EF = 0.87.
Figure 2.31. Observed vs. Predicted plot of Mazzaella percent cover for the stochastic CA models. EF = modelling efficiency.

A. Stochastic, nonseasonal CA, 24 months after initial sampling, EF = 0.37.
B. Stochastic, nonseasonal CA, 22 months after initial sampling, EF = 0.92.
C. Stochastic, seasonal CA, 24 months after initial sampling, EF = -0.19.
D. Stochastic, seasonal CA, 22 months after initial sampling, EF = 0.70.
Figure 2.32. Observed vs. Predicted plot of Mazzaella percent cover for the zero neighbour distance (ZND) CA models. EF = modelling efficiency.

A. ZND, nonseasonal CA, 24 months after initial sampling, EF = 0.10.
B. ZND, nonseasonal CA, 22 months after initial sampling, EF = 0.63.
C. ZND, seasonal CA, 24 months after initial sampling, EF = 0.37.
D. ZND, seasonal CA, 22 months after initial sampling, EF = 0.53.
Figure 2.33. Observed vs. Predicted plot of *Fucus* density for the deterministic CA models. EF = modelling efficiency.

A. Deterministic, nonseasonal CA, 24 months after initial sampling, EF = - 0.94.
B. Deterministic, nonseasonal CA, 22 months after initial sampling, EF = - 0.79.
C. Deterministic, seasonal CA, 24 months after initial sampling, EF = - 0.74.
D. Deterministic, seasonal CA, 22 months after initial sampling, EF = - 0.77.
Figure 2.34. Observed vs. Predicted plot of *Fucus* density for the stochastic CA models. EF = modelling efficiency.

A. Stochastic, nonseasonal CA, 24 months after initial sampling, EF = -0.88.
B. Stochastic, nonseasonal CA, 22 months after initial sampling, EF = -0.59.
C. Stochastic, seasonal CA, 24-months after initial sampling, EF = -0.72.
D. Stochastic, seasonal CA, 22 months after initial sampling, EF = -0.36.
Figure 2.35. Observed vs. Predicted plot of *Fucus* density for the zero neighbour distance (ZND) CA models. EF = modelling efficiency.

A. ZND, nonseasonal CA, 24 months after initial sampling, EF = -1.38.
B. ZND, nonseasonal CA, 22 months after initial sampling, EF = -0.13.
C. ZND, seasonal CA, 24 months after initial sampling, EF = -1.46.
D. ZND, seasonal CA, 22 months after initial sampling, EF = -0.14.
CHAPTER 3

BIRTH-DEATH MODEL

INTRODUCTION

Difference equation models have been used in ecology for decades as models of single and multi-species population dynamics. A classic example is the Nicholson-Bailey model used to describe host-parasitoid dynamics (Hassell and May 1989). Part of the reason for their early and widespread use in ecology is that such models are very simple, both to construct and analyze. Despite their simplicity, such models can give rise to extremely complex dynamics, including chaos (May 1974, 1976).

Difference equation models are simply the mathematical formulation of what Begon et al. (1990) called "the fundamental and unalterable ecological fact of life": the size of a population at the present moment is determined by the population size at some previous instant in time with the addition of all individuals born into the population between the previous time and now, minus the number of deaths, minus the number of emigrants, plus the number of immigrants. Difference equation models are discrete in terms of time and do not explicitly incorporate any spatial information of the system being modeled. In other words, such models assume that all members of a population respond instantly to changes in population size or other model parameters, and thus no local interactions are possible.
The purpose of this study was to construct two simple difference equation models, hereafter referred to as birth-death (BD) models, one to predict the population dynamics of *Mazzaella cormicopiae*, the other to predict the dynamics of *Fucus gardneri*, the two dominant macrophytes in Kim’s community (Kim 1995). The intention was to use these BD models as a non-spatial alternative to the CA models discussed in chapter 2, thus providing a kind of control for the importance of local spatial interactions in Kim’s community.

**METHODS**

Two birth-death (BD) models were created, one for *Mazzaella* and the other for *Fucus*. Their equations are shown below:

\[
M_{t+1} = M_t + \beta_M M_t - \delta_M M_t \\
F_{t+1} = F_t + \beta_F F_t - \delta_F F_t
\]

where \(M_{t+1}\) and \(F_{t+1}\) are the percent cover of Mazzaella and density of Fucus at time \(t+1\), \(M_t\) and \(F_t\) are the percent cover of Mazzaella and density of Fucus at time \(t\), \(\beta_M\) and \(\beta_F\) are the probabilities of recruitment for Mazzaella and Fucus, respectively and \(\delta_M\) and \(\delta_F\) are the probabilities of mortality for Mazzaella and Fucus, respectively.

\[\text{In general, it is not known for the clonal species Mazzaella whether a collection of fronds is made up of more than one genetically distinct individual, or whether a patch is formed from vegetative growth from a large holdfast (Scrosati, pers comm.). Thus, all references in this thesis to Mazzaella recruitment should be taken to mean that the plant was found in a cell which was previously unoccupied. It was not possible, given the available data, to determine whether this 'new' Mazzaella was the result of actual recruitment or vegetative growth.}\]
The parameters $\beta_M, \beta_F, \delta_M$ and $\delta_F$ were estimated from Kim's (1995) 16 permanent quadrats of field data, sampled bimonthly over a two-year period. As with the CA models described in Chapter 2, data from the 16 permanent quadrats were split into two groups: 1) a parameterization data set and 2) a model testing data set. The parameterization data set consisted of four of the 16 quadrats, three of which represented different treatments performed by Kim (1995) and a control quadrat. The three treatment quadrats from which parameterization data were obtained were Mazzaella removal, Fucus removal and Pelvetiopsis removal. The testing data set consisted of 12 quadrats, including three controls, three Mazzaella removal quadrats, three Fucus removal quadrats and three Pelvetiopsis removal quadrats.

In order to estimate the probability of recruitment for a given species, the number of recruitment events between each pair of consecutive sampling dates in the four quadrats for that species were counted and summed to give the total number of recruitment events over the entire two-year data set. This was divided by the total number of possible recruitment events to give the probability of recruitment over a two-year period. The total number of possible recruitment events were estimated by the total number of empty sites in which recruitment was able to occur (i.e., by counting every empty cell for each sampling date for each quadrat, summed over the two-year period). The probability of mortality for the two species was determined similarly, by counting all the mortality events for a given species and dividing by the total number of possible mortality events. In this case, the total number of possible mortality events was simply the total number of the species in question in all of the samples of all of the quadrats. Table 3.1 shows the
estimated recruitment and mortality parameters for Mazzaella and Fucus. As with the CA models described in Chapter 2, because the parameters were calculated using data pooled over a two-year period, the time-step in the BD model (i.e., the interval from \( t \) to \( t+1 \)) represented two years in the 'real world'.

In order to generate predictions from the BD model, \( M_r \) for equation (1) and \( F_i \) for equation (2) were taken from the percent cover of Mazzaella and the density of Fucus present in the first sampling date of a given test quadrat. Equations (1) and (2) were then solved to determine model predictions for the percent cover of Mazzaella and the density of Fucus after a two-year period. This was done 12 times, once for each test quadrat.

The 12 model percent covers of Mazzaella and densities of Fucus were then compared to the actual field data taken from the 12 test quadrats. As with the CA models in Chapter 2, the BD model results were compared with both the last sampling date (which occurred 24 months after the initial sampling), and the sampling date just prior to the last one (which occurred 22 months after the initial sampling), for each test quadrat.

Comparison of the BD model results to the field data from the test quadrats was performed in the same way as for the CA models in Chapter 2. Observed vs. predicted plots were made and model efficiencies were calculated (Mayer and Butler 1993) for both BD models (equations (1) and (2)). The two models were then tested for systematic errors in prediction by calculating model bias (Power 1993). Finally, Theil's inequality coefficient was calculated for both BD models to compare their predictions with a naive zero change prediction, and also to facilitate comparison of the BD models with the CA models.
RESULTS

Figure 3.1 shows the BD model and field data percent cover 24 months after the first sampling date for each of the 12 test quadrats. For the control quadrats C-2 and C-3 and the *Mazzaella* removal quadrats I-2, I-3 and I-4, the model predictions were very close to what was actually observed in the field. For control quadrat C-4, no field data were available for this sampling date, so comparison was not possible. For the remaining test quadrats, the model overestimated the percent cover of *Mazzaella* (Fig. 3.1). The model predictions were much closer to what was actually observed in the field, when the BD model results were compared with field data taken 22 months after the initial sampling date (Fig. 3.2).

*Fucus* density, on the other hand, was very poorly predicted by the BD model, both for the 24 month sampling date and the 22 month sampling date (Figs 3.3 and 3.4). On average, the BD model tended to underestimate *Fucus* density in both the 24 month and 22 month sampling dates. Model predictions were especially poor for the *Pelvetiopsis* removal quadrat, P-2, 24 months after initial sampling (Fig. 3.3) and the control quadrat, C-4, 22 months after initial sampling (Fig. 3.4).

Validation of the BD models is shown graphically in Figures 3.5 to 3.8. The tendency for the *Mazzaella* BD model to overestimate test quadrat percent covers of *Mazzaella*, 24 months after initial sampling can be seen quite clearly in Figure 3.5, where the majority of the points fall below the line of perfect fit. The model efficiency (EF) of this model is 0.46, further suggesting that the predictive ability of the *Mazzaella* BD model 24 months after initial sampling is quite poor. The predictive ability of the same model...
22 months after initial sampling is markedly better, having an EF of 0.92. In the observed vs. predicted plot for this model, all of the points tend to cluster closely to the line of perfect fit (Fig. 3.6). Examination of the observed vs. predicted plots of the *Fucus* BD models indicates very clearly that this model was unable to predict densities of *Fucus*, both 22 and 24 months after the initial sampling date, since the scatter of points bears no relationship with the line of best fit (Figs. 3.7 and 3.8).

The results from graphically comparing model predictions to the 12 test quadrats were reiterated in the results obtained from examining the predictive bias of each of the models (Table 3.2). No statistically significant predictive bias was present in the *Mazzaella* BD model (for both the 22 and 24 month sampling date). The *Fucus* BD model, on the other hand, was statistically significantly biased in its predictions, meaning that this model produced systematic errors in its predictions of *Fucus* densities for both the 22 and 24 month sampling date. This can be easily seen in the observed vs. predicted plots of the *Fucus* BD model, since nearly all of the points fall above the line of perfect fit, indicating that most of the model predictions underestimated the densities recorded in the field (Figs. 3.7 and 3.8).

Both the *Mazzaella* and *Fucus* BD models were found to predict better than a naive zero-change prediction (Table 3.3), although the *Mazzaella* model predicted much better than the *Fucus* model, since the $U^2$ statistics for the *Mazzaella* BD model were much closer to zero than those for the *Fucus* BD model.
DISCUSSION

Simple birth death models (BD) were constructed to describe the population dynamics of *Mazzaella* and *Fucus*. The predictions of the BD models were compared to an independent set of field data, both statistically and graphically, in order to determine how well they predicted the percent cover of *Mazzaella* and the density of *Fucus* after a two year period.

The predictive abilities of the BD models closely paralleled those of the CA models: model predictions were better for the sampling date 22 months after the initial sampling compared to 24 months after the initial sampling. Because these models were simple one-species models, any extraneous factors in the system such as weather, temperature, or other stochastic events were only incorporated implicitly in the models through the field data used in parameterization. Consequently, the effects of sudden variation in such factors on the population dynamics of *Mazzaella* and *Fucus* could not be predicted by these models. Thus the discussion in Chapter 2 pertaining to the possible causes for the differences in predictive ability of the models for the 22 month and 24 month sampling dates also applies here.

The fact that *Fucus* densities were very poorly predicted by the BD model supports the notion from chapter 2 that important interactions were lacking in this simple model, since incorporating only *Fucus* into the BD model was obviously not able to generate the observed dynamics. This was similar to the results observed for the CA models in Chapter 2. Because most of the model predictions of *Fucus* density underestimated the densities recorded in the field, it may be possible to improve the BD model predictions of
Fucus density by incorporating a 'correction factor' to increase Fucus densities in the model. Such 'correction factors' must be used with caution, however, since their inappropriate use results in models which are nothing more than curve-fitting devices (Law and Kelton 1991). Because there was no apparent biological basis for the use of a 'correction factor' in the BD model, none was employed.

Similarly to the CA models, the percent cover of Mazzaella was predicted far more accurately than Fucus densities, irrespective of whether the 22 month or 24 month sampling dates were used. This supports the suggestion that Mazzaella dynamics can be predicted well using a simple model. In this case, the presence of Fucus was not explicitly encoded in the Mazzaella BD model, and yet the model predictions were quite good, especially for the 22 month sampling date. This suggests that even Fucus is not required in order to predict Mazzaella population dynamics. This result is surprising, given the number of interactions that Mazzaella has with other organisms in this community, and indicates that some ecological systems may not require enormously complex models in order for ecologists to be able to understand and predict them. Indeed, this was found to be the case with several different models by Brooks and Tobias (1996), including a model of radiation interception by plant canopies, models of freshwater ecosystems and timber harvesting models. In addition, since the BD models are not spatial (in the sense that they do not explicitly incorporate spatial information), the ability to predict the percent cover of Mazzaella suggests that local, spatial interactions are not important for this species, in contrast to the results from chapter 2.
Thus, simple BD models can be useful for predicting the population dynamics of a single species, even when that species, such as *Mazzaella*, is enmeshed within a complex set of interactions with other members of the community. They are also useful as 'controls' for more complex models because they serve as a baseline of predictive ability from which to judge the power of more complex models.
Table 3.1. Parameters for the two Birth-Death (BD) models. $\beta$ is the probability of recruitment and $\delta$ is the probability of mortality.

<table>
<thead>
<tr>
<th></th>
<th><em>Mazzaella</em> BD Model</th>
<th><em>Fucus</em> BD Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>0.092</td>
<td>0.019</td>
</tr>
<tr>
<td>$\delta$</td>
<td>0.12</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Table 3.2. Birth-Death model predictive bias, calculated using the $W$ statistic, for 22 months after initial sampling and 24 months after initial sampling.

<table>
<thead>
<tr>
<th></th>
<th>22 month sample</th>
<th>24 month sample</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Mazzaella</em> BD Model</td>
<td>-0.23</td>
<td>1.58</td>
</tr>
<tr>
<td><em>Fucus</em> BD Model</td>
<td>-4.39*</td>
<td>-2.08*</td>
</tr>
</tbody>
</table>

* Statistically significant predictive bias present, $\alpha = 0.05$.

Table 3.3. Birth-Death model predictions as compared with a naive zero-change prediction, calculated using Theil's inequality coefficient ($U^2$), for 22 months after initial sampling and 24 months after initial sampling. A $U^2 = 0$ indicates perfect predictions, $U^2 < 1$ indicates model predictions are better than a naive zero-change prediction, $U^2 > 1$ indicates the predictive power of the model is worse than a naive zero-change prediction.

<table>
<thead>
<tr>
<th></th>
<th>22 month sample</th>
<th>24 month sample</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Mazzaella</em> BD Model</td>
<td>0.167*</td>
<td>0.467*</td>
</tr>
<tr>
<td><em>Fucus</em> BD Model</td>
<td>0.654*</td>
<td>0.813*</td>
</tr>
</tbody>
</table>

* Predictive ability of the model is better than a naive zero-change prediction.
Figure 3.1. Percent cover of *Mazzaella* predicted by the *Mazzaella* Birth-Death model and the field data from the final sampling dates of the 12 test quadrats. The $x$ denotes no field data available for the final sampling date for that quadrat. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.
Figure 3.2. Percent cover of *Mazzaella* predicted by the *Mazzaella* Birth-Death model and the field data from sampling dates just prior to the final sampling dates of the 12 test quadrats. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.
Figure 3.3. Density of *Fucus* predicted by the *Fucus* Birth-Death model and the field data from the final sampling dates of the 12 test quadrats. The x denotes no field data available for the final sampling date for that quadrat. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus removal* quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.
Figure 3.4. Density of *Fucus* predicted by the *Fucus* Birth-Death model and the field data from sampling dates just prior to the final sampling dates of the 12 test quadrats. C-2 to C-4 are control quadrats, I-2 to I-4 are *Mazzaella* removal quadrats, F-2 to F-4 are *Fucus* removal quadrats and P-2 to P-4 are *Pelvetiopsis* removal quadrats.
Figure 3.5. Observed vs. predicted plot of *Mazzaella* percent cover, 24 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = 0.46 (EF = modelling efficiency).
Figure 3.6. Observed vs. predicted plot of Mazzaella percent cover, 22 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = 0.92 (EF = modelling efficiency).
Figure 3.7. Observed vs. predicted plot of *Fucus* density, 24 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e., if the model predicts perfectly, all the points will line up along the line). EF = -2.2 (EF = modelling efficiency).
Figure 3.8. Observed vs. predicted plot of *Fucus* density, 22 months after initial sampling. The line indicates the position of the 'perfect fit' (i.e. if the model predicts perfectly, all the points will line up along the line). EF = - 0.085 (EF = modelling efficiency).
CHAPTER 4

GENERAL DISCUSSION

In total, eight different models were constructed in order to predict the population dynamics of Mazzaella cormicopiae and Fucus gardneri in Kim’s (1995) community. Six of the models were cellular automata (CA) models, and two were birth-death (BD) models. A rationale for constructing these different models of Kim’s community was that it allowed for the different models to be thought of as different hypotheses which attempt to explain the dynamics of Mazzaella and Fucus observed in the field. The use of alternative, competing models allowed for the selection of the best models among many when subjected to appropriate tests. In this case, the predictions of the different models were tested against an independent set of field data.

Choice of modelling formalism

Whenever one attempts to construct a model of a system, there is always the choice of what particular type or class of model to employ. Often, such choices are made with less thought than they should be given, since the choice of modelling formalism largely determines the potential results from the model (Hogeweg 1988). In other words model choice will affect the ease or difficulty with which the system can be modeled.

In this thesis, CA and BD models were used for a variety of reasons. Firstly, Kim’s research indicated that the survivorship of Fucus individuals was dependent on the distance between the Fucus plant and the nearest patch of Mazzaella turf (Kim 1995).
This suggested that local spatial interactions were important in this system. The explicit spatial nature of CA lends itself very well to the simulation and description of local spatial interactions and thus CA were an obvious choice for modelling interactions between *Mazzaella* and *Fucus* in Kim's community. In addition, since the purpose of the models was simulation of population dynamics, rather than simply the resolution of 'what-if' questions, it was important that the resultant models mimic, as closely as possible, the experimental design used in the field work. CA models again seemed the natural choice because of their ability to explicitly and quantitatively reproduce the results of experimental manipulations (Ermentrout and Keshet 1993). In this case, the spatial maps produced by Kim (1995) during his field work provided an ideal data set for the construction of CA models. This close match between the model and the experimental investigations in the field was also important because it would allow for a much easier (and clearer) interpretation of the model results. All individual-based models (of which CA are a subset), and most computer simulations, are capable of producing enormous amounts of results and data. This can make the interpretation of model results extremely difficult, thus it is a great advantage to keep the model as unambiguous as possible, as was done here (Caswell and Brault 1992, Law and Kelton 1991).

There are other modelling approaches which could have been used to simulate this system. One such approach, which is very similar to CA is the coupled map lattice (CML) (Hendry et al. 1996, McGlade 1995, Hassell et al. 1991). Like a CA, a coupled map lattice is usually constructed as a grid of cells where each cell can interact locally with other cells in its neighbourhood. In contrast to CA, however, the state of each cell
in a CML is a continuous variable (rather than a discrete variable) and thus varies as a continuous function of its own value and those of its neighbours. This approach was not as suitable for modelling Kim's community because the field data were coded in discrete form in 20x20 cm grids where the individual cells of the grid were assigned one of a finite number of discrete states representing the life history stage of the species occupying the grid. Although in reality, each individual *Fucus* plant, for example, grows in a continuous (rather than discrete) fashion, and thus it would seem at first glance that a CML would be a better approach to modelling this system, the continuous variable representing plant size was approximated by Kim using discrete life history classes. Thus *Fucus* was placed into three disjoint size classes (Kim 1995). For this reason, it was more appropriate to use a CA approach, rather than a CML.

Another possible approach was to employ a different individual-based formalism, one which attempted to generate the observed dynamics of *Mazzaella* and *Fucus* from even lower-level processes, such as physiology. This approach has been used very successfully in simulating the population dynamics of *Daphnia* (Murdoch et al. 1992). Unfortunately, detailed physiological data were only available for *Fucus* (Chapman 1995) and almost nothing of the physiology of *Mazzaella* was known. Thus, there simply were not enough available data to warrant this approach.

Matrix population models, which are more distantly related to CA than the previous examples, could also have been used. Such models consider population dynamics as a Markov chain (Caswell 1989). The problem with this approach is that it does not
incorporate any spatial information in the model, which seemed to be important in Kim's community.

Yet another alternative was to use a continuous, differential equation model. This modelling approach was not used because the discrete nature of the data and the seeming importance of local spatial interactions could not easily be specified in a continuous model.

One alternative to CA which was employed was a simple birth-death (BD) difference equation model, where the population density at a given time step is determined by the density at the previous time step minus the number of deaths plus the number of births in the population. Such a model had the properties of being discrete in time and explicitly nonspatial. Thus, it would be compatible with the CA models since they would all use the same discrete time intervals, while at the same time providing a test of the alternative 'hypothesis' regarding the importance of local spatial interactions in Kim's community.

A summary of model performance

The predictions of the eight different models (six CA and two BD) discussed above were tested against an independent set of field data. In addition to statistics for measuring the predictive bias of the models, Theil's inequality coefficient was calculated for each of the models in order to compare the predictions to a naive zero change prediction. The main purpose for using Theil's inequality coefficient (TIC) was to rank the different models in terms of predictive ability and thus be able to determine the model which best simulated Mazzaella and Fucus dynamics in Kim's community. This
approach has often been used to determine the best model from a group of competing models (Power 1993). Unfortunately, it was not possible to select a ‘best model’ from those constructed here, since the models predicted percent covers of *Mazzaella* and densities of *Fucus* with differing ability for the two different sampling dates (24 months and 22 months after the initial sampling); however, since all of the models were found to be extremely poor predictors of *Fucus* density, it was concluded that none of the models were adequate at predicting *Fucus* dynamics. In terms of predictions of *Mazzaella* percent cover, all of the models consistently predicted the 22 month sampling date percent cover of *Mazzaella* better than the 24 month sampling date percent cover. Thus, for the purposes of selecting the best model for predicting *Mazzaella* dynamics, only the 22 month sample data were considered.

Based on Theil’s inequality coefficient (TIC), the two best models for predicting *Mazzaella* percent cover at the 22 month sampling date were the stochastic, nonseasonal CA and the simple BD model for *Mazzaella*. Because the BD model was originally constructed to serve as a kind of ‘control’ for the importance of local spatial interactions in governing the dynamics of *Fucus* and *Mazzaella* in Kim’s community, some important questions arise from the fact that the two models were equally good at predicting *Mazzaella* percent cover. Using the principle of Occam’s razor, the best model is the simplest model which does the job, hence the *Mazzaella* BD model is the most utilitarian for the purpose of predicting *Mazzaella* percent cover.

The choice of best model is not so simple, however, since models must be constructed with a particular purpose in mind. Depending on the questions that are being
asked, either model can be considered as appropriate. For example, because the BD model is strictly deterministic and all future population predictions are based on the size of the population during the previous time step in the model, it cannot answer questions related to recovery time of Mazzaella after Mazzaella is removed from the system (such as in the Mazzaella-removal treatments performed by Kim (1995)). The reason for this is that, according to the BD model, once the percent cover of Mazzaella reaches zero, there will be no individuals left from which new ones can be born and so the population will stay at zero forever. The stochastic CA model, on the other hand, does not suffer from this restriction and would be a good model with which to answer such a question. In addition, the CA model is able to answer questions regarding spatial patterning or organization in the Mazzaella population which simply cannot be asked for the BD model, since the BD model has no way of specifying spatial information. On the other hand, if one simply wanted to know the percent cover of Mazzaella in a given patch in two years time, it is much easier to type a few numbers into a calculator (using the BD model) than to construct a computer program, debug it, generate results from a hundred runs of the model, and take the average percent cover from those runs (using the CA model). Thus the utility and appropriateness of the models depends on the questions being asked and the purpose for using the model.

The inverse problem in cellular automata and the DIPS algorithm

To date, most of the research on cellular automata and their applications in various fields have focused on what is known as the forward problem: given a rule or set of rules for a CA, predict the properties of the resultant CA (Adamatzky 1994, Gutowitz 1990a).
There is, however, another way of approaching CA: given a set of properties of a system, derive a rule or set of rules which produce these properties in a CA (Adamatzky 1994 and 1992, Gutowitz 1990a, Richards et al. 1990). This alternative approach is called the inverse problem, and is the way in which CA were treated in this thesis. Adamatzky (1994, 1992) developed algorithms for solving the inverse problem for a wide variety of systems and CA, one of which was the DIPS (Direct Identification of Probabilistic Synchronous Cellular Automata) algorithm used here.

There are several assumptions inherent in DIPS. The first is that the system from which the CA rules are being extracted is probabilistic (i.e., stochastic). This is obviously a false assumption when dealing with a completely deterministic system, yet DIPS can still be successfully applied to such a system because as the algorithm proceeds, all of the transition probabilities in the CA will converge to either zero or one in a deterministic system. Thus, whether the system of interest is deterministic or stochastic, DIPS can be used to extract the rules from the system. The other major assumption of DIPS is that the system of interest is synchronous. In this case, synchronous simply means that each transition in the system occurs in one time step and that all transitions are synchronized so that each change occurs simultaneously. This assumption is almost certainly not valid, since it is unlikely that each species examined in Kim's field quadrats responds to the various biotic and abiotic interactions in the community in the same time frame and all at the same time.

In general, this assumption of synchronous updating for modelling ecological systems is false. Interactions and processes in ecology take place on many different temporal
scales (Holling 1992 and Levin 1992). In the particular case of the models presented here, however, synchronous updating made sense because the field data used to parameterize the model came in synchronous form (each quadrat was resampled every two months, on average). Thus, in this case, the assumption of synchronous updating was upheld because the field data were collected synchronously. It is important to realize that any CA models resulting from the application of DIPS will necessarily be synchronous models. Although synchronous CA are considered, in general, to be inappropriate for ecological modeling (Hogeweg 1988), it was both necessary and appropriate for the models considered here because it allowed for each iteration in the model to correspond to the time interval between sampling dates in the field, thus mirroring Kim’s experimental design as closely as possible. Because ecologists usually collect field data in a synchronous fashion, the DIPS algorithm should be generally applicable in ecology, provided that the resultant CA models are used for the purposes of simulation. The construction of CA models using DIPS for the purpose of resolving ‘what-if’ questions would suffer from the problem that any model results would be specific to a synchronous system, thus placing rather severe (and very likely unrealistic) restrictions on the generality of the conclusions based on the model.

There is another assumption inherent in the DIPS algorithm, however, which did not hold in this study. DIPS is an iterative algorithm in that it improves the estimates of transition probabilities at each iteration by successively increasing the neighborhood radius from which it calculates transition events until the transition probabilities satisfy a Cauchy criterion (i.e., until each individual transition probability converges to a stable
value). All of the CA models presented here used a neighbourhood radius of one. An examination of the convergence of the transition probabilities for these models revealed that, on average, only 60 percent of the transition probabilities converged to stable values using a neighbourhood radius of one. This suggests that the neighbourhood radius for the CA models used here should have been larger than one to obtain better estimates for the transition probabilities. Although true in theory, there are practical problems with this conclusion. One problem arises from the fact that DIPS estimates the CA rules as the frequency of occurrence of the different states in the system, thus approximating a probability distribution (Adamatzky 1994, 1992). For this approximation to be valid, the data set used by DIPS must grow exponentially as the neighbourhood radius increases, since increasing the neighbourhood radius results in an exponential increase in the number of possible neighbourhood configurations for that neighbourhood (Richards et al. 1990). From an information theory standpoint, the problem here is that the mutual information\(^3\) (a measure of how two variables, or symbol sequences, are correlated) is being overestimated because the number of instances of the given states in Kim’s field data is relatively small in comparison to the number of possible neighbourhood configurations. Richards et al. (1990) give a good intuitive example of how this can occur when tossing a pair of dice. After one toss, the dice appear to be completely correlated. As the dice are tossed repeatedly, however, this apparent correlation rapidly disappears. Thus, increasing the neighbourhood radius in order to obtain better estimates of transition probabilities should only be attempted if there is a corresponding increase in the data

\(^3\) In the case of DIPS, mutual information measures the correlation between the state of a given cell at time \(t+1\) and its neighbourhood configuration at time \(t\).
available from which to estimate the probabilities. In the case of the CA models presented here, the data were already so sparse that it was necessary to pool the data together into a two-year block just to obtain reliable probability estimates for a neighbourhood radius of one. It is doubtful that enough data would have been available to increase the radius further.

Exponentially increasing amounts of data are generally not possible to obtain in most ecological field studies. The data collected by Kim (1996 and 1995) required two years of intensive sampling. Given that most grants last less than four years and graduate students typically spend at most two to three years collecting field data, it is improbable that enough data for constructing complex models using DIPS would ever be available for the majority of ecological systems. This is the greatest problem with the DIPS approach to simulating ecological systems.

An additional practical problem encountered when increasing the neighbourhood radius is that, as the radius is increased (and thus the number of configurations for that neighbourhood is increased exponentially), the memory requirements for the CA model increases exponentially, as well as the time complexity of the DIPS algorithm itself. Thus DIPS is a very computationally inefficient algorithm. Genetic algorithms (Richards et al. 1990), which are more efficient than DIPS, have also been developed for solving the inverse problem, and may prove to be a better approach. Although the genetic algorithms are more efficient, they are also more complicated to implement (especially as

---

4 Time complexity refers to the number of operations performed by an algorithm for a given input. In the case of the DIPS algorithm, the input is the neighbourhood radius. Thus the number of operations performed by DIPS increases exponentially with increasing neighbourhood radius.
the number of states in the system increases) and still suffer from the problem of requiring exponentially increasing amounts of data with increases in the neighbourhood radius.

All these problems having been mentioned, it is important to remember that, even though the convergence of the transition probabilities in the CA models constructed here was only 60%, it was still possible to produce a CA model which was able to predict the percent cover of *Mazzaella* very well. This leads to a very practical question: what is the relationship between parameter convergence using DIPS and the predictive ability of the resulting CA model? It may be that the theoretical requirements for convergence need not always be satisfied in order to produce CA models with good predictive abilities.

This is an important open question that has so far received no attention in the literature.

**Cellular automata as a modelling formalism in ecology**

Because of the simplicity of their construction, and the apparent lack of formal mathematical training required for their use, CA models have almost started to become 'standard fare' with ecologists. Cellular automata models, however, are not a panacea for ecologists and the construction and use of such models give rise to a myriad of problems.

The first difficulty with these models arises from their accessibility which, paradoxically, is also one of their greatest advantages. It is easy to naively construct CA models, without resorting to any mathematics whatsoever. Unfortunately, once constructed, the results of these models can be nearly impossible to interpret. This is because CA models, as already mentioned, (and individual-based models in general) are capable of producing vast amounts of data, often with no clear indication of how such
data should even be visualized, let alone analyzed (statistically or otherwise) (DeAngelis et al. 1994). Because the data produced from CA (and nearly all other simulation models) are often nonstationary (i.e., the distributions of the successive observations change over time) and autocorrelated, standard statistical treatments familiar to ecologists may be generally inapplicable (Law and Kelton 1991). So far, this has not really been a problem in the ecological application of CA models for the reason that the results of such models are almost never tested against field results to begin with (Mayer and Butler 1993)!

Another problem with the use of CA models which is almost never mentioned in the literature, is that these models are implemented by writing computer programs. It is not widely known to ecologists that the average number of errors in computer programs written by industry professionals is about 5 to 10 errors per thousand lines of code (Kusec, pers. comm.). Debugging and testing of computer programs has been a long-standing issue in computer science, and has an obvious impact on the use of computer programs in modelling. In the software design process, great pains and a highly structured approach are taken to ensure the absolute minimum number of bugs possible in the resulting code, and extensive testing of the program is performed to ensure the reliability of the software produced. Ecologists who construct CA models and are not aware of rigorous program testing and validation methods should become familiar with these.

Traditional modelling approaches, using ordinary differential equations for example, have an advantage over CA models in that their behaviour is well understood and
characterized, and can be analytically derived from the equations in the model. Even simple CA, on the other hand, are still poorly understood, and those used by most ecologists are analytically intractable (i.e., they cannot be solved analytically using currently available mathematics) (DeAngelis et al. 1994). Thus, it can be difficult, or even impossible, to understand or even identify the components of the model most critical to its behaviour (Cushing 1994). Progress is being made in this area by researchers who are beginning to accurately characterize the statistical properties of the simpler CA and are searching for the underlying parameters which give rise to those properties (Gutowitz 1990b, Li et al. 1990 and Wolfram 1984). Eventually, it is hoped, a general theory of cellular automata, and perhaps even complex systems in general, will be the result. Until such theory is available, however, we must rely on extensive simulation to give an indication of the robustness and sensitivity of the behaviour of a particular CA.

Spatial scale may also pose problems in CA models. For example, in a model which includes new seedlings and adult plants, these different life history stages would usually occupy only one cell each in the CA lattice, even though in the ‘real world’, the size difference between these two stages may be an order of magnitude or more. Attempting to overcome these problems by incorporating numerous additional states (such as additional growth stages) or creating enormous lattices to allow the sizes all life history stages to be accurately represented in the model add considerable complexity to the model and may quickly outstrip the ability of the computer to simulate the CA.

DeAngelis et al. (1994) discuss many of the other problems associated with individual-based models, of which CA are a proper subset. These include changes in the
global behaviour of the models resulting from the order in which model rules are evaluated, exceeding the limits of available computing power when simulating large populations and problems associated with the simulation of several different populations simultaneously.

A problem which is seldom discussed in the ecological CA literature is the effect of boundary conditions and synchronous updating on the behaviour of CA models. It is known that changing the updating of cells in some CA from synchronous to asynchronous can dramatically alter the ensuing behaviour of the model, and it is often the case that the more interesting structures produced by CA are an artifact of synchronous updating (Hogeweg 1988). It has been suggested (Hogeweg 1988) that for the majority of ecological applications, asynchronous CA models are much more appropriate; however, most of the CA models used in ecology continue to be synchronous. Finally, it is important, when basing conclusions on the results of CA models, to ensure that the model results are invariant with respect to the dimensions of the lattice used. This is generally not done with CA models in ecology, even though statistical techniques for doing so are available (Hendry et al. 1996, McGlade 1995).

Although there are problems with the CA formalism, many of them can be resolved with some careful thought about the purpose for the model and the reasons for selecting particular boundary conditions and lattice sizes. Thus it is unacceptable to simply employ the designs used by other researchers unless the system being modelled is very similar. It is hoped that the approach taken here, carefully justifying the reasons for choosing to use CA in the first place and for implementing the models in a particular way, will help other
ecologists in deciding how and when to use CA. If care is taken in their specification and implementation, CA models can be an extremely useful and powerful modelling approach for ecologists.
REFERENCES


neighbourhood model of an annual plant with a seedbank. Journal of Ecology 81: 
453-63.


stoloniferous N-fixing legumes in mixed pasture with grass. Journal of Ecology, 
84: 815-826.

interspecific competition for space--the effect of pattern on process. Journal of 

interception models with different levels of detail. Ecological Modelling, 60: 31- 
44.

116.

APPENDIX

The Appendix contains the source code for the nonseasonal zero neighbour distance cellular automata model. It consists of the model specification (C header file) and implementation (C source file). The code for the other models was similar.
Nonseasonal Zero Neighbour Distance Cellular Automata Model: ca8.c
Version 1.0, Sept. 15, 1996.
by Nickolas Grabovac
Botany Department, University of British Columbia

Features include: initial conditions loaded from map file,
model parameters obtained from mapr-x.c for birth and
death probabilities of Mazaella and Fucus
3 cell states: empty; Mazaella; Fucus;
simultaneous updating of cells in the Lattice

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <ctype.h>
#include <graphics.h>
#include <math.h>
#include <dos.h>

// DEFINITIONS OF CONSTANTS

// Definition of cell states in Lattice:
#define EMPTY 0
#define F 1
#define M 2
#define F1 1
#define F1r 2
#define F2 3
#define F2r 4
#define F3 5
#define F3r 6
#define F1 7
#define F1r 8
#define F2 9
#define F2r 10
#define F3 11
#define F3r 12

// Definitions for random number generator rmarin and rmar:
#define NO 78 // ASCII code for "N"
#define TRUE -1
#define FALSE 0
#define boolean int
#define ILATTICE_FILE "d:\ca\ilattice\ilatc2.map"
#define PARAM_FILE "d:\ca\params\ca6par.txt"
#define RESULTS_FILE "d:\ca\results\r8.txt"
#define SAVE_LATTICE_FILE "d:\ca\results\18.txt"

#define MAZ_BIRTH_PROB 0.0924
#define MAZ_DEATH_PROB 0.117
#define FUC_BIRTH_PROB 0.01937
#define FUC_DEATH_PROB 0.25

// GLOBAL VARIABLE DECLARATIONS

// Global Declarations for Files:
FILE *resultsFile;
FILE *saveLatticeFile;

// Global Declarations for CA Lattices:
int currLattice[52][52]; // Array which holds the current CA Lattice
pastLattice[52][52]; // Array which holds a copy of the Lattice
// of the previous time step so that
// simultaneous updating can be performed

// Global Declarations for Graphics:
const int colourIndex[16] = { BLACK, LIGHTGREEN, LIGHTRED };

// Global Declarations for Model Results:
long int hammingDist; // The hamming distance for each iteration
int density[3]; // Density of the different states after each iteration

// Global Declarations for Model Setup:
int treatmentType,
latticeType,
maxDistSize = 12;
float disturbanceRate = 0.1;

// Global Declarations for Model Initial Conditions:
int latticeR,
latticeC, // The X and Y dimensions of the lattice
runs, // The number of runs to perform
iterations = 500, // The number of iterations to perform
currRun, // The current run
currIteration; // The current iteration

// Global Declarations for Random Number Generator:
static float u[98], c, cd, cm;
static int i97, j97;
static boolean test = FALSE;
float randomNumbers[1001];
int randNumCount;

// FUNCTION DECLARATIONS

// File Functions:
void LoadLattice(void);
void LoadParams(void);
void CloseResults(int runUsed);
void SaveResults(void);
void SaveIteration(void);
void SaveIterationLattice(void);
void FGetWord(FILE *fileptr, char fileinput[]);

// Model setup and initializations Functions:
void GetInitialConditions(void);
void ConvertLattice(void);
void SetUpLattice(void);
void SetUpResults(void);

// Graphics Functions:
void SetUpGraphics(void);
void DisplayGraphics(void);
void PlotCell(int x, int y, int state);
void PlotDensity(int stateDen, int state);
void ShutOffGraphics(void);
void ShowResults(void);

// CA Calculating Functions:
void MakeClearing(void);
void DoTreatment(void);
int CalcDisturbanceRadius(int season);
unsigned long int Factorial(int n);
void DoDisturbance(int season);
void RemoveMaz(void);
void RemoveFuc(void);
void RemovePel(void);
int Odd(int num);
void CalcDensities(void);
void CopyLattice(void);
void DoOneIteration(void);
void CalculateCell(int c, int r);
void GetNeighbours(int c, int r, int numNei[]);
void MazRules(int c, int r);
void FucRules(int c, int r, int LHstage);
void PelRules(int c, int r, int LHstage);
void FucGrowthRules(int c, int r, int LHstage);
void PelGrowthRules(int c, int r, int LHstage);
void FucReprodRules(int c, int r, int LHstage);
void PelReprodRules(int c, int r, int LHstage);
void BirthRules(int c, int r);

// Random Number Generation Functions:
static void rmarin(int ij, int kl);
void ranmar(float rvec[], int len);
float RandNum(void);
void GenerateRandNums(void);

// Debugging Functions:
void TestLoadLattice(void);
void TestLoadParams(void);
Nonseasonal Zero Neighbour Distance Cellular Automata Model: ca8.c
Version 1.0, Sept. 15, 1996.
by Nickolas Grabovac
Botany Department, University of British Columbia

Features include:
- initial conditions loaded from map file,
- model parameters obtained from mapr-x.c for birth and
depth probabilities of Mazaella and Fucus
- 3 cell states: empty; Mazaella; Fucus;
- simultaneous updating of cells in the Lattice

// INCLUDE FILES
#include "ca8.h"

// FUNCTION IMPLEMENTATIONS

// File Functions:

void LoadLattice() {
  // Loads the next CA lattice pair from a file called ilatca2.map.
  // This file contains a single 20x20 array of integers (ranging from 0 to
  // 13) which represents the first permanent quadrat map for a given
  // treatment). The numbers in the file are separated by TABS and each row
  // is separated by a LF.
  FILE *latticeFile; // the file pointer to the ilattice.map file
  int r; // the current row of the 20x20 array in the file
  int c; // the current column of the 20x20 array in the file
  char fileinput[13] = ""; // a string used to hold the characters read
  // from the ilattice.map file.

  if ((latticeFile = fopen(ILATTICE_FILE, "r")) == NULL) {
    printf("Can't open ilatt.map file!");
    exit(0);
  }
  for (r = 1; r <= latticeC; r++) {
    for (c = 1; c <= latticeR; c++) {
      FGetWord(latticeFile, fileinput);
      currLattice[r][c] = atoi(fileinput);
    }
  }
  fclose(latticeFile);
}

void LoadParams() {
  // Loads the CA parameters from a file called ca2par.txt. This file contains
  // 36 8x8 float arrays where each number is separated by a TAB and rows are
  // separated by a LF. Note, individual 8x8 arrays are NOT separated from one
  // another by an extra LF (i.e. the 8x8 parameter blocks not separated
  // from one another by a blank line). Also, each float number is 12 char's
  // long, thus str must be dimmed as 13. The order of parameters is:
  // birth params, death params, reprod params, growth params. The order
  // within each of these groupings corresponds to the order of states given
  // in the global state definitions.
  FILE *paramFile; // pointer to the parameter file
  char str[13]; // a string used to hold data read from the file
  ...
int numNeil, numNei2; // the number of neighbours of a given type denoted as 1 and 2) around the current state.

int state; // the current state

if ((paramFile = fopen(PARAM_FILE, "r")) == NULL) {
  printf("\n\nCan't open parameter file ca6par.txt!");
  exit(0);
}
/* for (state = F; state <= M; state++) {
  for (numNeil = 0; numNeil <= 8; numNeil++) {
    for (numNei2 = 0; numNei2 <= 8; numNei2++) {
      FGetWord(paramFile, str);
      birthProb[state][numNeil][numNei2] = atof(str);
    }
  }
}
for (state = F; state <= M; state++) {
  for (numNeil = 0; numNeil <= 8; numNeil++) {
    for (numNei2 = 0; numNei2 <= 8; numNei2++) {
      FGetWord(paramFile, str);
      deathProb[state][numNeil][numNei2] = atof(str);
    }
  }
  fclose(paramFile);
} // end LoadParams

void FGetWord(FILE *fileptr, char fileinput[]) {
  // Retrieves a single word from the file pointed to be *fileptr and places it in fileinput[].
  //
  const int TAB = 9; // the ASCII value for TAB
  const int LF = 10; // the ASCII value for LF
  const int SP = 32; // the ASCII value for Space
  int index; // the number of characters read from the file
  int anykey; // used for error checking
  int endloop = FALSE; // if == TRUE, then end the while loop
  char blankStr[13] = ""; // a blank string used to clear fileinput

  index = 0;
  strcpy(fileinput, blankStr);
  fileinput[index] = getc(fileptr);
  while (endloop != TRUE) {
    if (fileinput[index] == EOF) {
      // This is error-checking code
      // printf("\n\nEnd of File reached. (FGetWord).") ;
      // printf("\nindex = %d", index);
      // anykey = getch();
      endloop = TRUE;
    } else if ( (fileinput[index] == TAB) || (fileinput[index] == SP) ) {
      if (index == 0) {
        fileinput[index] = getc(fileptr);
      } else {
        endloop = TRUE;
      }
    } else if (fileinput[index] == LF) {
      if (index == 0) {
        strcpy(fileinput, blankStr);
        fileinput[index] = getc(fileptr);
      } else {
        endloop = TRUE;
      }
    } else {

index++;  
fileinput[index] = getc(fileptr);
}
} // FGetWord

void CloseResults(int runUsed) {
  // Closes all the results files
  fprintf(saveLatticeFile, "\n\nRun Used: %d", runUsed);
  fclose(resultsFile);
  fclose(saveLatticeFile);
} // CloseResults

void SetUpResults() {
  // Opens the results file and sets it up for recording data
  if ((resultsFile = fopen(RESULTS_FILE, "w")) == NULL) {
    printf("\nCan't open results file resca6.txt!");
    exit(0);
  }
  fprintf(resultsFile, "Run\tIteration\tMazDen\tFucDen\n");
  if ((saveLatticeFile = fopen(SAVE_LATTICE_FILE, "w")) == NULL) {
    printf("\nCan't open results file latt7d.txt!");
    exit(0);
  }
  fprintf(saveLatticeFile, "Lattice maps from CA7d: Deterministic CA
with 3 states.\n");
} // SetUpResults

void SaveResults() {
  // Saves the results of the current run to a file
  int r, c;
  fprintf(resultsFile, "%d\t%d\t%d\n", currRun, density[M], density[F]);
  fprintf(saveLatticeFile, "\n\n");
  for (r = 1; r <= latticeR; r++) {
    for (c = 1; c <= latticeC; c++) {
      fprintf(saveLatticeFile, "%d\t", currLattice[r][c]);
    }
  }
  fprintf(saveLatticeFile, "\n");
} // SaveResults

void SaveIteration() {
  // Saves the results of the current iteration to a file
  int r, c;
  fprintf(resultsFile, "%d\t%d\t%d\t%d\n", currRun, currIteration,
density[M], density[F]);
  fprintf(saveLatticeFile, "\n\n");
  for (r = 1; r <= latticeR; r++) {
    for (c = 1; c <= latticeC; c++) {
      fprintf(saveLatticeFile, "%d\t", currLattice[r][c]);
    }
  }
  fprintf(saveLatticeFile, "\n");
} // SaveIteration
void SaveIterationLattice() {
    // Saves the results of the current iteration to a file
    int r, c;
    fprintf(saveLatticeFile, "\n\n\n");
    for (r = 1; r <= latticeR; r++) {
        fprintf(saveLatticeFile, "%d\t", currLattice[r][c]);
    }
    fprintf(saveLatticeFile, "\n");
}

void GetInitialConditions() {
    // Gets input from the user to specify the initial conditions of the
    // model such as, latticeC, latticeR, runs, and iterations.
    char userInput[6], yesno;
    clrscr();
    printf("Set CA Rules:\n\n");
    printf("\nSpecify Lattice Type (0 = random, 1 = load from file): ");
    gets(userInput);
    latticeType = atoi(userInput);
    if (latticeType == 0) {
        latticeC = 50;
        latticeR = 50;
    } else {
        latticeC = 20;
        latticeR = 20;
    }
    printf("\n\nSpecify Treatment (\n	0 = none
	l = 20x20 clearing
	2 = remove all Mazaella\n	3 = remove all Fucus\n	4 = remove all Pelvetiopsis: ");
    gets(userInput);
    treatmentType = atoi(userInput);
    /* printf("\n\nSpecify Lattice Dimensions:\n\n"); printf("\tX-dimension: ");
        gets(userInput);
        latticeC = atoi(userInput); */
    printf("\n\nSpecify Lattice Dimensions:\n\n");
    printf("\tX-dimension: ");
    gets(userInput);
    latticeC = atoi(userInput);
    printf("\n\nSpecify Lattice Dimensions:\n\n");
    printf("\tY-dimension: ");
    gets(userInput);
    latticeR = atoi(userInput);
    printf("\n\nNumber of Model Runs: ");
    gets(userInput);
    runs = atoi(userInput);
    printf("\n\nDo you wish to run the model (y/n): ");
    yesno = getche();
    yesno = toupper(yesno);
    if (yesno == NO) {
        printf("\nyou said NO!");
        exit(0);
    }
void ConvertLattice() {
    //
    // Converts the currLattice from containing all 14 possible states to
    // containing 4 states. All Fucus LH stages are given the value F1 and
    // all Pelvetiopsis LH stages are given the value Pl.
    //
    int r; // the current row of the 20x20 array in the file
    int c; // the current column of the 20x20 array in the file
    for (r = 1; r <= latticeC; r++) {
        for (c = 1; c <= latticeR; c++) {
            switch (currLattice[r][c]) {
                case FI: currLattice[r][c] = F; break;
                case F2: currLattice[r][c] = F; break;
                case F3: currLattice[r][c] = F; break;
                case P1: currLattice[r][c] = EMPTY; break;
                case P2: currLattice[r][c] = EMPTY; break;
                case P3: currLattice[r][c] = EMPTY; break;
                default: break;
            }
        }
    }
}
} // ConvertLattice

void SetUpLattice() {
    //
    // Creates a Lattice of dimensions (latticeC, latticeR)
    //
    int rnum,
    int r, c,
    for (r = 0; r <= latticeC+1; r++) {
        for (c = 0; c <= latticeR+1; c++) {
            currLattice[r][c] = 0;
        }
    }
    if (latticeType == 0) {
        for (r = 1; r <= latticeC; r++) {
            for (c = 1; c <= latticeR; c++) {
                rnum = (int)3.0*RandNum();
                switch (rnum) {
                    case 0: currLattice[r][c] = EMPTY; break;
                    case 1: // A LH-stage of Fucus
                        currLattice[r][c] = F; break;
                    case 2: currLattice[r][c] = M; break;
                    default: break;
                }
            }
        }
    }
} // SetUpLattice
default: break;
}
}
} else {
  LoadLattice();
  ConvertLattice();
}
} //SetUpLattice

// Graphics Functions:

//
void PlotCell(int r, int c, int state) {
  // Graphically displays the state of a cell.
  // c is the current column of the CA Lattice
  // r is the current row of the CA Lattice
  int left, top, right, bottom; // the coordinates of the square which
  // defines the CA lattice cell
  const int caWindowTop = 15; // (caWindowLeft, caWindowTop) are the
  const int caWindowLeft = 15; // coordinates for positioning the
top-LH
  corner of the CA lattice on the graphics
  // output.
  const int cellSize = 10; // the size in pixels of the diameter of a
  // CA lattice single
  cell.
  left = cellSize * (r - 1) + caWindowLeft;
  top = cellSize * (c - 1) + caWindowTop;
  right = left + cellSize - 1;
  bottom = top + cellSize - 1;
  setcolor(colourIndex[state]);
  setfillstyle(SOLID_FILL, colourIndex[state]);
  //rectangle(left, top, right, bottom);
  bar(left, top, right, bottom);
} // PlotCell

void PlotDensity(int stateDen, int state) {
  // Graphically displays the density of a species.
  // (note: the x-axis of the graph is 600 pels long and the y-axis is
  // 200 pels high.
  int modifiedDen; // the density which is plotted on the y-axis
  int maxDensity; // the maximum density a state can attain in the CA
  float denDivisor; // a scaling parameter for plotting the density on
  // the y-axis.
  const float yAxisHeight = 200.0;
  const float xAxisLength = 600.0;
  int x, y; // the (x, y) coordinates of the pixel to plot
  maxDensity = latticeC * latticeR;
  denDivisor = (float) maxDensity / yAxisHeight;
  modifiedDen = (int) ( (float) stateDen / denDivisor);
  if (iterations > 600) {
    x = (currIteration / 2) + 14;
  } else {
    x = currIteration + 14;
void SetUpGraphics() {
    // Sets up the graphical display and draws the background screen.
    int gd = VGA;  // VGA display mode
    int gm = VGA2;  // VGA graphical mode
    int errorcode;
    // Register the display driver
    errorcode = registerfarbgidriver(EGAVGA_driver_far);
    if (errorcode < 0) {
        printf("nGraphics error! Problem registering driver!");
        exit(1);
    }
    initgraph(&gd, &gm, "d:\tc\bgi");
    DisplayGraphics();
} // SetUpGraphics

void DisplayGraphics() {
    // Draws the background screen.
    clearviewport();
    setcolor(LIGHTGRAY);
    rectangle(14, 14, 215, 215);
    line(14, 250, 14, 450);
    line(15, 450, 615, 450);
    settextstyle(SMALL_FONT, HORIZ_DIR, 4);
    outtextxy(14, 1, "CA MODELLING KIT: CA6 v1.0, copyright 1996, Nickolas Grabovac");
    outtextxy(295, 455, "ITERATION");
    settextstyle(SMALL_FONT, VERT_DIR, 4);
    outtextxy(1, 320, "DENSITY");
    gotoxy(31, 9);
    printf("Legend:");
    gotoxy(34, 10);
    printf("Fucus");
    setcolor(colourIndex[F]);
    rectangle(245, 145, 255, 155);
    setcolor(colourIndex[Pall]);
    rectangle(245, 161, 255, 171);
    gotoxy(34, 12);
    printf("Mazaella");
    setcolor(colourIndex[M]);
    rectangle(245, 176, 255, 186);
} // DisplayGraphics

void ShutOffGraphics() {
    // Closes the graphical display.
    closegraph();
} // ShutOffGraphics

void ShowResults() {
    // Displays the results graphically on the screen
gotoxy(31,2);
printf("Run: %d", currRun);
gotoxy(31,3);
printf("Iteration: %d", currIteration);
gotoxy(31,5);
printf(" ");
gotoxy(31,5);
printf("Total Fucus Density: %d", density[F]);
gotoxy(31,6);
printf(" ");
gotoxy(31,6);
printf("Mazaella Density: %d", density[M]);
//gotoxy(31,7);
//printf(" ");
//gotoxy(31,7);
//printf("Total Pelvetiopsis Density: %d", density[Pall]);
PlotDensity(density[F], F);
PlotDensity(density[M], M);
//PlotDensity(density[P], Pall);

} // ShowResults

// CA Calculating Functions:

void DoTreatment() {

// Performs a treatment on the lattice according to the treatmentType:
// 0 = none; 1 = 20x20 clearing; 2 = remove all Mazaella;
// 3 = remove all Fucus; 4 = remove all Pelvetiopsis;

switch (treatmentType) {
    case 0: break;
    case 1:
        if (currIteration == 0) {
            MakeClearing();
        }
        break;
    case 2: RemoveMaz(); break;
    case 3: RemoveFuc(); break;
    default: break;
}
} // Do Treatment

void MakeClearing() {

// Creates a 20x20 clearing in the CA lattice, located randomly within
// the lattice.

const int clearingSizeX = 20; // the size of the clearing on the x-
axis
const int clearingSizeY = 20; // the size of the clearing on the y-
axis
int clearX; // the x-coord of the clearing (from 1 to
clearingSize)
int clearY; // the y-coord of the clearing (from 1 to clearingSize)
int r, // (r, c) is the top left hand corner of the
    // clearing
c;
int minR, // the smallest r to locate the clearing
    maxR, // the largest r to locate the clearing
    minC, // the smallest c to locate the clearing
    maxC; // the largest c to locate the clearing
float rnum; // a random number from 0 to 1
minR = 1;
minC = 1;
maxR = latticeC - clearingSizeX - 1;
maxC = latticeR - clearingSizeY - 1;
rnum = RandNum();
r = minR + (int) (maxR * rnum);
rnum = RandNum();
c = minC + (int) (maxC * rnum);
for (clearX = 0; clearX < clearingSizeX; clearX++) {
    for (clearY = 0; clearY < clearingSizeY; clearY++) {
        currLattice[r+clearX][c+clearY] = EMPTY;
    }
}
} // end MakeClearing

int CalcDisturbanceRadius (int season) {
    // Calculates the size of the disturbance (radius) based on a Poisson
    // distribution, where x is the size of the disturbance (distSize) and
    // lambda is the disturbance rate or frequency (disturbanceRate).
    // The function returns the disturbance radius (distRadius).
    //
    const float e = 2.718281828;
    int distSize; // the possible sizes of the disturbance (radius)
    int distRadius; // the actual disturbance size (radius)
    unsigned long int fact; // the factorial of distSize
    float probDistSize[20]; // the probability a disturbance of a given
    // size will occur.
    float cumProbDistSize[20]; // the cumulative probability a disturbance
    // of a given size will occur.
    float rnum; // a random number between 0.0 and 1.0;
    for (distSize = 1; distSize <= maxDistSize; distSize++) {
        fact = Factorial(distSize);
        probDistSize[distSize] = pow(e, -1*disturbanceRate)*
        pow(disturbanceRate, distSize) / fact;
    }
    for (distSize = 1; distSize <= maxDistSize; distSize++) {
        cumProbDistSize[distSize] = probDistSize[distSize] +
        probDistSize[distSize-1];
    }
    cumProbDistSize[0] = probDistSize[0];
    cumProbDistSize[maxDistSize] = 1.0;
    rnum = RandNum();
    if (rnum > cumProbDistSize[0]) {
        for (distSize = 1; distSize <= maxDistSize; distSize++) {
            if ( (rnum > cumProbDistSize[distSize-1]) &&
                 (rnum <= cumProbDistSize[distSize]) ) {
                distRadius = distSize;
                distSize = maxDistSize;
            }
        }
    } else {
        distRadius = 0;
    }
    return distRadius;
} // CalcDisturbanceRadius

void DoDisturbance(int season) {
// Creates a disturbance in the lattice by setting a randomly located
// circular area of radius distRadius to EMPTY.
// season is the season of the current iteration of the model.

int distRadius;    // the size (radius) of the disturbance
int x;             // the horizontal aspect of the disturbance circle
int y;             // the vertical aspect of the disturbance circle
int centreY;       // the y-coord of the centre of the disturbance
int centreX;       // the x-coord of the centre of the disturbance
float rnum;        // a random number between 0 and 1.

distRadius = CalcDisturbanceRadius(season);
rnum = RandNum();
centreY = (int) (latticeR * rnum);
rnum = RandNum();
centreX = (int) (latticeC * rnum);
for (y = centreY - distRadius; y <= centreY + distRadius; y++) {
    for (x = centreX - distRadius + abs(centreY - y);
        x <= centreX + distRadius - abs(centreY - y); x++) {
        if ((x <= latticeC) && (x >= 1) && (y <= latticeR) &&
            (y >= 1)) {
            currLattice[x][y] = EMPTY;
        }
    }
}
} // DoDisturbance

unsigned long int Factorial(int n) {
    //
    // Returns the factorial of an integer (n).
    // The maximum size of n is 12, otherwise it will overflow!
    //
    int counter;    // a counter that goes from 1 to n.
    unsigned long int factorial;    // the factorial of n

    factorial = 1;
    for (counter = 1; counter <= n; counter++) {
        factorial = factorial*counter;
    }
    return factorial;
} // Factorial

void RemoveMaz() {
    //
    // Changes the state of all Mazaella cells to EMPTY
    //
    int r;    // the current row of the Lattice array in the file
    int c;    // the current column of the Lattice array in the file

    for (r = 1; r <= latticeC; r++) {
        for (c = 1; c <= latticeR; c++) {
            if (currLattice[r][c] == M) {
                currLattice[r][c] = EMPTY;
            }
        }
    }
} // RemoveMaz

void RemoveFuc() {
    //
    // Changes the state of all Fuc cells to EMPTY
    //
int r; // the current row of the Lattice array in the file
int c; // the current column of the Lattice array in the file

for (r = 1; r <= latticeC; r++) {
    for (c = 1; c <= latticeR; c++) {
        if (currLattice[r][c] == F) {
            currLattice[r][c] = EMPTY;
        }
    }
}

int Odd(int num) {
    // Returns 1 (TRUE) if num is odd, returns 0 (FALSE) if num is even.
    //
    div_t divideNum; // holds the quotient and remainder of the division
    int divResult; // the result of the division, odd or even
    const int two = 2; // divide num by this to see if it is odd
    const int odd = 1;
    const int even = 0;

    divideNum = div(num, two);
    if (divideNum.rem == 0) {
        divResult = even;
    } else {
        divResult = odd;
    }
    return divResult;
}

void CalcDensities() {
    // Calculates the densities of Mazaella, Fucus and Pelvetiopsis,
    // as well as the density of bare rock, after each iteration of
    // the model.
    //
    int r, c;
    int state; // a counter for initialization for..next loop

    for (state = EMPTY; state <= M; state++) {
        density[state] = 0;
    }

    for (r = 1; r <= latticeC; r++) {
        for (c = 1; c <= latticeR; c++) {
            switch (currLattice[r][c]) {
                case EMPTY:  density[EMPTY]++;  
                            PlotCell(r, c, EMPTY);
                            break;
                case F:      density[F]++;       
                            PlotCell(r, c, F);
                            break;
                case M:      density[M]++;       
                            PlotCell(r, c, M);
                            break;
                default:    break;
            }
        }
    }
}

void CopyLattice() {

// Copies the currLattice into another array: pastLattice to enable
// simultaneous updates in the model.
//
int r, c;
for (r = 1; r <= latticeC; r++) {
    for (c = 1; c <= latticeR; c++) {
        pastLattice[r][c] = currLattice[r][c];
    }
}
} // CopyLattice

void DoOneIteration() {
    // Does one iteration of the model

int r, c;
    hammingDist = 0;
    for (r = 1; r <= latticeC; r++) {
        for (c = 1; c <= latticeR; c++) {
            CalculateCell(r, c);
        }
    }
} // DoOneIteration

void CalculateCell(int r, int c) {
    // Calculates simultaneously the state of every cell in the
    // Lattice based its current state and the state of the cells around it.

switch (pastLattice[r][c]) {
    case EMPTY: BirthRules(r, c); break;
    case F: FucRules(F, r, c); break;
    case M: MazRules(r, c); break;
    default: break;
}
} // CalculateCell

void BirthRules(int r, int c) {
    // Determines whether the EMPTY cell will change state and if so, what
    // state it will change into.
    // r is the current row of the Lattice, c is the current column.

int numNei[3]; //the number of neighbours of each state type
    float rnum; // a random number from 0.0 to 1.0
    int Mbirth; // flags to determine whether or not M or F can be
    int Fbirth;
    Mbirth = FALSE;
    Fbirth = FALSE;
    rnum = RandNum();
    if ( rnum <= MAZ_BIRTH_PROB ) {
        Mbirth = TRUE;
    }
    rnum = RandNum();
    if ( rnum <= FUC_BIRTH_PROB ) {
        Fbirth = TRUE;
    }
if ( (Mbirth == TRUE) && (Fbirth == TRUE) ) {
    rnum = RandNum();
    if (rnum > 4.0) {
        currLattice[r][c] = M;
    } else {
        currLattice[r][c] = F;
    }
} else if ( Mbirth == TRUE ) {
    currLattice[r][c] = M;
} else if ( Fbirth == TRUE ) {
    currLattice[r][c] = F;
} else {
}
} // BirthRules

void FucRules(int LHstage, int r, int c) {
    // Determines whether the Fucus cell will change state and if so, what
    // state it will change into. (i.e. Growth, Reproduction, or Death)
    // LHstage is the life history stage of Fucus in the cell (r,c).
    // r is the current row of the Lattice, c is the current column.
    int numNei[3]; //the number of neighbours of each state type
    float rnum; // a random number from 0.0 to 1.0
    rnum = RandNum();
    if ( rnum <= FUC_DEATH_PROB ) {
        currLattice[r][c] = EMPTY;
        hammingDist++;
    }
} // FucRules

void MazRules(int r, int c) {
    // Determines whether the current cell with the state of M will stay as M
    // or become EMPTY, based on the states of the surrounding cells.
    // r is the current row of the Lattice, c is the current column.
    int numNei[3]; //the number of neighbours of each state type
    float rnum; // a random number from 0.0 to 1.0
    rnum = RandNum();
    if ( rnum <= MAZ_DEATH_PROB ) {
        currLattice[r][c] = EMPTY;
        hammingDist++;
    }
} // MazRules

void GetNeighbours(int r, int c, int numNei[]) {
    // Determines the number of neighbours of each state in the von Neumann
    // neighbourhood around the cell with position r (row #), c (column #).
    int nr; // the row of the von Neuman neighbourhood
    int nc; // the column of the von Neuman neighbourhood
    int i; // initialization for..next counter
    for (i = EMPTY; i <= M; i++) { // initialize numNei to 0
        numNei[i] = 0;
    }
    for (nr = -1; nr <= 1; nr++) {
        for (nc = -1; nc <= 1; nc++) {

if ( (nr == 0) && (nc == 0) ) {
    switch (pastLattice[r+nr][c+nc]) {
        case F:
            numNei[F]++;
            break;
        case M:
            numNei[M]++;
            break;
        default:
            break;
    }
} // GetNeighbours

//
// Random Number Generating Functions:
//
void GenerateRandNums() { //
// Creates an array of 1000 random numbers
//
    const int len = 1000; // the number of random numbers to generate
    struct doctime_t t; // the current system time
    float seed1, seed2; // the seeds for the random number generator
    dos_gettime(&t);
    seed1 = (float) (t.hsecond + t.second + t.hour);
    seed2 = (float) (t.hsecond + t.second + t.minute);
    rmarin(seed1, seed2); // initialize random number generator
    ranmar(randomNumbers, len); // generate a random number
} // GenerateRandNums

float RandNum() {
//
// Returns a random number between 0.0 and 1.0
//
    const int MAXRANDNUMS = 1000;
    randNumCount++;
    if (randNumCount >= MAXRANDNUMS) {
        GenerateRandNums();
        randNumCount = 1;
    }
    return randomNumbers[randNumCount];
} // RandNum

static void rmarin(int ij, int kl) { //
/*
 C This is the initialization routine for the random number generator RANMAR()
 C NOTE: The seed variables can have values between:  0 <= IJ <= 31328
 C     0 <= KL <= 30081
 C The random number sequences created by these two seeds are of sufficient
 C length to complete an entire calculation with. For example, if several
 C different groups are working on different parts of the same calculation,
 C each group could be assigned its own IJ seed. This would leave each group
 C with 30000 choices for the second seed. That is to say, this random
 C number generator can create 900 million different subsequences -- with
 C each subsequence having a length of approximately 10^30.
 C
 C Use IJ = 1802 & KL = 9373 to test the random number generator. The
 C subroutine RANMAR should be used to generate 20000 random numbers.
 C Then display the next six random numbers generated multiplied by 4096*4096
 C If the random number generator is working properly, the random numbers


C should be:

```
137
C

653892.0
14220222.0
7275067.0

6172232.0
8354498.0
10633180.0

*/
```

```
int i, j, k, l, ii, jj, m;
float s, t;

if (ij<0 || ij>31328 || kl<0 || kl>30081) {
  puts("The first random number seed must have a value between 0 and 31328." );
  puts("The second seed must have a value between 0 and 30081.");
  exit(l);
}

i = (ij/177)%177 + 2;
j = ij%177 + 2;
k = (kl/169)%178 + 1;
l = kl%169;

for (ii=l; ii<=97; ii++) {
  s = 0.0;
  t = 0.5;
  for (jj=l; jj<=24; jj++) {
    m = (((i*j)%179)*k) % 179;
    i = j;
    j = k;
    k = m;
    l = (53*l + 1) % 169;
    if ((l*m)%64 >= 32) s += t;
    t *= 0.5;
  }
  u[ii] = s;
}

c = 362436.0 / 16777216.0;
cd = 7654321.0 / 16777216.0;
cm = 16777213.0 / 16777216.0;
i97 = 97;
j97 = 33;

void ranmar(float rvec[], int len) {
  /* len random numbers are placed in rvec[1..len] */
  /*
  ** This is the random number generator proposed by George Marsaglia in
  ** Florida State University Report: FSU-SCRI-87-50
  ** It was slightly modified by F. James to produce an array of pseudorandom
  ** C numbers.
  */

  int ivec;
  float uni;

  if (test==FALSE) {
    puts("Call the init routine rmarin() before calling ranmar().");
    exit(2);
  }

  for (ivec=1; ivec<=len; ivec++) {
    uni = u[i97] - u[j97];
    if (uni < 0.0) uni += 1.0;
    u[i97] = uni;
    i97--;
    if (i97==0) i97 = 97;
    j97--;
    if (j97==0) j97 = 97;
```
void main(int argc, char *argv[]) {
    // The main handling loop of the model.
    // argv[1] holds the treatment-type for the model:
    // i = remove mazzaella
    // f = remove fucus
    // c = make a 20x20 clearing
    // n = no treatment
    //
    const int NO_TREATMENT = 0;
    const int MAKE_CLEARING = 1;
    const int REMOVE_MAZ = 2;
    const int REMOVE_FUC = 3;
    const int LOAD_FROM_FILE = 1;
    const int RANDOM = 0;

    int rnum;

    if (argc < 2) { // no arguments given, then get input from user
        GetInitialConditions();
    } else if (argc > 2) {
        printf("\nERROR: too many command line arguments!\n");
        exit(1);
    } else {
        switch(*argv[1]) {
            case 'i': latticeType = LOAD_FROM_FILE;
                      treatmentType = REMOVE_MAZ;
                      latticeR = 20;
                      latticeC = 20;
                      runs = 100;
                      break;
            case 'f': latticeType = LOAD_FROM_FILE;
                      treatmentType = REMOVE_FUC;
                      latticeR = 20;
                      latticeC = 20;
                      runs = 100;
                      break;
            case 'c': latticeType = RANDOM;
                      treatmentType = MAKE_CLEARING;
                      latticeR = 50;
                      latticeC = 50;
                      runs = 100;
                      break;
            case 'n': latticeType = LOAD_FROM_FILE;
                      treatmentType = NO_TREATMENT;
                      latticeR = 20;
                      latticeC = 20;
                      runs = 100;
                      break;
            default: break;
        }
    }
    SetUpGraphics();
    SetUpResults();
    randNumCount = 1;
GenerateRandNums();
    rnum = 99*((int)RandNum()) + 1;
    for (currRun = 1; currRun <= runs; currRun++) {
        SetUpLattice();
        currIteration = 0;
        DoTreatment();
        CalcDensities();
        ShowResults();
        SaveIteration();
        for (currIteration = 1; currIteration <= iterations;
            currIteration++) {
            DoTreatment();
            CopyLattice();
            DoOneIteration();
            CalcDensities();
            ShowResults();
            SaveIteration();
            if (currRun == rnum) {
                SaveIterationLattice();
            }
        }
        // SaveResults();
        DisplayGraphics();
    } // SaveResults();
    CloseResults(rnum);
    ShutOffGraphics();
} // Main