REFERENCE ONLY

UNIVERSITY OF LONDON THESIS

Degree PhD Year 2006 Name of Author PIYATHAMNONGC HAR K

COPYRIGHT

This is a thesis accepted for a Higher Degree of the University of London. It is an unpublished typescript and the copyright is held by the author. All persons consulting the thesis must read and abide by the Copyright Declaration below.

COPYRIGHT DECLARATION

I recognise that the copyright of the above-described thesis rests with the author and that no quotation from it or information derived from it may be published without the prior written consent of the author.

LOANS

Theses may not be lent to individuals, but the Senate House Library may lend a copy to approved libraries within the United Kingdom, for consultation solely on the premises of those libraries. Application should be made to: Inter-Library Loans, Senate House Library, Senate House, Malet Street, London WC1E 7HU.

REPRODUCTION

University of London theses may not be reproduced without explicit written permission from the Senate House Library. Enquiries should be addressed to the Theses Section of the Library. Regulations concerning reproduction vary according to the date of acceptance of the thesis and are listed below as guidelines.

A. Before 1962. Permission granted only upon the prior written consent of the author. (The Senate House Library will provide addresses where possible).

B. 1962 - 1974. In many cases the author has agreed to permit copying upon completion of a Copyright Declaration.

C. 1975 - 1988. Most theses may be copied upon completion of a Copyright Declaration.

D. 1989 onwards. Most theses may be copied.

This thesis comes within category D.

☑ This copy has been deposited in the Library of UCL

☐ This copy has been deposited in the Senate House Library, Senate House, Malet Street, London WC1E 7HU.

A Dynamic Settlement Simulation Model:

Applications to Urban Growth in Thailand

KAMPANART PIYATHAMRONGCHAI

Centre for Advanced Spatial Analysis,

and The Department of Geography

University College London

University of London

February 2006

A thesis submitted for the degree of Doctor of Philosophy

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

ABSTRACT

Evolution at the urban-regional scale reflects complex characteristics in connection with both space and time. An effective way to study urban-regional growth and expansion is to build a hybrid simulation model to represent spatial phenomena at different scales, capable of generating different scenarios in an observable simulated time period. This work reports an implementation of such a simulation model - the Dynamic Settlement Simulation Model (DSSM) - which has been developed based on integrating two different cell-based modelling techniques: cellular automata (CA) and raster GIS. The CA model is used to dynamically simulate the growth of urban cells consistent with a set of probabilistic rules which reflect the system's complexity. The raster GIS module plays the role of controlling mutually static and dynamic constrained spatial variables that significantly affect urban-regional growth. Conceptually, DSSM has been developed using a theory of spatial organisation based on the nodal region from which we need to infer the growth process of urban and regional development over space and time. DSSM has been developed using an object-oriented programming approach; the model is composed of all the modules necessary to input the data, visualise the temporal simulation, and yield practical outcomes. For experiments, fabricated data at two different scales has been used to construct the model and explore different growth hypotheses. Furthermore, the model has been applied to two other sets of real data from two major cities in Thailand, Chiang Mai and Phitsanulok city, in order to evaluate its usability and efficiency. The simulation has produced acceptably accurate results when compared quantitatively to the actual land use/cover imagery of both cities. Finally, this work demonstrates how the model can be used as a part of a spatial decision support system (SDSS). It is able to provide other outcomes that represent the possibility of implementing predictive and scenario-based applications, which are applicable to urban and regional planning and related fields.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF BLOCKS

ACKNOWLEDGEMENTS

I would like to express my profound gratitude and great appreciation to my supervisor Professor Michael Batty for valuable advice, timeless guidance, encouragement and providing me in deep concepts and ideas during the entire research period. I am deeply grateful to Dr. Paul Densham for kindly being second supervisor and for their helpful suggestions.

I am very grateful for all of Centre for Advanced Spatial Analysis (CASA) for providing useful knowledge and facilitating all necessary equipments during the time of my studies. I would like to express my sincere thanks to the following, for providing necessary data for this research.

Office of Agricultural Economics, Ministry of Agriculture and Cooperatives, Thailand.

Department of Natural Resources and Environment of Naresuan University, Thailand.

Deep appreciation and thanks are extended to Royal Thai Government and University College London for the financial support during the entire study period. I would like to say thank you to all of CASA and Department of Geography staff for their technical support and taking care of the official formalities during the time of my study and my research.

Many thanks to all of my friends for their help to complete this research.

I would like to acknowledge the endless love and encouragement of my parents who honestly encourage only great things in my life.

CHAPTER ONE
Introduction

This dissertation reports the implementation of an urban computer model that was developed in order to simulate and visualise complex processes of urban and regional growth. The model implementation is essentially based on two well-established techniques of spatial simulation: cellular automata (CA) modelling and the spatial analysis of nodal regions. These techniques have been consistently developed into a suite of computer programs, which are supported and elaborated both by arbitrary spatial datasets and by real world spatial data which pertains to urban growth in small and medium sized cities in developing countries. This short first chapter presents the background to the study, identifies its key problems, describes its significance, and presents an overview of the methodology used. The chapter concludes by noting the limitations of the study and by outlining the progress of this research as it is developed in the rest of this dissertation.

1. Background of the Study

In recent years, there has been a growing literature on cellular automata (CA) techniques for simulating the growth and form of human settlements. Many urban development researchers have applied the CA model to both developed and developing countries situations where they have integrated such models with other techniques, such as GIS and complexity theory. These have been applied within spatial modelling in terms of traditional aggregate simulation models as well as more recent agent-based modelling structures. The Simland model due to Wu (1998), for example, integrates the CA model with multi criteria evaluation (MCE) techniques in order to derive behaviour-oriented rules of transition. Li and Yeh (2000) have improved CA models by integrating them with the spatial analysis of sustainable urban development as developed using constrained transition

rules. The SprawlSim model by Torrens (2002) has been developed using traditional modelling such as spatial interaction, discrete choice, econometric and input-output models. Most researches in this field have attempted to take advantage of several complementary theories to create such models. Moreover, these models have tended to combine the dynamic and self-organizing behaviour within CA models with a variety of spatial functions so that such models might yield more efficient and relevant results.

In consequence, at least three different points should be emphasized at this early stage; first, in the context of regional development, the growth of cities and towns in a regional system has become ever more significant as settlements have begun to fuse together and as globalization and regionalization have rapidly increased in their impact. Increasing from the urban to the regional scale intensifies the significance of changes in relevant variables and processes of urban-suburban growth. In fact, the concept of regional development based on the nodal regional system which is long established in spatial settlement analysis (Haggett 1965) can be used to explain such complex processes of regional growth. According to this concept, five processes which operate in space are relevant based on ideas about the spatial node, hierarchy, network, interaction and diffusion. These processes clearly raise questions which need to be considered in any realistic model or urban growth in a regional setting. For instance, 'how does a main city interact with other towns in its hinterland?', and 'how can the hierarchy of a city be measured if some attributes of the city's urban functions are changing over time?'. Moreover, each of these processes could be separately considered as submodels within a more integrated model framework.

Second, in a technical context, changes in the urban area of any city or small town over each time period may affect a multitude of constraints and conditions on existing space, such as changes of land use/cover, increasing or decreasing demand and supply for land, and changes in the infrastructure of cities. To handle such complexities, the traditional CA model is not comprehensive enough to represent all these changes in

functions and activities that precondition the space. Therefore, a model-based approach which we refer to as the Dynamic Spatial Model (DSM) was developed as a foundation for this study. The DSM approach is a spatio-temporal model based on embedding spatial analysis functions in GIS but structuring these within an explicit set of temporal events. Finally, the most important question that we pose in this thesis is how to consistently integrate these two different techniques to model the complexity of the human settlement system. That is the main objective of this study.

We develop this DSM model using the long–established modelling process which begins with model definition and specification. But unlike most other CA-based models, we set out to calibrate and verify the model against real world phenomena derived from two different areas in Thailand: Chiang Mai and its hinterlands in Northern Thailand and Phitsanulok city region in the central part of the country. Both are the significant case studies as the respective cities are widely promoted as growth poles for which there are explicit urban and economic policies at the regional level.

2. The Problem Statement

As mentioned earlier, there are three focus questions which this research will respond to. First, in case of urban and regional development, this study will attempt to measure interactions between cities, and how these interactions affect urban and regional expansion. The purpose is thus to simulate 'interaction' which we consider enable key differences between the cities to be measured and this leads quite naturally to methods for estimating interaction between them.

To achieve the first point, two other matters emerge which we can treat as a second question. This involves how several static and dynamic spatial variables reflecting factors which clearly affect a regional space can be efficiently organized and analysed. To visualize the process and to monitor

changes in these key variables, spatial simulation based on the DSM structure was developed. Two technical approaches were integrated in this model construction: the CA model structure and its embedding with a spatial model based on regional dynamics. Thus the second main purpose of the study was to implement a hybrid model which we call the Dynamic Settlement Simulation Model (DSSM) which is a fusion of CA and DSM. The hybrid model depicts the complexity of urban and regional development, building on well–established urban and regional theory. The last question that we set out to address is to ensure that this hybrid model is consistently applied, and focuses on how the model works in the real world situation. Actual datasets obtained from real urban situations in cities in Thailand were collected and as well as the theoretical experiments that we designed for testing the model, this data was used for validating the fit of the model to real urban growth patterns.

3. The Problem Rationale

Three different foci for the rationale of this study can be identified: first, there is the contribution made by choosing and developing a new model framework which integrates the two techniques of CA modelling and DSM. This can be represented in two other ways. First, and in general, DSM can be easily implemented in any proprietary GIS software, but the framework does encounter some difficulties when huge data sets required by such analyses and which are often hidden from the user, continuously cycle—in and –out of such classes within the program itself. This often doubles the processing time if it is integrated with a CA model. Moreover, implementing both the DSM (Dynamic Spatial Model) and CA in GIS software which is robust for the end user, is extremely inflexible when it comes to embedding model operations within such software. This study was therefore developed using native code in the form of the JAVA-based programming language; functions necessary to GIS were then selected and coded directly into the program. The JAVA programming codes developed in this study, in fact, can be transformed and extended into a DSSM

library which would be applicable to other applications and further research somewhat independent of the applications proposed here. In short, the programming codes which form a large part of this thesis are generic enough for wider development as a basis for spatial modelling.

Another advantage of the two fused techniques is more related to applications. As mentioned above, urban growth simulation without considering explicit dynamics is not enough. Many urban management officials, planners and policy makers cannot gain enough benefit from urban simulation because many models do not exploit the wide array of other spatial and non-spatial activity data. To fill this gap, the model in this study was set up as a dynamic spatial analysis module which could be used to manipulate changes in any of the dynamic factors as well. Although the model developed here does not cover all dynamic factors in the real world, it confers substantial benefits in urban policy making while the final framework from this research will hopefully guide new researchers in expanding their ideas about how to model rapidly growing towns and cities in developing countries as well as developed countries. Although the model has been developed with the former applications in mind, its structure is generic and is equally applicable to other parts of the world as well as South East and East Asia.

Second, theoretically, application of the regional concept within an urban simulation model is important to urban-regional management. For instance, urban planners usually require regional information for preparing local and structure plans where the focus is on designing new residential areas or new central business districts while land agents may need to know spatial trends in city growth both to avoid and to anticipate exorbitant profits associated with land development in the near future. Obviously, a city cannot evolve without any interactions or contacts with other towns or cities, and thus the concept of spatial interaction which, in this case, is used to detect directions and trends in urban growth, has also been implemented as a module in the DSSM. This is so that such interactions can be visualized dynamically in terms of the effects of urban growth on regional development.

Third, the program was developed in a flexible programming language which enables the graphic user interface (GUI) to be changed depending upon the end-user's applications. End-users can prepare map layers in proper format and define other input variables to make 'a scenario' which simulates urban change in a city. Eventually, we consider that the model can be improved within a wider framework of spatial decision support which would be embedded more generally within the plan-making process.

4. Overview of the Methodology

As previously mentioned, the purpose of this study is delineated in both implementation and application. In this section, we will briefly describe the main components of the model. Obviously, details of the model implementation will be explained in later chapters but an overview of the conceptual framework is required in order to implement DSSM. All necessary classes belonging to the traditional form of JAVA programming language were determined to depict the entire structure of the model. The main components of the model include: CA and diffusion modules, the DSM module, the spatial interaction module, the population and economic estimation module, and the nodal manipulation and hierarchy modification module.

The diffusion model which is rooted in the traditional 'Hägerstand Mean Information Field' (MIFs) principle (see detail in Haggett 1975) was utilized to generate new clusters of developing cells in the growing urban space. The CA module was developed based on a simple set of transition rules in probabilistic fashion. Consequently, the cells generated from the diffusion module would 'grow' and 'spread' within this process. To inject a little more reality, other constraints were considered within the simulation model, derived from spatial data and thematic maps. These constraints were realized as 'dynamic' and thus part of DSM module in the model. Centres and nodes in any area were generated based on the nodal and hierarchical modification module dealing with the population and other economic information simulated in turn by the population and economic

module. From these processes and functions, it is then possible to visualize the population rank size distribution of cities or towns in a regional system. To simulate the relationship between cities or towns, the spatial interaction module was based on several established concepts, such as the gravity/entropy model, potential surface models (more detail see Haggett et al. 1977), and general notions about the sphere of influence for various kinds of urban forces (see Colby 1933).

In the implementation stage, arbitrary datasets were primarily assumed in terms of defining dynamic constraints within the DSM module. Data layers are included such as: river maps, accessibility maps, boundary or zone maps, and land use/cover and topographic feature (slope map) maps. In addition, other thematic maps were analysed and created from primary map layers, such as the population map, dasymetric density maps, land requirements maps and nodal maps. Two testing areas with different spatial dimension - 50 x 50 pixels and 250 x 250 pixels - have been adopted in testing the model. Other parameters were assumed or guessed and thus in each run of the model, all modules were completely parameterized. Certainly, these parameters are all adjusted in the final stages of model use where the user has control over the process.

On the other hand, in evaluating and applying the model, real datasets absolutely similar to the structure and meaning of arbitrary data used in model testing and experimentation were prepared and replaced into the model. Two different satellite images, approximately 10 years apart in date of sensing, were collected and classified. These were used for two purposes: for setting as the seeds which drive growth in the model or as initial states in some map layers in the simulation. Of course such data was also used for verifying whether the simulation results were close to the real world phenomena. Other map layers were prepared from secondary data or collected as primary data in the field.

Final stages of implementation in this study concerned improvements to the graphic user interface. The outcomes, from parameter testing with the model using both assumed and real datasets, were then used in order to

modify the interface. This provided ultimate flexibility for users in defining possible scenarios for urban and regional growth phenomena and testing such scenarios with the model.

5. Delimitations of the Study

The purpose of DSSM is to develop a prototype model that integrates two different but complementary spatial techniques for simulating and visualizing urban-regional growth where there are many different variables or factors influencing such growth. This can be considered in two contexts. First, in the context of developing the model, limits posed by the spatial dimension in terms of resolution affected the scale and resolution of map inputs which in turn depend upon computational limits associated with acceptable computer processing.

Another matter relates to the variables, factors, parameters or conditions/constraints used in the model. The model imitates only a part of the real world phenomena which is associated with the problem application and with its data: no perfect models exist and this model, like all others, is not able to absolutely represent the real phenomena. It is therefore important to note that there are still many other factors influencing the existing phenomena with both spatial and non-spatial effects that were not used in this study. Only the main variables and parameters that obviously influence the rapid growth of urban areas and relate to changes in regional structure were contemplated and used.

As mentioned in a previous section, in this study the list of spatial factors were strictly determined including: rivers (barrier maps), transportation, boundary or zone maps, land use/cover, topographic features (slope maps), population, population density dasymetric maps, and land requirements maps. In addition, to drive the model and create the dynamic thematic maps, non-spatial aspects had to be realized as well: predicted populations though time were projected and set as active variables to generate thematic maps such as density, land requirements

and nodal region maps. These are the major spatial and non-spatial dynamic variables that this study invoked and this reflected both the length of the study and data availability within the context of the applications made. These will be discussed extensively in later chapters.

6. Outline of the Dissertation

As explained earlier in this chapter, the DSSM is based on a lot of detail which mixes technical and theoretical concepts e.g. cellular automata, diffusion models, dynamic spatial models, spatial interaction surface models, and the nodal regional system. These are often difficult to clarify. This thesis is therefore laid out to represent these technical ideas starting from a very general perspective to very detail specific algorithms which can enables the reader to understand all the crucial points of the model. Moreover, the thesis also demonstrates some experiments that represent how the model can apply to both hypothetical and real world cities, thus guiding the reader to learn more about the context of the model mechanism and its applications. The details of each chapter are described below.

This thesis contains 8 chapters which each represent a different focus on the problem. Chapter 2 considers separately the structure of ideas, theories and concepts behind the model construction. It shows contexts, approaches to research, and development. It presents a review of related urban modelling using cellular automata and GIS together with some detailed arguments for the definition and use of nodal regional spatial analysis as well. And finally, the chapter also represents an outline for the implementation of the DSSM based on an integration or synthesis of possible techniques from CA and GIS.

Once, the model layout has been presented, Chapter 3 represents more detail concerning the general conceptual framework of DSSM. This takes the reader from very rough outline of the entire concept to details of each portion of the model, i.e. dynamic maps, cellular automata and diffusion

models, node generation and hierarchy ranking, land use/cover change rules, population dasymetric mapping, and accessibility and spatial interaction. This chapter concludes with the practical structure of the model, which is consequently to be used as the blueprint to construct the actual computational model.

Chapter 4 depicts detailed programming matters. It begins with some concepts concerning the object-oriented programming approach (OOP) and the unified modelling language (UML). Each part of the DSSM structure as discussed in Chapter 3 is described separately using UML diagrams and pseudo programming codes.

The context to Chapter 5 relates to experiments with the implemented model in terms of a set of testing data. This chapter represents characteristics of an idealized space for the model and how to prepare model cell spaces at two different scales in terms of different spatial data layers. Each component of the DSSM discussed above is examined and the model is illustrated with many visual results. Lastly, Chapter 5 reports and visualises results and quantitative information as shown in different varieties of charting from simulations with the whole DSSM system.

Chapter 6 explains the background, location, physical and socio-economic information, and the importance of both case study areas, i.e. Chiang Mai and Phitsanulok city. This chapter gives some basic information about these two cities as exemplars for towns and cities in Thailand in general. This chapter also shows how we prepare the data starting with defining the study areas and finally presenting the thematic maps that are used to examine the model in Chapter 7.

Chapter 7 firstly depicts the model evaluation beginning with how to design and prepare experiments for both cities: synchronising reality time and simulation time, preparing parameters and variables, and defining different scenarios. This chapter also provides all results generated from the simulation of each scenario and indicates acceptable results using an assessment method in quantitative way. The last part of this chapter

represents possible applications of the DSSM, which can predict future growth of both cities and the capability of the model to implement 'what-if' scenarios.

Chapter 8 concludes the study from two general perspectives: the first based on the model and the second on the applications. In terms of the model, this represents improvements to major parts of the model, e.g. expanding temporal variables in GIS, increasing spatio-temporal aspects of the CA model and representing complex behaviours within the model's transitions rules and hierarchical structures. For applications, it discusses in the context of real world urbanised processes and the nodal regional system, some limitations of the current model. This leads to putting forward some further possible expansion of the DSSM suggesting potential research areas. This chapter finally envisages the DSSM as an essential component in a model library based on general spatial decision support systems, thus building on the idea that DSSM is a generic framework suitable not only for other kinds of urban modelling but also for other kinds of decision support and formalized computer-based planning.

CHAPTER TWO
Models for Urban Development

1. Introduction

This chapter presents a review of the theories, models and techniques that are used ultimately to compose the model but it also discusses the evolution of the field of urban and regional modelling insofar as the ideas discussed are used in model development. It is written to introduce readers to two key themes that are crucial to the DSSM model. First, the model was constructed based on an integration of several theories within urban and regional development, using two techniques of space-time modelling based on cellular automata (CA) and that variety of spatial dynamic modelling that is generally associated with the use of the GIS as a tool. Hence, it is useful to introduce some basic ideas about these concepts and techniques so as to understand the relative importance of these ideas as they will be shown in latter chapters. The second issue involves reviewing recent improvements in urban and regional models, which are important to reflect on how the model in this study has been developed and implemented and how future models for urban and regional development should be constructed and applied in the future.

For ease in describing the concepts behind the model in detail, this chapter has primarily divided the discussion of models into three components: cellular automata, spatial modelling, and geographical concepts for urban-regional development. Each will be explained in both their theoretical concept and their fitness for purposeful usage in the context of current urban-regional research. Finally the last part of this chapter will critically discuss considerations for constructing the model based on integration of these concepts and techniques. Up to this point, all concepts will then be summarised and used as a basis to explain the systems in the model that are developed and implemented in the next few chapters. We will begin with a thorough review of CA models and principles, then move on to deal

with spatial modelling within GIS, particular raster-based models, then deal with geographical theories of regional-nodal development, finally pulling all these ideas together in anticipating the model structure that we call DSSM.

2. Cellular Automata and Urban Applications

> "Cellular automata are examples of mathematical systems constructed from many identical components, each simple, but together capable of complex behaviour" (Wolfram 1984).
>
> "Cellular automata are systems of cells interacting in a simple way but displaying complex overall behaviour" (Xie 1994, citing Wolfram 1986).

These definitions from Wolfram, one of the main popularisers and proponents of CA, provide a good sense of their mechanisms. All such definitions inform us that CA deal with simple interaction between cells but simultaneously provide complex behavioural outcomes. CA usually consist of four elements which are defined as cells, states, neighbourhoods and rules (Li and Yeh 2000). White and Engelen (2000) have argued somewhat more specifically that they consist of: 1) a grid or raster space; 2) a set of states which characterise the grid cells; 3) a definition of a neighbourhood of each and every cell; 4) a set of transition rules which show how cells in a neighbourhood can affect the state of other cells in the same neighbourhood; and 5) a sequence of discrete time steps.

The most basic CA procedure is thus simply described by the statement: "at each *time step*, a *state* of a *cell* can be transformed based on a set of *transition rules* applied to its *neighbourhoods*", and this implies of course that cells have their own states. For instance, in the CA Game of Life invented by John Conway in 1970 (see Gardner 1970), at each time step there are two possible states a cell can take on: *alive* or *dead*. To turn one state into another, the transition rules are simple and threefold: for instance if a dead cell has exactly three live cells in its neighbourhood, then the state of the cell changes to become alive in the next iteration. If a live

cell has two or three live cells in its neighbourhood, the cell survives. And in all other cases, a cell dies or remains dead as can be seen in Figure 2.1 where each line of the figure shows each of the three transition rules in terms of cells that become alive (blue surrounded by yellow) and cells that die (blue with a red cross).

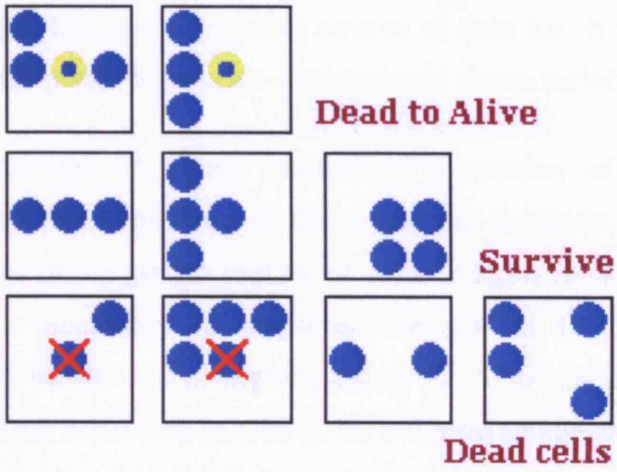


Figure 2.1 Conway's 'Game of Life': Transition Rules
Source: http://www.math.com/students/wonders/life/life.html

Cell Space

Typically, the cell space of a CA applied to urban and city modelling is assumed to be two-dimensional, rectilinear and homogeneous (White and Engelen 2000). However, some scholars have suggested more complex shapes of cell space such as those based on hexagonal tessellations (see Weimar 2000). It is clear however that almost of all CA applications in spatial models from the past have been developed using the rectangular cell space as seen in several works (see Batty 1998; 2003, White and Engelen 2000, Xie 1994, Li and Yeh 2000; 2001, Clarke et al. 1997, Almeida et al. 2003, Wu 2002). This is possibly because most of geographical or spatial applications are conveniently modelled on a two-dimensional space, and from this assumption, Wagner (1997) states that the basic requirements of a CA are provided by raster GIS, which in turn is a similar kind of two-dimensional space. Using the analogy with raster GIS, this is not only appropriate in implementing the model in rectilinear

two-dimensional space, but it also provides the convenience of taking advantage of the raster GIS functions. As Wagner (1997) claimed, in applying raster GIS, such a model could share some raster-based functionalities such as multiple bit planes, neighbourhood structures, a language for specifying rules, and the ability to apply rule sets synchronously with the GIS platform on which it was based. Moreover, satellite images that are a possible source of data for the CA models also follow a similar raster based two-dimensional characteristic.

A two-dimensional CA space consists of a number of cells which are determined by the dimension of the space; for instance, 50 rows and 50 column spaces imply 50x50 matrix of cells or 2500 cells. From this point of view, it is necessary to consider the size of the space whenever attempting to model a real system using CA. If considering the CA cell space as a raster structure in GIS or as a digital image structure, this will definitely determine the spatial resolution, which in turn fixes levels of detail or the scale of the space. For instance, if a space represents a cell as 10x10 sq. m., this certainly has a better quality than cells based on 100x100 sq m. resolution. However, the greater the resolution, the greater the number of neighbourhood comparisons that have to be made in even the simplest of CA such as the Game of Life, and this can cause excessive model computation. Hence, there are no clear or strict rules or restrictions on choosing the size of space or cell resolution from a theoretical point of view. It is all conditional on the purpose of the applications and processing power of the computer which will in the last analysis determine a suitable cell size or resolution. For instance, White and Engelen (1997) constructed a CA application of St Lucia at 250 m; the constrained CA model of Li and Yeh (2000) set the cell space as 619x889 pixels with 50 m resolution as applied to the city region of Dongguan in the Pearl River Delta in China; and Cheng and Masser (2002) chose 100 m resolution space to develop their model of Wuhan in China in order to avoid the overload posed by model computation. Clarke et al. (1997) have constructed a self-modifying CA to simulate the historical urbanisation in the San Francisco Bay area, and in this case although their cell size was 100 m, the model had a very

large number of cells taking many hours to run on a high performance computer

Cell States

One of the simplest examples of cell states is shown in the Conway's Game of Life, where cell states are set as two possible conditions – dead or alive. Each cell in grid space can exist only in one predetermined state at any one time (Deadman et al. 1993) as can be seen in Figure 2.1, where the central cells can be only dead or alive at any one time. These cell states correspond individually to different colours in visualisation and different values in their computation. For instance, dead cells in the Game of Life are shown as "white" or "0--zero", live cells, on the other hand, are represented as "blue" or "1--one". Depending on applications, cell states can be predefined for more than two classes and they can even be considered as either discrete or continuous (Deadman et al. 1993).

In applications to natural systems, cell states can be used to represent examples such as sediment load in seawater, groundwater levels or soil moisture (White and Engelen 2000). Li and Yeh (2000) have defined cell states across varying levels as gradual changes in level; not two states such as 'undeveloped and developed', but defining a series of levels ranging across different degrees of 'partial development'. Batty and Xie (1997) developed the DUEM model (Dynamic Evolutionary Urban Model) assigning 4 classes of land use to cells in the simulation based on housing, industrial, commercial and transportation. Cell states, in other applications such as disease outbreak simulations used in spatial epidemiology, can be defined as the three standard classes of susceptible to infections, infected and recovered or non-infectious states.

Cell states also relate to the cell space. For example, if an urban application assigns 100 m resolution to the cell space and cell states are set as urbanised and non-urbanised, whenever a cell is created within the cell space, this implies that 10,000 sq m are immediately urbanised. In the same fashion, if modelling the outbreak of a disease in an area with the

same dimensions, population affected in the same size of cell suggest that a 10,000 sq m outbreak area begins to generate the infection. All this is saying is that it is crucial to choose a cell size that is consistent with the representation of appropriate states. If the cell size is too big, it will suppress important details and if it is too small, it will reduce the efficiency of the model. Hence, to implement a CA application, cell state and size need to be considered in terms of how reasonable the choice of each is as representing the real world.

Neighbourhoods

Each cell in grid space is surrounded by a neighbourhood of cells, which determine its size and shape. In each discrete time step, the existence of different cell states in the neighbourhood affects its 'central cell'. Classically, two popular patterns of neighbourhood are mainly utilised in CA applications: the Von Neumann (four cells adjacent to the sides of a central cell) and Moore neighbourhood (the eight adjacent cells) (see Figure 2.2).

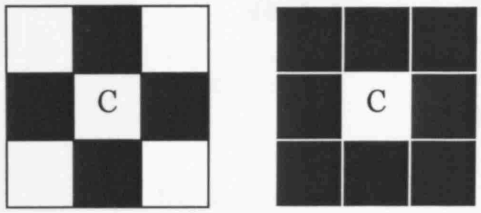


Figure 2.2 Classical Neighbourhood Patterns

However, some scholars (White and Engelen 1997; 2000, Li and Yeh 2000) have suggested that in the case of human systems, the vicinity of influence is possibly much larger. Hence, spatial and urban models probably need to define larger neighbourhoods to cover all influential matters on the central cell's range, which in turn represents all the action at a distance that can take place around any cell in question. Figure 2.3 represents alternative configurations of the neighbourhood.

The analogy that should be made is to Tobler's first law of geography which says that: "everything is related to everything else, but near things are more related than distant things" (Barnes 2004). The idea is practically

related to defining the CA-neighbourhood at least in two contexts. First, in a CA model, a cellular automaton is a system where all cells on the space have a relation, and second, their neighbourhood is critical as 'near things' affect the state transformation of its central cell. From this point of view, it is clear that the idea of CA-neighbourhood matches up to some of the geographical measuring methods, based on crucial operators such as those in map algebra which relate to raster-based GIS. Wagner (1997) summarises some of the map algebra operators that have and have not been implemented using the CA called Cellular as shown in Table 2.1. Some concepts of raster-based map algebra used in this model will be discussed later in this chapter.

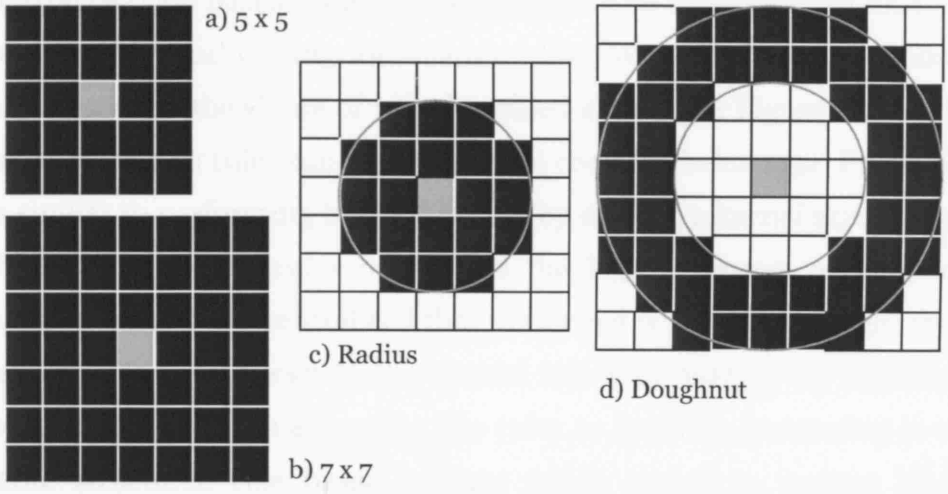


Figure 2.3 Alternative Neighbourhood Patterns

Table 2.1 The Map Algebra Operators (Tomlin 1990) in Cellular Automata
Source: Wagner (1997)

Operators				
Focal	**Focal phrases**	**Incremental**	**Local**	**Zonal**
Bearing	At Distance	Area	*Combination*	*Combination*
Combination	At Direction	Aspect	Difference	*Majority*
Gravitation	Spreading	Drainage	Majority	*Maximum*
Insularity	*Radiating*	*Frontage*	Maximum	*Mean*
Majority		Gradient	Mean	*Minimum*
Maximum		*Length*	Minimum	*Percentage*
Mean		*Linkage*	Minority	*Percentile*
Minimum		*Partition*	Product	*Product*
Minority		Volume	*Rating*	*Rating*
Neighbour			Ratio	*Ratio*
Percentage			Root	*Root*
Percentile			Sum	*Sum*

Operators				
Focal	**Focal phrases**	**Incremental**	**Local**	**Zonal**
Product Proximity *Ranking* Rating Sum *Variety*			*Variety*	*Variety*
Notes: Operators in italics are not currently implemented in Cellular.				

Transition Rules

The transition rules are the most important part of regulating the state of the central cell in the next iteration of any CA process where the future central cells are based on the current states of its neighbouring cells. These rules are the real driving force behind the CA model (Torrens 2000). As can be seen in the Game of Life described above (see Figure 2.1), a simple set of transition rules can generate quite complex behaviour. Practically, it is similar to performing image filtering by moving a kernel across a space; then the transition rules operate as the logical process applied to the neighbourhood of a central cell that the kernel is passing through; the new cell state is then drawn to the central cell according to the logical rules which may vary from averaging like rules as in image processing to much more structured rule based systems which accord to various kinds of physical and human behaviour.

The transition rules can be very simple like the Game of Life or can be more complex depending on states of the cell and types of cell state (discrete or continuous). They can be applied to several patterns of neighbourhood as shown in Figure 2.3. For instance, the transition rules in the Game of Life are applied to the 8 adjacent cells of the Moore neighbourhood. Recently, several urban models have been developed using the CA concept but instead of the most simple of transition rules, they have been improved to better suit their applications where action at a distance is important. Some applications introduce a stochastic element into the transition rules enabling cells to change state only if certain probability thresholds are met in terms of the state of the cells in the

relevant neighbourhood (see White and Engelen 2000, Xie 1994, Batty and Xie 1997, Almeida et al. 2003, Wu 2002, Ward et al. 2000).

Time Steps

The particular time step in most CAs is discrete, and takes place after all procedures in the model have been executed. If the applications are developed for exploring complex spatial systems without concern for reality, the time step will conveniently be defined as one of iteration. On the other hand, in CA applications in several research fields, it is usually necessary to find a simulation time scale corresponding to the time scale in the real world. Cheng and Masser (2002) create a CA model for simulating urban development in Wuhan city in China from 1993 to 2000. They fixed 7 years of urban growth as 50 iterations based on the averaging of temporal growth patterns of each class (state) which they computed from real world observations of urban change. On the other hand, Li and Yeh (2000) set up scenarios of their CA land consumption model, predefining the different time steps in the simulation for each particular scenario so that they were able to ensure that the amount of final land consumption would be equal to the actual or the optimal. Again, as in defining neighbourhoods and levels of spatial resolution, there are few if any guidelines as to how to match CA simulation time to the reality. It is thus of concern as to find a compatible simulation time scale that thoroughly corresponds with the real time scale of application in question and this remains an important research question.

Hybrid CA

Recently, several applications of CA particularly to spatial problems are not as simple as those discussed above. In contrast to the simplest and strictest CA, they have been developed by manipulating procedures within CA to be more intricate in that the concern has been to adapt the simulation model as accurately as possible to the real world phenomena. It is clear that there are currently two trends of development for CA applications. First, there is a trend to enhance the material phenomena

that CA is applicable to in the urban realm, and second, there is the development of hybrid CA applications.

Generally, it is possible to manipulate CA by playing on two sets of operators from the main CA components: the structural and functional rules. First, the spatial structure of the automata is in its number of dimensions, the character of its cells (square or hexagonal, for example), and the types of its neighbourhoods. The second one relates to the number of states and the transition rules (Rennard 2000).

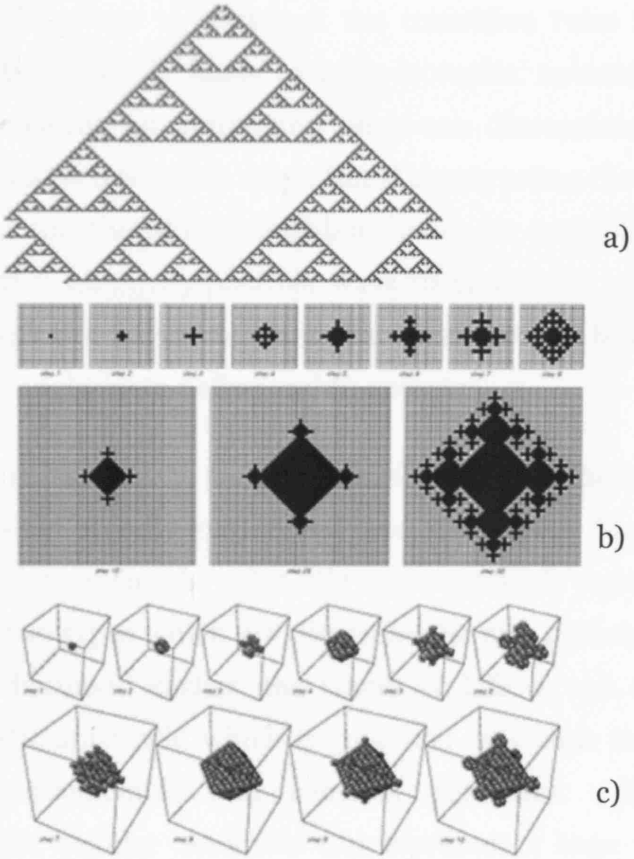


Figure 2.4 CA Simulation with Different Numbers of Dimension: a) One- b) Two- and c) Three-Dimensions
Sources: a) Rennard 2000, b) and c) Wolfram 2002

Besides increasing the number of dimensions for the CA as shown in Figure 2.4, another clear example of this shifting in the first group is in the choice of neighbourhood patterns shown in Figure 2.3 above. Some of the more recent CA applications have been built up in much more variety than

the traditional 4- or 8-neighbourhoods (see White and Engelen 1997, White and Engelen 2000, Li and Yeh 2000, Dijkstra and Timmermans 2002, Barredo et al. 2003). This leads to CA algorithms and processing to be more extensive and time-consuming, but probably provides better theoretical results, and perhaps, offers more accurate outcomes. In this matter, some researchers have kept the characteristic of simplicity of classical CA building their models with 4- or 8- neighbourhood (see Clarke et al. 1997, Ward et al. 2000, Batty 1998).

For the second group, firstly varying the number of cell states makes many more possible ways to construct the transition rules for the automata, which in turn usually leads to more complex automata outcomes. For instance, creating an automaton on a one dimensional space with two states gives $2^3 = 8$ possible rules. But if constructing the automaton in two dimensions (on the Moore neighbourhood, for example) with 5 states, it becomes $5^9 = 1{,}953{,}125$ possible ways to create the transition rule. This kind of explosion gives an immediate insight into how complex such a simulation might be to define and to operate.

The CA transition rule is another crucial concern in the second set of issues that has very broadly researched and developed. Many scholars have focused on improving the rules. We may commonly separate CA transition rules into two types which are deterministic and probabilistic (stochastic) rules. As discussed earlier, the Game of Life is one created using the deterministic approach, which is governed by simple transition rules but offering unpredictable and complex outcomes. Even though the deterministic cellular automata can represent a huge range of patterns generated based on different kinds of transition rules (see Wolfram 2002), there is a sense in which probabilistic models are preferable. It is obvious that there are, presently, a variety of the CA projects in diverse fields that are making an effort to amend deterministic models by specifying them as stochastic in the quest to move closer to the way natural systems appear to behave. As Wilensky (2003) says:

> "Stochastic cellular automata are models of 'noisy' systems in which processes do not function exactly as expected, like most processes found in natural systems. The behaviour of these cellular automata tend to be very rich and complex, often forming self-similar tree-like or chaotic behaviour".

The probabilistic approach has changed the concept of simplicity of classic cellular automata. It has transformed CA simulation into much more unpredictable simulation worlds in which accident and historical path dependence is an important feature. These worlds are much more intricate and capricious than the deterministic as is clearly demonstrated in an increasing number of applications.

Batty and Xie (1997) have suggested a CA application using several probabilistic functions as an alternative. In their model, cell states that include housing, industrial use, commercial use, and transportation are governed by transition rules based on random numbers and thresholds which reflect the neighbourhood function. The model resulted in a panorama of 'possible worlds' as it represented probabilistically the generation of each cell state on the space. Consequently, the CA applications have been developed in variety of probabilistic and indeterminate techniques such as those employing Bayes' theorem (Almeida et al. 2003), fuzzy set theory (Liu and Phinn 2002), and artificial neural networks (Li and Yeh 2001).

Hybrid cellular automaton is another vital approach that has been increasingly developed of late in applications of CA to real world applications. CA applications which mix both the deterministic and probabilistic approach return outcomes which mirror spatial complexity. However, real world phenomena are more multifaceted and often cannot be automated or generated by cell states and their neighbours only through local causation. This has forced many scholars whose desire to build models a bit closer to the real world phenomena to seek how other spatial and non-spatial factors can be integrated into their models.

Constrained cellular automata has become a widespread hybrid CA technique which has been extended from classical CA. In addition to

governing central cells using endogenous interaction of its neighbours through the set of transition rules, the cellular model has also been dictated by exogenous causations which are embodied often as constraints. Constraints can be generally characterised into three types: local, regional and global. Local constraints contain spatial information for each cell; regional constraints include aggregated spatial information for a wider area; and global constraints are generally characterised by temporal and non-spatial information which pertain to the wider context, often the so-called rest of the world sector (Li and Yeh 2000). Constraints can be physical characteristics such as the impacts of slopes on development, and/or the existing road network (Clarke et al. 1997) or mandated features, associated with governing the simulations to meet regulations and costs associated with land suitability (Batty and Xie 2005, Li and Yeh 2000, White and Engelen 2000), environmental suitability, urban form or issues related to land consumption (Yeh and Li 2001).

It becomes clear that some parts of the procedures within constrained CA are very similar to the key GIS raster-based operators as discussed by Wagner (1997). From this point of view, it is possible to use GIS data and to integrate a variety of GIS functions to build more reliable CA real world applications (see Table 2.1 above). These have been called 'Geographic Automata Systems' (Torrens 2002) or more colloquially 'Geoautomata'. There is growing literature on integrating cellular automata techniques and GIS for simulating various spatial phenomena such as urban growth, wildfire (Killough 2003), epidemics (Fu 2002), and traffic flow (Hafstein et al. 2004). In a wide variety of research disciplines, a huge number of applications are picking up on the notion and efficiency of CA in their role as both specific tools as well as in extended techniques for modelling such kinds of dynamic spatial phenomena.

CA and Urban Models

Previous sections have described the general CA concept as well as its development from its classical to a more modern form. The CA concept has several benefits in terms of applications which we have tried to integrate

both in spatial terms and through issues of time. In this section, we will review in more detail as to how urban applications can be exploited using the CA concept.

Batty et.al. (1997) have briefly compiled a useful history of the development of urban models which can be summarised as follows. The first attempts to build mathematical models of urban systems were begun in early 1950s by Hägerstand whose diffusion models for human migration based on the population record in Sweden were early forerunners of models that spread growth and change in diffusion like ways. Chapin and Weiss modelled the land development process articulated as a cell-space (CS) model with cell states and neighbourhood effects (Batty 1997 cited Chapin and Weiss 1968). Lathrop and Hamburg (1965) built another cell-based simulation for Western New York State in 1965. Strict CA models began to develop in the urban domain from theoretical quantitative geography in the 1970s, principally by Waldo Tobler. In 1974, he started to explore the technique that strict CA might be applied in geographic systems and finally published his important paper 'Cellular Geography' in 1979. That abruptly changed the face of cellular automata in geographic applications. In the 1980s, inspired by Tobler, Couclelis kept on this track until real applications through computer graphics, fractals, chaos and complexity which to agent-based models were instigated in late 1980s (see detail in Batty et al 1997).

CA applications became more reliable when fused with GIS concepts and this began in the 1990s after the GIS boom in the 1980s, as well as due to the growth and availability of standard UNIX workstations and personal computers during this period. Moreover, in the 1990s, the low cost of high performance personal computers and GIS software generated widespread interest and development of spatial applications both in GIS and in CA simultaneously, and in parallel.

Since 1990, an increasing number of research applications have been implemented using cellular automata and related technologies to study urban and regional growth. Some of these have focused on extending the transition rules with stochastic features. Some have continued to improve the model to

make it more realistic by expanding the various types and extent of neighbourhoods, defining more possible states, transforming discrete cells to be continuous states, creating possible simulation worlds based on hypothetical urban growth patterns, and testing urban theories with simulation. Some have been associated with calibrating models with actual data, integrating other concepts from complexity theory such as fuzzy-set theory and artificial neural networks with the model, and building fused or integrated models to critically predict the future growth of the urban system. However, most have been developed in the fashion of hybrid CA integrating with further related aspects.

The following are worth noting as representing some examples of CA since 1990. Deadman and colleagues (1993) built a stochastic CA simulation model to predict patterns in the spread of rural residential development in 80 sq km of rural countryside near Toronto, Canada by setting up 2 scenarios based on differing sets of transition rules. Another prominent model, DUEM (Dynamic Urban Evolutionary Modelling), invented by Xie (1994), encompasses an outstanding improvement in the transition rules for simulating transportation which interacts with other urban land use features. In 1997, based on his SLEUTH model (Slope, Land cover, Exclusion, Urbanisation, Transportation, and Hillshade), Clarke and colleagues applied the model to simulate historical urbanisation in San Francisco and other large US city regions. The model was implemented using efficient transition rules applied to Moore neighbourhoods and constrained by the factors as contained in SLEUTH (see Clarke et al. 1997). The field was pulled together in several practical CA applications which were published in a special issue, "Urban systems as cellular automata", in *Environment and Planning B* in 1997. More recent applications of urban models based on CA are given by Benenson and Torrens (2004) and Batty and Xie (2005).

Considering material in terms of publications and web sites after 1997, it is clear that most of CA researches have developed more complicated techniques, more tuned to the real world using higher spatial resolution. The focus on actual cities represents the real features of space, relating to prominent policy approaches such as sustainable development, more variety in combining new other approaches, more factors for constrained

CA and more use of GIS functionalities, either through direct embedding in GIS packages or through some more loosely coupled manner. Most have been developed using higher resolutions for actual data instead of fabricated and low-resolution urban space (see Li and Yeh 2000; 2001, Yeh and Li 2001, White and Engelen 2000, Ward et al. 2000, Wu and Martin 2002, Almeida et al. 2003, Barredo et al. 2003, Loibl and Toetzer 2003). This leads to greater accuracy and provision where comparisons and evaluations against real data are essential (Ward et al. 2000, Yeh and Li 2001, Francesca and Giovanni 2003, Almeida et al. 2003, Barredo et al. 2003, Loibl and Toetzer 2003).

The constrained CA approach is still an important one which is leading to an increase in the number of factors and techniques being embedded into ever more realistic models. Li and Yeh focussed their efforts on the development of an integrated constrained CA and GIS on a GIS package platform. This could conveniently use the GRID-based or raster structure from the GIS software to model the urban form of sustainable development while adding some quite sophisticated routines for calibrating the model to data using artificial neural networks. They used their model in policy formulation by defining top-down policies based on sustainable urban development which were crucial to their model development, e.g. agricultural suitability and land resources (Li and Yeh 2000; 2001, Yeh and Li 2001). Ward et al. (2000) included stochastic features in a constrained CA specific to transport network planning, whereas White and Engelen (2000) have implemented a model inputting map layers of land suitability for low-density housing, zoning of low-density housing, accessibility values and other economic policy measures/constraints. Some works have concentrated on other relevant factors in addition to transportation and land use; for instance, spatial interaction between built-up areas (Loibl and Toetzer 2003), whereas Wu and Martin (2002) have embedded a population surface model into a CA, cementing the feedback between growth and distribution which updates information constraining urban expansion (see Wu and Martin 2002, and Martin and Wu 1999).

Constrained CA is also the key to combine other techniques especially GIS with the spatial CA paradigm. A number of researchers have built their CA models in association with GIS environments (Li and Yeh 2000; 2001, Yeh and Li 2001), while some have brought GIS operations such as buffering functions, focal statistics, and surface analysis (see Li and Yeh 2000, Almeida et al. 2003, Wu and Martin 2002, White and Engelen 2000) into the definition of constraints within the model.

In summary, several applications noted above clearly show the possibility of including more constraining factors, which can be both top-down and bottom-up and both static and dynamic within CA. Many of these can be derived from GIS functions, thus keeping CA closer to the reality of the data. Nevertheless, this section can be finally summarised from the specific to the more general with some noticeable trends in CA research in the urban-regional field since the year 2000. Torrens and O'Sullivan (2001) suggest three principal goals: 1) the exploration of spatial complexity, 2) the development of abstract models for testing hypotheses and theories relating to the city and 3) the development of operational urban models to inform policy-making and urban management in the real world (see also Torrens and O'Sullivan 2000, and O'Sullivan and Torrens 2000).

The model developed in this study is based on the second and third of these trends suggested by Torrens and O'Sullivan. Integrating the constrained CA and dynamic spatial model using GIS is a major objective of this study, and embedding some classic urban-regional theory into the models is another goal. This is all to be discussed in a later section.

3. Spatial Analysis, GIS Modelling and CA

As described in the previous section, urban CA modellers have tended to integrate their models with the advantages of GIS capacity and availability. The capability of spatial database management and a variety of functionalities especially for raster- or grid-based structures have been applied to derive more useful spatial information for the models. This

section will discuss in much more detail the kind of functionalities and analysis methods involved so that readers can quickly gauge how useful GIS functions applied in this study actually are and how they can be used in further development.

Raster-Based GIS

GIS data structures can be broadly classified into two types: vector and raster. The vector data model represents geographic features similar to the way we represent them in conventional street and road maps. Each location (*point*) is recorded as a single x and y coordinate referred to by the Cartesian coordinate system. *Lines* are recorded as a series of x and y coordinates and enclosed to build up closed or open structures termed *polygons*. All are connected by a relational geometry called 'topology'. In topology, it is convenient to link to other non-spatial information in the form of geo-databases. Raster data, on the other hand, is based on image formats using small grid squares called 'pixels' to represent them graphically. Each pixel has specific location and its individual values can be assigned to some range reflecting their various attributes. Basically, each location is represented as a cell. The matrix of cells organised by rows and columns is called a 'grid'. In the case of raster data, the connectivity is represented as adjacent cells or neighbourhoods of each cell, similar in manner to the nearest neighbour concept in spatial statistics and to the cellular Von Neumann neighbourhood in CA. There is thus no topology as such applied in the raster-based data model and in one sense, this makes working with rasters a little simpler than working with vectors. Both have their own advantages and disadvantages; hence to choose one, depends on the nature of the application and user needs (see more details of both data models in Demers 2005, Longley et al. 1999, Chrisman 2002, Burrough and McDonnell 1998). This section will now focus on the raster-based model as this is the main data structure used in this study.

Wagner (1997) compares 'GIS as CA' and 'CA as GIS', and this point is pertinent for two reasons. Firstly, the basic requirement for a CA can be provided by raster GIS in terms of two-dimensional space and

neighbourhood structure, for example. And secondly, CA transition rules can be formulated as GIS raster operators, e.g. focal statistics, for instance. From this point of view, it is certainly possible to exploit the entire GIS raster structure to implement a CA model and as a corollary, it is also possible to bring some raster operators to manipulate CA transition rules or to integrate them in constructing additional information such as land suitability constraints in constrained CA.

Li and Yeh (2000; 2001), for example, implemented a CA urban model using the GRID module in the ArcInfo platform package developing the coded script language called AML (Arc Macro Language) to control dynamic behaviour. Heuegger (2002) has built a CA extension based on the CA concept in a paper of White et al. (1997) for the Arcview package linking the Spatial Analyst module to a GRID analysis extension (see http://srf.tuwien.ac.at/lva/students/ca/index.html). Wagner (1997) has provided examples of CA simulations: the Game of Life and diffusion in the GIS and Image processing package IDRISI. To do so, it needs to apply two GIS operations including 'filtering' and 'reclassifying' to drive the simulation. Akin to some other modules in recent GIS software packages such as Spatial Modeller and ArcGlobe in ArcMap that broadly developed many prototypes that have now become standard, it is possible that GIS users may soon be able to use one of the standard CA modules in their usual GIS package.

In another case, raster-based GIS have a range of functions from the very simple to the very complex, and these can be extended to simulation models. Some CA work directly by using them to modify transition rules and there is some circuitous use of these functions in preparing or creating static or dynamic constrained factors. No matter whether a model has been created in a GIS environment or on other programming platforms, it can use these useful operators to build in constraints and functions which bring it much closer to reality. In the following sections, we represent these concepts and show some vital raster operations which can be attached to CA models.

Map Algebra and Cartographic Model

A well-known GIS operator applied to both the raster and vector data model is *map overlay*. However, it is completely different for raster or vector processing. In raster-based GIS, map algebra is an ordinary operation that every raster GIS packages encompasses, and these are referred to as map algebra and cartographic modelling (Tomlin 1990). Raster maps represent each attribute as separate thematic layers, capable of performing any mathematical operation which can be built from the most basic operations of addition, subtraction, and so on. This is similar to algebraic notation applied to grid data as on single numbers and the procedure for using map-algebraic techniques to build models for spatial analysis is called cartographic modelling (Burrough and McDonnell 1998).

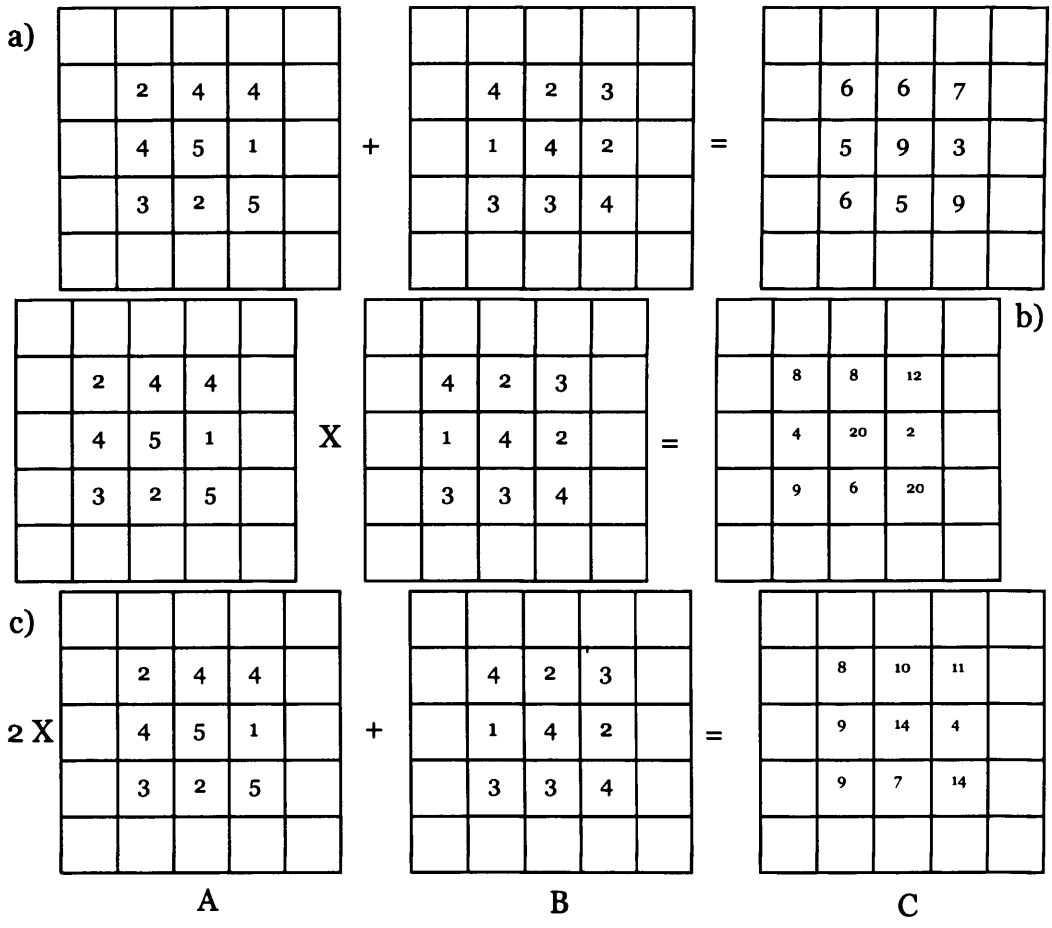


Figure 2.5 Basic Map Operations with Map Algebra

Figure 2.5 represents simple map overlay operations based on map algebraic notation and concepts. The operations shown in Figure 2.5 can be written as commands: [MAP C = MAP A + MAP B] for Figure 2.5a, [MAP C = MAP A x (multiplies by) MAP B] for Figure 2.5b and [MAP C = 2(MAP A) + MAP B] for Figure 2.5c. Cell by cell, the operations are performed on the input maps (MAP A and B) which create new raster maps (MAP C).

In addition to map operations shown in Figure 2.5, Tomlin (1990) has provided other map algebra operators which can be used to manipulate a variety of input data to construct new maps as shown in Table 2.1 above. These can be visualised in similar ways to those we show in Figure 2.5 above (see more details in Tomlin 1990, Demers 2005, Burrough and McDonnell 1998).

Local

The term 'local' in map algebra is different from 'local' in cellular automata. It is defined as a point or cell-by-cell procedure used to operate or compare each cell from map inputs. For instance, if applying 'localMaximum' and 'localMean' to MAP A and B as shown in Figure 2.6 below, the operators will return new maps as MAP C and MAP D representing the maximum number from MAP A and B (6 > 4) and an average value of both maps ([4+6]/2) respectively. As shown in Table 2.1, there are a variety of local operators, which can be used for several similar purposes, e.g. localMinimum, localDifference, localMajority and localMinority for example.

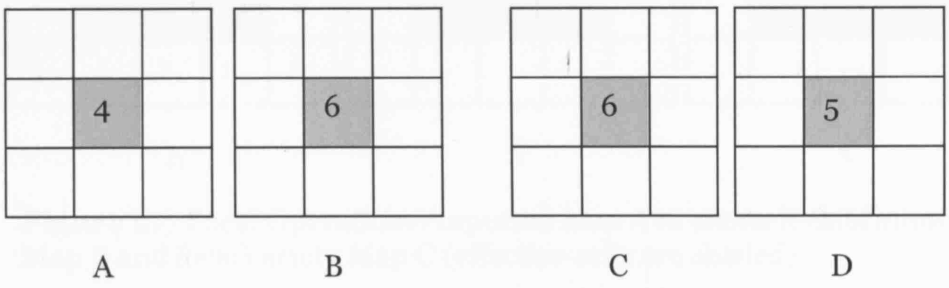


Figure 2.6 Local Operations: Input Maps A and B to create localMaximum Map C and localMean Map D

Focal

The focal operators are very close to the CA neighbourhood concept and also similar to filtering in an image processing context. The focal operators include the central cell and its neighbours (which can be the 8 cells in the Moore neighbourhood + the central cell or some action-at-a-distance). Once a focal operation is performed, a roving window will move along the grid and collect the information to calculate or compare and construct a new map as a result. Figure 2.7 demonstrates two focal operators applied: focalMinimum and focalVariety. For instance, the first shaded cell in MAP B for focalMinimum is defined from: Minimum [5,3,2,4,3,4,4,4,2] = 2. On the other hand, the first shaded cell in MAP C is counted from Variety [5,3,2,4,3,4,4,4,2] = 4 (2, 3, 4, 5). Other focal operators exist such as the focalMean (similar to the low pass filter), focalSum and so on. The focal operators have been used in several analyses; for instance, Fortner et al. (2004) used focalVariety operator to look for the unique occurrence of abandoned mine areas with different focal radii. Craighead et al. (2001) applied focalSum to calculate density of wildlife habitats, while Convery et al. (2003) employed the focalMajority command to build maps representing high forest cover and high development.

5	3	2	5	1
4	3	4	2	5
4	4	2	2	1
5	3	1	4	2
3	4	1	2	1

A

	2	2	1	
	1	1	1	
	1	1	1	

B

	4	4	4	
	5	4	4	
	5	4	3	

C

Figure 2.7 Focal Operations: Inputted Map A to create focalMinimum Map B and focalVariety Map C (effective cells are shaded)

Zonal

Zonal operators are used to create aggregated summaries as zone maps. These construct new maps from two input maps: a zone map and a value map. Take zonalMean for example: Figure 2.8 demonstrates the map constructed from this operation and can be written as command: MAP C = zonalMean[MAP A, MAP B]. For instance zone 1 of MAP A, Mean [5,3,2,4,3,4,4,4] ~ 4. Normally, these operators are used for summarising information from one layer to one another; for instance, the zonalMajority command was applied to summarise the modal potential vegetation type in each polygon delineated by textual analysis of the TM imagery (see Keane et al. 2000). Möller et al. (2004) create a summation map from biomass summarised for each instance of travel cost for use in analysing the costs and supply for sustainable energy production.

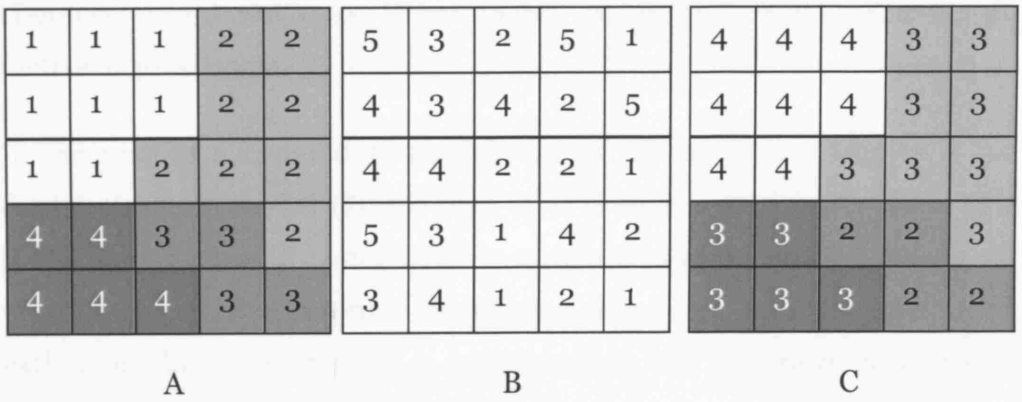


Figure 2.8 Zonal Operations: Inputted Zone Map A and Value Map B to create zocalMean Map C

These operations are vital; they can be used in many applications in different ways for raster-based spatial analysis. They can also be used for making final map products and they can be performed during processes of cartographic modelling. As they are cell-based operators similar to procedures in CA, they can then be easily applied to cellular modelling as this study has attempted to demonstrate. Various map algebraic commands are used in order to create maps such as dasymetric population density maps, and urban land demand maps, which are part of the

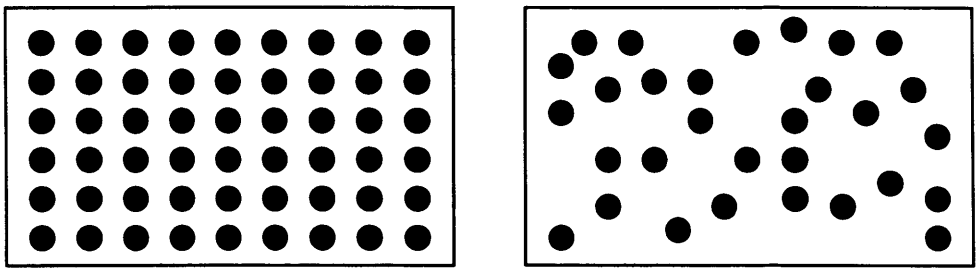
dynamic spatial information used in the model. We will detail these applications in Chapters 3 and 4.

Statistical Raster Surface

> "Continuous surfaces are usually represented by images or lines. Image methods include regular and irregular grids and tessellations in which the variations of the values of the mapped attribute is indicated by graded zones of colour or grey levels. Line representation includes isolines (lines of equal value), vertical slice (profiles), and critical lines such as ridges, stream courses." (Burrourgh and McDonnell 1998)

From the statement above, 'contour lines' which represent isolines of equal elevations (often called 'Z' values) on a space, are not necessarily elevational values only. Instead, for any measurable values in ordinal, interval and ratio data scales, the values can be thought of as Z values and the term generally used for such representations is 'statistical surface' (Demers 2005) such as those based on isobars and isotherms in meteorological maps.

In the case of a raster data model, one of the obvious examples is the digital elevation model (DEM) data as it is represented in the form of a continuous surface. A raster DEM is constructed as a grid or is cell-based where each cell is represented as only one elevation Z value. As mentioned earlier, in the same fashion, a raster surface can be drawn in order to represent other measurable Z values. To prepare a raster surface, one normally needs a set of Z value data which is basically provided in two forms — on a regular and an irregular lattice (see Figure 2.9). For the regular lattice, if it is small enough to fit a predefined grid cell size, then each lattice value can be directly converted to a grid value. However, available data are not usually directly usable as a raster surface for the data are often provided in the form of an irregular lattice. To generate a raster surface, it is necessary to estimate or predict all missing values on the grid space.



Regular Lattice Irregular Lattice

Figure 2.9 Two Forms of Lattice

To estimate and resample Z values in a grid space, there are several well-known methods from simple computation using distance values to more complicated methods using spatial statistics, such as Inverse Distance Weights (IDW), Spline functions, Nearest Neighbour statistics and Kriging (for details of the IDW methods and others used in Chapters 3 and 4, see the GIS textbooks by Demers 2005, and Burrough and McDonnell 1998).

As raster surfaces represent some form of continuous value in a space, hybrid CA modellers can take advantage of this when it is necessary to transform a set of data from a regular or irregular lattice to the same CA grid space. Then it can easily be combined with other spatial data and conveniently operated upon with CA procedures. Raster surfaces can be created from any such quantitative information. The SLOPE surface (created from a DEM surface using some commands from map algebra) as a constrained layer appears in several hybrid CAs (Clarke et al. 1997). Surfaces can also be in the form of density potential (Batty 1998) or infrastructure attractiveness (Loibl and Toetzer 2003) or population surfaces (Wu and Martin 2002). In most hybrid CA models, constrained factors are combined and can then be converted to probability surfaces that are finally used as decision rules for cell generation.

In this study, raster surfaces were used for constructing interaction between each node within a space that can be dynamically created, based on generated cells which integrate with the spatial interaction concept. These will be conceptually discussed later in this Chapter and in Chapters 3 and 4.

Spatio-Temporal Concepts

Space and time as considered within GIS relate to four traditional disciplinary traditions: philosophy, mathematics, physics and geography with two new perspectives making some impact based on cognitive and socio-cultural theories. Geographic information systems and science have been slow in adopting these developments (see details in Couclelis 1999) but there has been massive progress in developing the spatial perspective with a consequent neglect of temporal aspects. On the spatial side, the GI sciences currently use the full range of spatial representation and analysis, which have mostly been inherited from a basis in physics and mathematics, adopting those conventions concerning the ontology of space. The temporal perspective, on the other hand, has been quite deficient with few links to developments in spatial ontologies. To address some of these problems, Couclelis (1999) argued the need to extend GIS to cover space and time in four ways: to enable the integration of space and time, to represent relative and non-metric spaces (and times), to represent inexact spaces (and times) and to represent commonsense views of space and time. These are fundamental to any conceptual frameworks for advanced applications that relate to space and time.

Geographic phenomena are dynamic in nature. Basically, when we start to analyse even a simple spatial phenomenon, we are immersed in incidents and events that embody space and time. Building models in the spatial sciences has hitherto been barely related to self-conscious approaches to 'time' and 'space' that are key to newly emergent spatio-temporal concepts. Practically, the spatio-temporal approach suggests that Temporal GIS (TGIS) should be a practical activity. TGIS is an attempt to store and analyse special objects and changes in their attributes through time (Castagneri 1998). Temporal information are integrated into GIS spatial data models in the form of time-stamping layers (see Armstrong's snapshot models [1988]), attribute (space-time composites modelled by Langran and Chrisman [1988]), and spatial objects (Worboy's spatio-temporal objects [1992]). Other approaches involve the representation of spatio-temporal information based on events, i.e. the temporal sequence

collection (STC) by Segev and Shoshani (1993), time objects (OODAPLEX by Wuu and Dayal 1992), the event-based spatio-temporal data model (ESTDM by Peuquet and Duan 1995) and the geomorphologic spatial model (Oogeomorph by Raper and Livingstone 1995) (see details of these spatio-temporal models in Yuan 1995).

Most spatial modellers are familiar with static spatial phenomena that often form the bedrock of most GIS applications, and these become complicated and often problematic when manipulated into dynamics by injecting an explicit time dimension into them. An advanced spatio-temporal classification suggested by Yuan (1995) consists of six major types of spatial and temporal change in geographic information as summarised in Table 2.2 below.

Table 2.2 Yuan's 6 Major Types of Spatial and Temporal Changes (Yuan 1995)

Type	Any given	Fixing	Controlling	Measuring
I	Site	Locations	Attribute	Time
II	Point in time	Time	Attribute	Locations
III	Period of time	Time	Locations	Attribute
IV	Event	Attribute	Locations	Time
V	Area	Locations	Time	Attribute
VI	Event	Attribute	Time	Locations

From the table, Type I changes involve measurement of the duration (time) of events or attributes that change from time to time in any given location; there is no change in spatial properties. Type II changes describe the static spatial distribution of geographic phenomena at any point in time such as topographic map and isobar contour map. Type III and IV are called spatial changes, which can be classified into 'static' and 'transitional' respectively. In any given period of time, Type III changes involve the representation of variations of attributes that may change from site to site over a period of time, i.e. land use changes for example. Type IV changes measure duration of an event that occurred varyingly over different areas.

For instance, measuring how long a monsoon will pass from site A through B. Type V changes are called 'mutation' that represent how attributes of an area will change over a control time. And Type VI changes are called 'movement' that concern changes in location of any given event over control duration, i.e. the movement of wildfire spread over a forest area, for example.

This broad concept concerning spatio-temporal circumstances suggested by Yuan is useful in order to classify how profound are the space-time interactions that modellers need to build or integrate into models, what types of situation modellers will use to analyse space-time data, or which types can be combined to make models more realistic. According to the six types above, we may say mostly that CA and agent-based models are similar to the last two types, mutation and movement, since the models are controlled by 'time' and varied by attributes and/or locations.

Types of time as we are familiar with in reality can be calendar time, seasonal time, and duration measured as days, hours, minutes or seconds. Particularly when we attempt to investigate a situation, we always think about duration or the period of occurrence or we even use time incidents as a factor for analysis, i.e. satellite image classification, for instance. Comparing this to real world time, most of CA and agent-based applications are implemented based on linear interval time and outcomes are mostly represented as snapshots or time slices; for instance, iterations in hypothetical urban models, synchronous real time and iteration in hybrid urban models (see the CA section above), durations as minutes in pedestrian movement models, or weekly duration for epidemiological models. However depending upon duration of the phenomena and precision of the time scale, such models can be defined using cyclical time as well. If modelling epidemic applications in the case of some diseases of longer duration, it is possible to relate this to cyclical seasonal time (see Egenhofer and Golledge 1998).

4. Geographical Concepts in Urban-Regional Development

Returning back to the big picture of the model to be implemented in this study, the two components that are technical practices already discussed in previous sections, based on CA and raster-based GIS, will be further explained conceptually and practically in various aspects of their implementation in later chapters. Other important pieces of the jigsaw puzzle are the urban and regional concepts and theories that underlie the technical parts to the model. These will be elaborated in this section which will deal with geographic concepts of urban and regional development based on nodes, interactions, hierarchies and diffusion.

The Nodal Region

One of the main ideas in the model created in this study derives originally from Haggett's (1965) work on the nodal region where he uses this concept to represent broad concepts of the locational and areal arrangement of human settlements. In this way, these notions and their evolution into structures which are called the nodal region are used to integrate the DSSM model into ideas about the structure and organization of urban morphology. Stages in the analysis of nodal regional systems include interaction, networks, nodes or centres, hierarchies and diffusion which we illustrate rather generally in Figure 2.10.

The stages illustrated in Figure 2.10 cannot strictly be seen as a sequential progression from the start to finish, for they can appear both sequentially and/or simultaneously. To apply these ideas about nodal regional structure in this study, it is necessary to generalise them in embedding them into an operational model which is tuned to real world data whose dynamics are based on the four key elements of interactions, nodes, hierarchies, and diffusion.

Surface elements were noted in the previous chapter, and in previous sections, and in the case of nodal regional systems, ideas about gradients in population densities, population fields, and attributes related to

interactions within and between cities related to the spatial organisation of land use are central. Network elements are generally very important for the development of nodal regional systems but applying these elements to predictive simulation models can be difficult in that these do not completely accord and are sometimes dissimilar to real world transportation networks that represent the physical structure of urban and regional systems. The other limitation is that network elements are complex vector representations of data, difficult to integrate in models such as DSSM which are essentially raster or cellular based and do not have an explicit network representation.

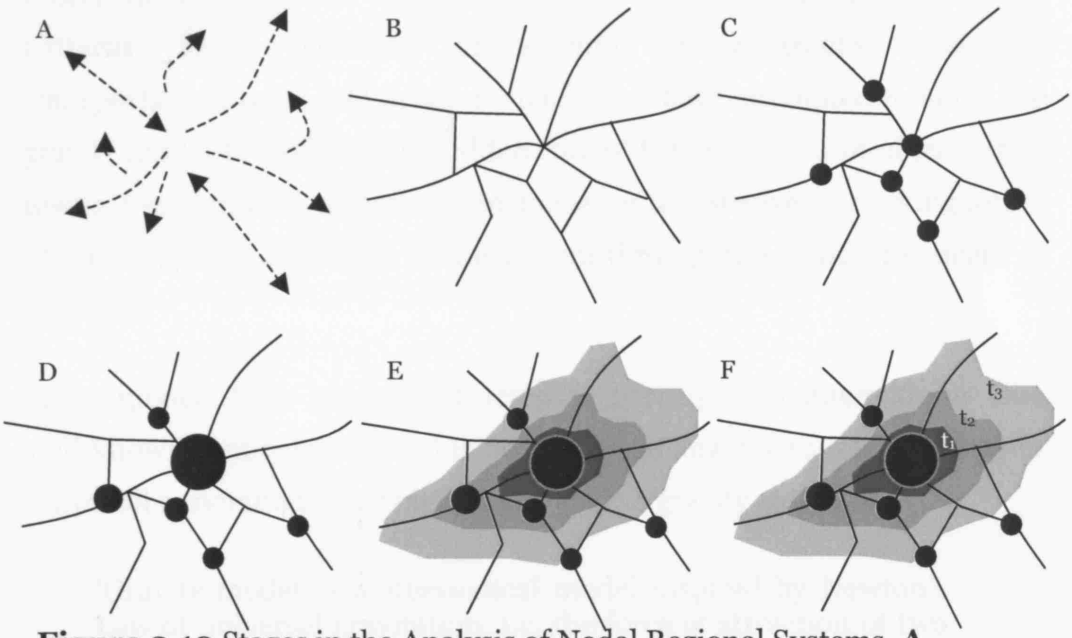


Figure 2.10 Stages in the Analysis of Nodal Regional Systems. **A** Interaction; **B** Networks; **C** Nodes; **D** Hierarchies; **E** Surfaces; **F** Diffusion. (Modified from Haggett et al. 1977)

As this study proposes to build a model to simulate and visually explore real cases of urban (local) interaction with their associated hinterlands (regional), the network component is identified with existing transportation in order to reduce the time for model development but still enabling us to represent growth in real transportation structures. The review here will not get into the detail of these network elements; all the basic and advanced ideas which mostly concern linkages and shortest paths from the simplest two or three point centres to more complex

polycentric networks varying from the local to the regional scale, are available in several textbooks in human geographical analysis and transportation planning (Haggett et al. 1977, Laura 1999). Network analysis in GIS is also explored in a range of GIS textbooks (see Demers 2005).

Interaction

It is not only physical science that is able to utilise the gravity model concept for movement and forces at different physical scales up to the universe itself, but this is also a concept that has been widely used to evolve describe human activities and location associated with spatial patterns. Many scholars, for example in geography, sociology, transportation, retail and urban management have attempted to apply the gravity model to explain the 'hidden forces' between pairs of objects (two towns, two shopping centres, for instance) or to visualize interacting forces of one object on all others in the system through the physical concept of potential.

Many models have been constructed to portray such interactions. One well-known elementary model is postulated in analogy to Newton's law of universal gravitation, and this is the so-called 'gravity model'.

> 'Gravity model: a mathematical model inspired by Newton's Law of universal gravitation, i.e. the force of attraction of two masses, or bodies, is proportional to products of their masses and inversely proportional to the square of the distance between them' (Haynes and Fotheringham 1984).

The model can be specified generically as:

$$F = \frac{G M_1 M_2}{d_{12}^2} \qquad (2.1)$$

where
F: Force between each mass
G: Universal constant
M_1 and M_2: Sizes of the two masses
d_{12}: Distance between M_1 and M_2

The gravity model clearly embraces three vital components including 'mass', 'distance' and the 'constant or scale' parameter, often referred to as

the gravitational constant in physics or a scaling constant in the human sciences (see equation 2.1) (Haynes and Fotheringham 1984, Huff and Jenks 1968). Haynes and Fotheringham (1984) argue that in the development of gravity models, there are three stages: modification, evolution, and expansion and generalisation. These modifications are associated with the three fundamental elements as mentioned earlier. In the spatial sciences, distance is normally defined according to the Euclidean distance measured between two masses distributed over a space. However, distance can also be defined as distance along a transportation route or travel time taken to make a trip from one location to another. Mass can be defined in many ways but must be comparable in a quantitative way, based on measures such as population, number of retail stores, etc. The constant or scale is a value that is used to adjust the quantities to reflect the real units characteristic of the way the phenomena is measured. Three modifications can be made on these elements and exponents associated with weight for the masses and the friction of distance (see equation 2.2). Such modifications lead to

$$T_{ij} = k P_i^{\lambda} P_j^{\alpha} d_{ij}^{-\beta} \qquad (2.2)$$

where
k: Constant
P_i P_j: Masses
d_{ij}: Distance
λ, α, β: Exponent values

The second stage of development of these models involves evolution. Flows and movements among two places are natural expressions of human activities which connect populations in the space of daily life. This was the impetus for social scientists to derive the gravity model for simulating and visualizing interaction in the space, and many applications developed after the Newtonian law became known within the social sciences in the late 19th century. In the 1920s and 1930s, gravity models of spatial interaction were applied to retail trade area studies. Reilly implemented his well-known model (called the Reilly-Converse model) based on his 'Law of Retail Gravitation' in 1929. Reilly's law manipulated the gravity model so that the analyst could find the sphere of influence associated with each pair of

centres (equation 2.3), and thereby determine the breakpoint or boundary between them. Other more advanced models emerged later, for example, Huff's model of consumer behaviour (1959), Dodd's interaction hypothesis model (1967), Zipf's principle of minimum effort model (1949), Stouffer's intervening opportunities model (1940) and Wilson's entropy model (1967) (see details in Haynes and Fotheringham 1984, Haggett et al. 1977, Huff and Lutz 1979). To give an idea of these developments, Reilly's model sought to compute the breakpoint between two centres as

$$D_1 = \frac{D_{12}}{1+\sqrt{\frac{M_2}{M_1}}} \qquad (2.3)$$

where
D: Distance from mass 1 to the breakpoint
D_{12}: Distance between two masses
M: Masses

The third stage relates to the expansion and generalisation of gravity models. According to Haynes and Fotheringham, four developments have been attempted to improve the model. First there is the development of potential measures. This expansion generalises the model from a flow between a single pair of centres to a set of flows among all centres in the system. Instead of a pair of centres being compared for observing the flow, the total flow estimate in a system of centres is derived by summing the gravity model estimates of all possible pairs. Potential as defined in physics and generalised as potential surface analysis (PSA) is an outcome of such development (see equation 2.4). Potential

$$P_i = \sum_{j=1}^{n} \frac{M_j}{D_{ij}} \qquad (2.4)$$

where
P_i: Potential of centre i
M_j: Mass of centre j
D_{ij}: Distance between centre i and j
n: Number of centre j

The second development involves the push- and pull-factors that are associated with the distinction between origins and destinations and involves defining appropriate measures to represent the importance of

these variables. In applications to urban examples, pull and push factors are usually interactive forces referred to as centripetal and centrifugal forces. Colby was one of the first to explain the detail of how both two forces applied to urban systems and produced considerable evidence from many sources for these patterns; these are summarised below (see Colby 1933).

Centrifugal Forces

The centrifugal forces cause decentralisation and urban sprawl as they tend to force dwellings and businesses away from congested and expensive inner city areas towards the suburbs. Colby suggested that there are two such forces which cause the migration of functions from central zones involving both the 'uprooting' tendencies of central zones and the attractive qualities of outer zones. For example in central zones, land and property values, traffic congestion and high cost of transportation, difficulty of securing enough space at a location, noxious manufacturing uses, the impossibility of acquiring appropriate sites in congested central zones, and irksome legal restrictions, out-of-date laws etc., are all issues that affect the movement of activities to the suburbs. The attractions of peripheral zones can be summarized in a similar but opposite manner: low cost large parcels of unoccupied land, the presence of suitable transport services, attractive site qualities, the control of sizable areas, and new, modern buildings are all features that attract housing and employment from the centre to the suburbs.

Colby finally summarised these centrifugal factors as six individual forces including: 1) the spatial force— from the congestion of central zones and vacancy of space in the outer zones; 2) the site force—from the intensive use and modification of natural landscape in the central zones to the relatively less dense occupation of the outer zones; 3) the situational force—based on satisfactory functions outside the central zone; 4) the force of social evaluation—from different pressures of land values, taxes, and legal restrictions between two zones; 5) the status and organisation of occupancy—where obsolete functional-forms exist inside the central zone;

and 6) the human equation—such as religious tenets, personal whims, and speculation from the anticipation of real estate booms.

Centripetal Forces

Centripetal forces cause centralization, attracting establishments to the central area where they may benefit from the advantages of accessibility and agglomeration economies. Colby classified these attraction qualities into five types: 1) site attraction—some features of the natural landscape within the urban complex: the River Thames in London, for example; 2) functional convenience at three levels: metropolitan (local), regional and inter-regional scales, where the central zone is the focal point for wholesale and retail trade, regional and inter-regional transportation, import-export of goods, for example; 3) functional magnetism— the attraction of concentrating one function in a central zone which leads to linkages to other functions, particularly in services and manufacturing industries; 4) functional prestige—almost every big city carries a greater or less degree of prestige: Fifth Avenue in New York, the Rue de la Paix in Paris as the street of centre of the fashion world, for instance; and 5) the human equation once again— different desires and preferences for pulling people back into the central zone: the attractions of urban living.

These two forces have been cited in many works associated with urban-regional spatial interaction (for recent examples see Gelan 2003, Anas et al. 1997, Brañas et al. 1999, Krugman 1998, Batty et al. 2003). However, many of these do not articulate these forces in quantitative terms and thus, they are often used to explain the evolution of the urban system as a balance of pull and push factors but in qualitative ways. But they do relate to spatial interaction, and it is possible to incorporate these ideas into gravitational and potential models so that an operational focus can be provided when these factors are integrated into models.

The third development concerns the separation of origins from destinations and the impact of this on intervening distance or space. This focuses on the presence of alternative destinations that intervene between

any pair of centres. Intervening opportunities reduce the flow from the place of origin to the place of destination by absorbing or diverting interactions to closer destinations. Such issues can then be measured in terms of the accumulation of intervening opportunities which then can be used to modify the distance functions within such models.

The fourth development concerns the spatial pattern of origins and destinations. Such patterns are indicated as spatial structures and intimately affect the spatial interaction pattern. These can be incorporated into models through existing parameters, but these are not usually included in gravity model formulations. However accessibility and related criteria have been included in such models in the form of competing destinations as articulated by Fotheringham (1983).

Nodes and Hierarchies

In the nodal region concept shown in Figure 2.10, nodes are used to locate the junctions on a network. Based on the scale of analysis, nodes can however serve as a 'collective' term, and can be used to represent other **space** and **size** variables such as cities, centres of a city, villages, population clusters, all ranging in scale from the local to the regional level. Nodes can spatially represent the existence of settlements or functions as lattice in a space. Multiple nodes are used to define patterns that can be basically classified into three types: regular, random and clustered where in a space, these three patterns convey different spatial meanings. Practically, these can be distinguished quantitatively using statistical methods such as quadrat analysis (Dacey 1964), spectral analysis (Rayner and Golledge 1972) and nearest neighbour analysis (King 1962) (for details see Demers 2005).

Actual settlement arrangements are difficult to describe using theoretical concepts such as a regular lattice. Practically, distributions of settlements are constrained by other factors which distort settlement morphologies and interactions over *space* and *time*. Haggett et al. (1977) have explained and provided some interesting evidence and models of both deformations

in their book, where they note the work by Bogue (1950) on distortion by agglomeration, settlement dynamics over time under different situations (Bylund 1960), and Monte Carlo simulation methods from Morrill (1962).

In addition to distributing nodes in space, their size is an important differentiator of each node or centre in the settlement spaces. Size can be measured by many attributes of centres such as the number of business firms and services or diversity in trading. One of the simplest but most efficient attributes at the regional scale is *population*, which is an immediate characterisation of abundance and diversity in each centre (Skinner and Handerson 1999). Population is also used to measure spatial distribution at the regional level as in the well-known rank-size rule by Zipf (1949), which can be related to the expressed interaction between actual populations of each centre and their predefined ranks in the central place hierarchy (Berry 1967).

The size and distribution of centres or nodes are used in measuring and classifying settlements into different orders or hierarchies. Most research in measuring hierarchies begins by using scatter diagrams which compare settlement functions and sizes of settlement. For example, Berry (1967 p.27-28) illustrated scatter diagrams of the logarithm of trade area size against the logarithm of population. Statistically, settlement hierarchies can be derived using cluster analysis as Huff and Lutz (1979) measured the urban system in Ireland using factor scores, which had been chosen by extracting the main factors from a series of relevant spatial variables using factor analysis. They classified all 114 urban centres into five levels: to one urban centre at the first level, three centres at the second, ten centres at the third level and so on (see Huff and Lutz 1979).

Sizes and hierarchies can also be measured and classified using 'centrality' concepts which originate from central place theory (Christaller 1966) and from the 'nodality' concept which considers settlement functions and services. Preston (1971) called centrality *relative importance* and nodality *absolute importance*, where centrality is nodality that excludes local consumption as shown in equation 2.5:

$$C_i = N_i - L_i \qquad (2.5)$$

where
C_i : centrality of place i
N_i : nodality
L_i : local consumption

and an operable formula is based on equation 2.6 below

$$C_i = R_i + S_i - \alpha M_i F_i \qquad (2.6)$$

where
C_i : centrality of place i
R_i : total sales of retail establishments
S_i : total sales in selected service establishment
α : average percentage of median family income spent on retail items and selected services
M_i : median family income for central place
F_i : total number of families in central place

Preston applied this concept to analyse centrality in the Pacific Northwest region by collecting data from the Census, complementing this with raw data from communities with 2,500 or more inhabitants in 1960. Results clearly illustrated absolute and relative importance of all centres. And finally, these quantities of centrality were transformed into measures representing 5 orders of central place hierarchy (see Preston 1971).

The size and hierarchy of each centre also corresponds to the spatial influence reflected at all levels of **spatial interaction** over all centres. Higher order settlements represent broader spheres of influence as obviously shown by Huff and Lutz (1979). They mapped the sphere of influence using the lines of equilibrium technique (Huff and Lutz 1979 cited Gambini 1966) which they applied to all centres in each hierarchy. The results visibly demonstrate influential zones according to levels in the urban hierarchy and these were ultimately used in this study to support governmental regional planning. Furthermore, these concepts of centrality and hierarchy offer practical spatial information useful for other consequential extensive settlement models and related applications: the core-periphery model (Skinner and Henderson 1999, Henderson et al. 1999), spatial competition and market centre models (Berry 1967) and various monocentric and polycentric models (Johansson 2002), for example.

Diffusion

Diffusion in the sense of the nodal region is related to dispersion from a centre over a period of time. Diffused phenomena can be many different movements at different levels and geographic scales, such as the dispersion of goods from a city centre to other towns, the transmission of innovative information from a country to others, and even the outbreak of disease from a room to others within a building. Haggett (1975) has classified the diffusion process in three main ways – expansion, relocation and combined expansion and relocation processes, while extending one process into another defines a cascade or hierarchical diffusion (Figure 2.11). These correspond to the spatio-temporal concepts, movement and mutation which we discussed earlier in Yuan's (1995) work and Figure 2.11 illustrates how such ideas can be related to earlier ones in spatial and temporal ontologies and dynamics which originate in GI science.

One of the most prominent early works in diffusion processes is the agricultural innovation-diffusion process originated by Torsten Hägerstrand and colleagues in 1953. According to Hägerstrand's model, there are six key elements: *area* or environment; discrete *time* at several scales—minutes, hours, days; *items*—material (people, telephone) or non-material (behaviour, message, disease); different places of *origin*; *destination*; and *path* (Haggett 1975). Additionally, processes of diffusion also depend on the friction of space or barriers which are reflected in topography and accessibility, which Haggett also classified into several classes such as absorbing, reflecting and so on. Following Hagerstrand, he suggested a diffusion model based on the probability of contact that was generalised spatially into the Mean Information Field (MIF). Details of simulations using Hägerstrand's diffusion model will be described later in our conceptual and methodology chapters but the contexts for these diffusion studies are given in Haggett (1975) and Haggett et al. (1977).

Since the 1950s, the idea of the spatial diffusion of innovations developed by Hägerstrand has inspired many scholars to implement their researches by improving the diffusion technique, integrating it with spatial analysis

and simulation models so that many more fields of application might be explored spatially. For instance, the study of dynamic processes of dengue hemorrhagic fever epidemics (Muttitanon et al. 2002), diffusion models in sociology (Palloni 1998), demographic processes (Omer 1999), the diffusion of financial business activities (Akhavein et al. 2001), or even the diffusion of GIS adoptions (Masser and Craglia 1996) are all significant examples. Of course, diffusion as in cellular automata processes is central to this thesis and this indicates how we might connect up these seemingly disparate fields of endeavour (Wagner 1997).

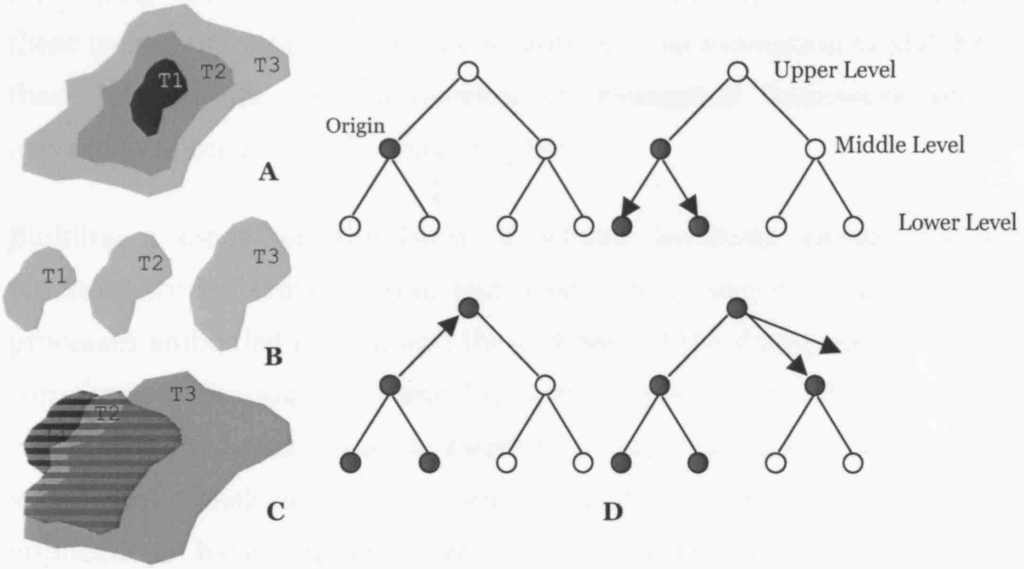


Figure 2.11 Types of Diffusion. **A** Expansion; **B** Relocation; **C** Combined Expansion and Relocation; **D** Cascade or Hierarchical Diffusion. (Modified from Haggett 1965)

In a dynamic sense, the stages in the development of spatial modelling mentioned above represent both the causes and the effects embodied in many developments in the fields of geographic information science, locational geography, urban and regional economics, and regional science. *Spatial interactions* are built around ideas of mass and distance, which certainly can be obtained from information about the distribution of *hierarchical nodes* or *centres* (size and space). Spatial interactions can be mapped in the form of continuous *surfaces*, which also affects directions and intensities of *diffusion* movement. Flows from the diffusion process are channelled into discrete *networks* which link together the individual

settlements through the *nodes*. In the ensuing development of DSSM, we will show how all these ideas fit together.

5. Implementation of the Dynamic Settlement Simulation Model (DSSM)

Thus far, all the conceptual components of the model in this study have been assembled and reviewed at least in general terms. It is time to join them together to find out what the model will look like and how it can be used to engage in urban simulation in rapidly growing urban contexts in developing countries. This section discusses crucially how to integrate these techniques and theoretical concepts into the simulation model. From these integrations, we will develop the conceptual framework and its implementation in the following chapters.

Building a model on the basis of cellular automata already has full functionality in terms of time and space which accord to the standard processes embodied in CA, and these represent the dimensions of spatial complexity (Xie 1994, Batty and Xie 1997). However, spatial and temporal complexity in standard CA, as mentioned earlier, is based on predefined states and transition rules in which implementations of standard CA applications have neglected other crucial static and dynamic spatial information. It is this information that the theories alluded to in the section above relate to.

Probabilistic and constrained CA applications, on the other hand, have been developed to fill this gap. Many CA researchers developing urban applications since the 1990s have implemented increasingly more and more factors which are both spatial and non-spatial from local to global levels (local cells to regional and thence to the whole space). These factors cover many aspects such as accessibility, topographic features, and definable policies, e.g. sustainable development, land suitability, and protected or restricted areas (Li and Yeh 2000; 2001, Yeh and Li 2001, White and Engelen 2000, Wu and Martin 2002, Clarke et al. 1997, Ward et al. 2000, Loibl and Toetzer 2003). To integrate these factors into a CA model, it is necessary to utilise one or more functions of GIS, at least, to

collect or prepare spatial data using GIS, since functions in GIS are able to support such integration easily and efficiently. Moreover much of the data now comes in a form that requires a GIS to unlock it.

Even though developing stochastic formulations and adding constraints using some GIS operations within models are quite well developed, it is still important to measure how close (or far) a CA simulation is from complex real world phenomena. Some essential questions spring from this context namely:

- *how have recent works applied the capability and functionality of GIS operators in manipulating spatial and non-spatial data to create new and more useful information within the processes of CA modelling?*
- *are there changeable constraints within recent CA applications which have been reflected in the dynamic aspects of CA?*
- and *are there any model components or modules developed based on dynamic spatial analysis which have been used to manage static and dynamic spatial information within such models?*

Many works in urban CA simulation have focused on urbanisation of a city or on major urbanised areas or have used fictional cities depicted using a wide range of land use classifications from binary (urban or non-urban) to multilevel developments which contain a complex array of urban land uses (Xie 1994, Batty and Xie 1994;1997 for example). Some works have included a regional dimension by applying the simulation to wider study areas within higher resolution spaces, which can represent regional growth simultaneously along with local growth which is simulated at the cell level (Li and Yeh 2000, White and Engelen 2000 for instances). However, many of these studies have overlooked the interactions between these different levels and thus the importance of how central a town is compared to the others in the hierarchy has been neglected. This is where the notion of *nodes and hierarchies* comes into its own; how interrelated such towns are within the same regional space is a function of *spatial interaction*, and how these interactions evolve dynamically impacting on the growth of

cities and towns through *diffusion*. These will be developed within DSSM using various developments of the nodal region concept, which concentrate on urban and regional interaction as mentioned in the previous section. The DSSM has been developed to respond to these questions mentioned above and will focus on integrating the techniques of CA and GIS to the key element of the nodal regional system. The conceptual framework for implementing the DSSM derived from these techniques and theories will be discussed in more detail in Chapter 3.

In terms of computer programming, there are two alternatives with different advantages and disadvantages with respect to implementing the model: 1) integrating CA components into GIS packages and manipulating time and space using script languages within the packages, and 2) implementing the CA model in a standalone programming language and including only necessary GIS operations into the model. Advantages of the first option relate to GIS capabilities such as data input-output, data management and most importantly the variety of GIS functions that can be used to thematically create constrained layers within the model. Disadvantages of using GIS as a model platform concern more detailed programming: script languages are not very flexible in using GIS functions, and script languages certainly take longer to compile in terms of their code compared to more general languages. In short, script languages are not suitable for compiling complicated models that relate to huge data input and outputs.

The second choice – developing standalone programs – is very powerful in program compilation and this makes running entire processes within the model much faster. Some functions that need to be refined can be more easily adjusted. Products created in a programming language can be flexibly constructed into a single package that can be easily exported and distributed; it is not necessary to distribute the package with reference to any other third party GIS package. On the other hand, the disadvantages rest in the inconvenient use of GIS functions for all these need to be implemented in the model. Alternatively, it may be possible to use external GIS libraries to extend the program, and these are becoming more widely

available as APIs (Application Program Interfaces) that is a set of routines, protocols and tools used for building software applications.

This project has been developed using an object-orientated programming language, JAVA, which is one of the most powerful and easy to distribute but one which is not so difficult to code, and thence extend. All necessary spatial analysis functions such as map operators, zonal statistics, cartographic functions and dasymetric mapping have been included, embedding their methods and algorithms in the code associated with the main program. The concept of object oriented programming and details of how we transform the conceptual framework into such a program in practice will be described in Chapter 4. In the next chapter, we will begin to elaborate the conceptual framework introduced in this chapter and translate this into the model structure before we program the model in Chapter 4.

CHAPTER THREE
The Conceptual Framework and Model Structure

1. Introduction

This chapter takes forward the concepts behind the model which we introduced and reviewed in the last chapter and shows how they can be integrated practically in order to implement the model. The dynamic settlement simulation model (DSSM) fuses many techniques such as cellular automata, GIS-based spatial analysis, and event-based simulation modelling. All these techniques have their own advantages in ways they approach the analysis and synthesis of spatial information, and many of these will be exploited in designing the model structure. Theoretically, this model has been constructed so that processes of urbanisation at the urban-regional scale can be explored. Apart from the need to develop the model across these scales so that various processes of interaction discussed in the last chapter can be implemented, changing the model scale enables us to envisage the city and its regions in a different dimensional perspective, letting us widen our scope and setting urban and regional development in context. Interaction between urban features is of course important in generating this wider perspective but other factors from the outside world – from the system's environment – should not to be ignored for in an increasingly global world, interactions at all levels are significant.

In this chapter, we will first argue that the conceptual framework provides us with details about the techniques used in this study which focus on model dynamics. We then deal with these techniques in turn showing how cellular automata, diffusion, node and hierarchy generation, spatial interaction, and the various urban forces driving centripetal and centrifugal growth and change, are integrated into the model structure. This will take us to the point where we can derive a generalised sequence for the various model operations which we will assemble in flow chart

form, preparing us for the detailed programming developments that we will describe in Chapter 4.

2. The Conceptual Framework

In integrating the CA model structure with the Dynamic Spatial Model (DSM) which is focused on simulating urban and suburban growth, the various concepts of spatial organization – *interaction, node, hierarchy* and *diffusion*, introduced in the last chapter will be dealt with in turn in designing a model of urban and regional growth phenomena. This section will explain conceptually the framework of this study in preparation for more technical issues in later sections.

Figure 3.1 illustrates various flows of activity in the model where it is clear that the structure is built around the dynamic spatial model (DSM) with the CA framework being used to control diffusion which is a central part of the way urban and regional growth is to be handled. The dynamic spatial model is the main module which creates active thematic map layers. Traditional GIS normally has little ability to process spatial data in a dynamic sense, and thus the DSM contains the various operators for manipulating diverse GIS functions which analyse and synthesise dynamic spatial data which drive the spatio-temporal structure of the model. Spatial variables concerning urban and regional growth are defined as the key data sources for DSM. All spatial and non-spatial data is to be classified and analysed by several GIS functions at every time slice and in every time period. Cartographic modelling is used to synthesize all the various map layers which are modified at each time instant or over each time period. Outputs from the DSM represent important local constraints on the operation of the CA and diffusion models in successive stages. Outcomes from this process are also represented as map layers as well as statistical information which are essential for visualisation and quantitative measurement of the performance of the model as well as its predictions. The way these functions are put together is illustrated in Figure 3.1.

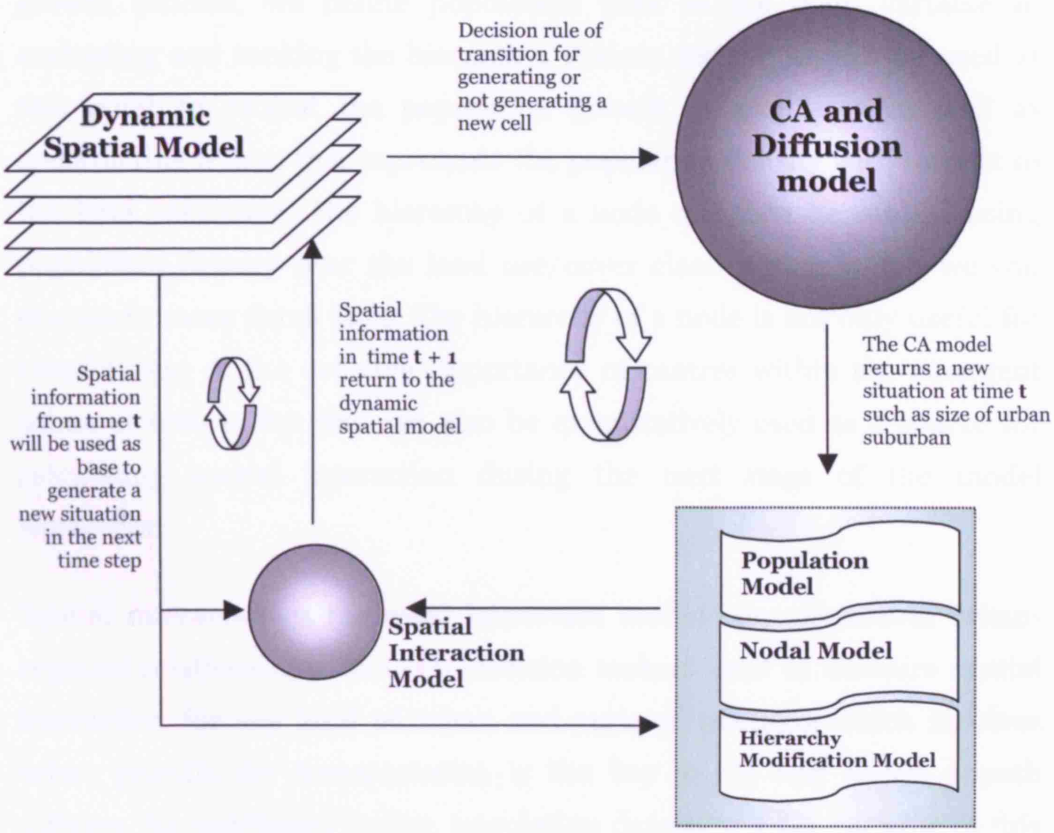


Figure 3.1: Conceptual Framework of the Model

The CA model enables **'cells'** to be generated from **transition rules** that use the values or attributes of **neighbouring cells**; cells are associated with 'births', 'survivorship', 'decline' and 'death' during a simulation **state**. The diffusion model enables random seeds to be generated within the cellular space where the processes of transition are based on dynamic probabilities.

The next stage focuses on spatial organization based on the concepts of node, hierarchy and spatial interaction. First we need to measure and define nodes and hierarchies. In this study, a node is assumed to provide the 'centrality' of an urban area, and they can be generated using an algorithm which searches for centres within any given search radius. Furthermore, an existing node can be replaced by a new location that is more 'central' than the old one. To define the order or hierarchy of nodes, one or more variables that are able to separate nodes from each other need to be considered. As population is a major driving force of the urban

growth process, we define population data as the main variable in evaluating and ranking the hierarchy. Various techniques can be used at this point to project the population growth in each area as well as constructing a map that represents the population density with respect to the land use/cover. The hierarchy of a node can then be ranked using population density over the land use/cover classification which we will discuss in more detail later. The hierarchy of a node is not only useful for visualisation of the evolving importance of centres within the emergent urban structure, but this can also be quantitatively used as a source for calculating spatial interaction during the next stage of the model simulation.

Spatial interaction is the most important module for visualizing urban-regional relationships. Users or decision makers need to measure spatial interaction for any kind of urban and regional planning which involves urban growth, for transportation is the key to the way urban growth diffuses. As mentioned earlier, population density is a key variable in this study which is set as 'mass' in the gravity model functions. Two different variables are output from the spatial interaction model which involve the **centrifugal** and **centripetal forces**, noted in the previous chapter. The centrifugal force originates from push factors that flow from the centre to hinterland while the centripetal force is influenced by pull factors that agglomerate the economic activities into a centre. The model used here is a modified spatial interaction which generates these two forces whose outputs from this module are set as layers in the DSM.

All these processes are implemented using relatively simple concepts and techniques. However, they all generate a degree of unpredictability in their feedbacks in each time period. The model is therefore based on complex processes and outputs where all these systems consistently work together, and this enables the model framework to replicate the kind of complexity that is observed in real urban and regional growth processes. In the next section, we will discuss in detail the components in this framework.

The Dynamic Spatial Model

The dynamic spatial model is an approach that transforms the static spatial model through incorporating the temporal dimension into the spatial databases. The DSM is to be implemented by applying GIS functions to the programming of various flows between input and output. In Figure 3.2, we show the cartographic model approach as we envisage it in the model. Readers and users of the model might be familiar with this concept as the overlay operation in traditional GIS. However, the cartographic model uses many more spatial functions to create new map layers than do traditional overlay techniques. The model is constructed on a raster data model (see Tomlin 1990 and Demers 2005) and map sources collected from GIS databases are classified for use in this evaluation process. GIS functions are applied to the system for generating new map layers, and all map layers are finally evaluated using the weight-overlay method or through application of other mathematical functions based on synthesizing different map layers from linear or quadratic forms of equation.

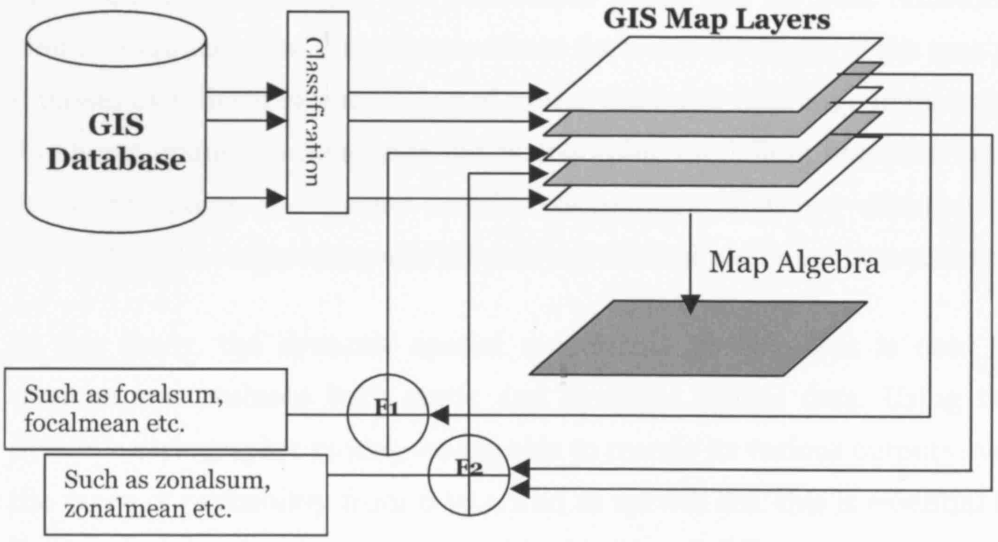


Figure 3.2 Cartographic Modelling Concepts

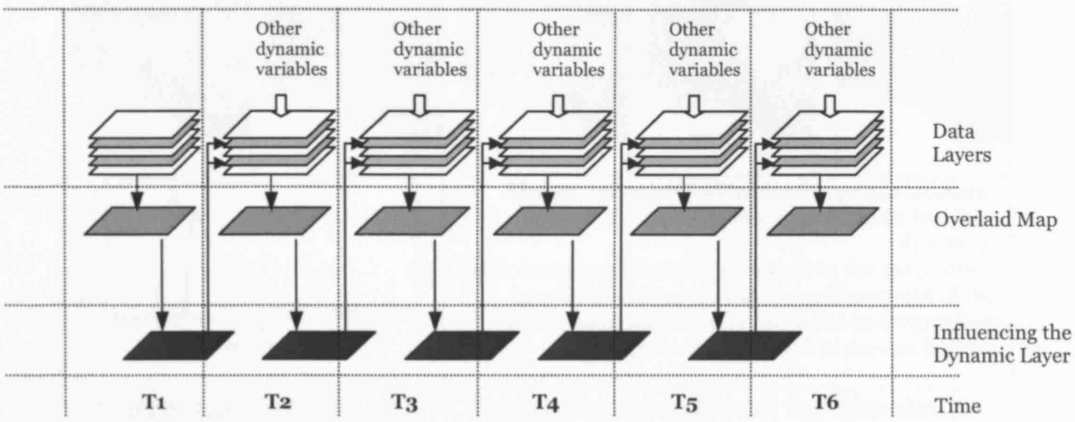
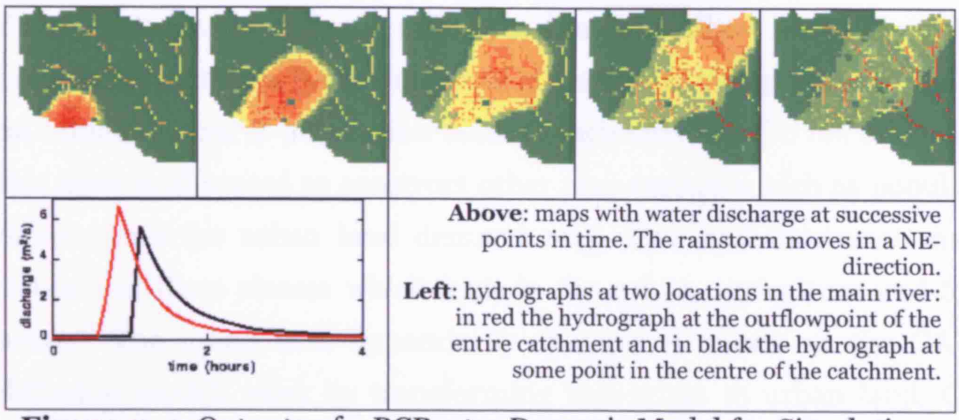


Figure 3.3 Example of Dynamic Cartographic Model

The dynamic model enables key variables to be fedback into the system. An example of the dynamic input and feedback system employed in this cartographic model is shown in Figure 3.3. This is not only useful for visualization, but it also can be set as input to other modules or applications (in this study, in the CA and diffusion modules, for example). Other modules also feed various changes back into the spatial model at each time state. An example of an application of this dynamic cartographic model is shown in Figure 3.4. This is taken from PC Raster (see more information in Burroughs and McDonnell 1998) and involves rainstorm over a catchment area. This storm affects the water discharge of the area as it passes over the area, increasing of course the water discharge in the area. As shown, maps are created using cartographic modelling functions such as interpolation and raster overlay techniques that are effective in elucidating the importance and intensity of the real phenomena over time.

In this study, the dynamic spatial model is a module that is used to manage and evaluate both static and dynamic spatial data. Using the dynamic cartographic model, we are able to rescale its various outputs over the range of probability from 0 to 1, and as we will see, this is essential in the constraining factors that are used in the CA and diffusion modules.



Above: maps with water discharge at successive points in time. The rainstorm moves in a NE-direction.
Left: hydrographs at two locations in the main river: in red the hydrograph at the outflowpoint of the entire catchment and in black the hydrograph at some point in the centre of the catchment.

Figure 3.4: Outputs of a PCRaster Dynamic Model for Simulation of Surface Water Discharge Resulting from a Rainstorm Moving over the Catchments
Source: http://pcraster.geog.uu.nl/introduction/time.html

Figure 3.5 illustrates the various map layers used in the model. Variables concerning urban and regional growth were selected, and as mentioned earlier, there are two main variable sets to be discussed here. Firstly, the main inputs of spatial data including river and water bodies, major transportation links, administrative boundaries, population distributions, land use/cover maps and the topographic slope map. Other datasets are generated from various input map layers using several GIS functions and these include the population density map/surface (dasymetric map), the urban land demand map, and the potential surface map (which is based on a synthesis of both centrifugal and centripetal forces).

Primarily, the river and water bodies map is used to constrain development by masking areas from consideration for growth or change. There are strictly no urbanized cells which can be placed on cells which represent the masked area. In practice, there are other features that can be set as masking areas: for example, restricted areas due to institutional constraints and green, open space, recreational areas. In fact in this study, we assume that only river and water body features are so determined because of lack of information on these other factors but also because the scale at which we are working tends to limit the importance of these other data. Transportation features illustrate the accessibility of urbanized cells. Cells close to existing roads have a higher potential for urbanization. The distance function is employed here is thus central to this model. Sub-

district boundaries (called 'Tambon' in Thai) are utilized as 'administrative boundaries' at this point in the model. This feature is important in defining an initial number of population units in each territory. On the other hand, this layer is also used to construct other map variables such as population density and the urban land demand map. The land use/cover map is defined as three classes which include the urban, agriculture and forest areas. The urban area dynamically changes of course as the CA and diffusion models work by transforming non-urban to urban land. Other changes in land use dynamics are reflected in the transition rules. The land use/cover map is thus only one component used for calculating population densities. The last initial variable used in the model involves topographic features where we invoke the obvious notion that urban cells cannot be placed in mountainous areas or on unstable land.

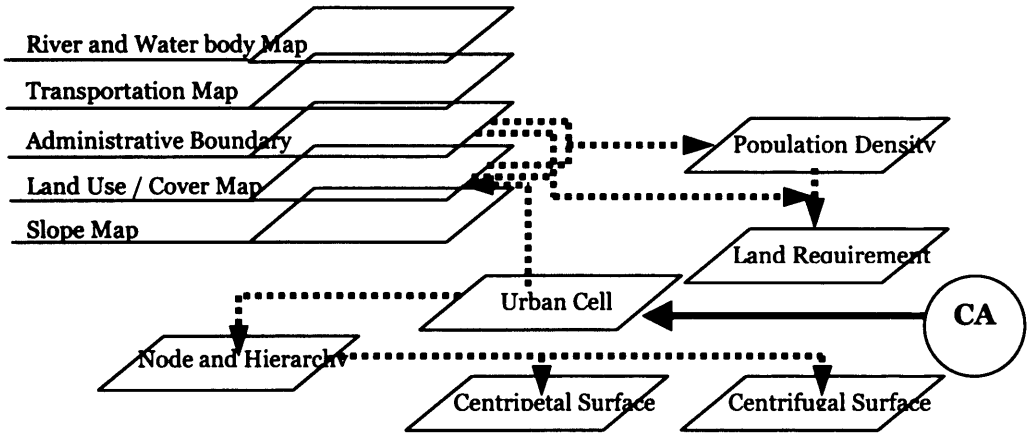


Figure 3.5 The DSM Maps Concepts

The map generation module is constructed around the raster-based model and this reads through the data layers to generate all the data used by the main model as raster images. All raster layers are defined as part of the initial state for the simulation. We should briefly note that there are other data preparation steps in the model that are complicated and tricky to implement. For example, the population density maps are based on administrative boundaries and then the land use/cover map and the urban land demand map are constructed based on this density of population weighted by land use types. The final map generated from the GIS functions involves the potential surface. Here we assume that there are two

forces influencing the urban system and its hinterland based on the centrifugal and centripetal forces (or push and pull forces respectively). The spatial interaction model which incorporates 'mass' and 'distance' is used to calculate both of these surface maps and this will be explained a little later when we review the spatial interaction model in more detail.

According to the DSM concept, some map data used in the model which we indicate in Figure 3.5 is to be defined as dynamic. Some other layers are assumed to be static in their measurement such as slope, administrative boundaries and the river map. All cells on the land use map of course change over time as the cells are generated from the CA and diffusion system. We discussed this a little earlier and it is the essence in the model of how feedback takes place and how nonlinearities and unexpected outcomes can be generated. The main product of the DSM is thus a dynamic constrained map, changing over the time, which is the central input used at the next stage involving the process of urban growth and transition.

The Cellular Automata and Diffusion Model

The second component is the CA and diffusion model. Conceptually, CA and diffusion are integrated in order to simulate the growth of an urban system that in general involves a spread or expansion around critical nodes or seed sites. The dynamic results from the DSM are also applied at this stage in order to constrain the CA simulation. As discussed at the beginning of this chapter, in each **time state**, whether a **dead cell** will change to a **live cell** depends upon a **transition rule** applied to **neighbouring** cells. This is the general function of a CA model which enhances these operations through the transition rules. It is at this point that cells are said to give **'birth'**, or to **'survive'** or to **'die'**.

CA and diffusion are applied to three dynamic processes which can be defined as **pure diffusion**, **spreading** or **growth**. The **diffusion** and **spreading** processes are conceptually applied to the cells that give **'birth'**. The growth process stage is developed to represent those cells that

'survive' or **'die'**. The diffusion process thus enables new urban cells to be generated within the cellular space. Generally, to place a cell on the cellular space, the easiest way is to select a random number of coordinates (X and Y) and locate these cells in the space but here a more complicated and realistic method has been developed. Based on Hägerstand's Mean Information Field (MIF) concept, diffusing cells can be generated by randomly defining a potential seed and then enabling diffusion to take place around such a seed.

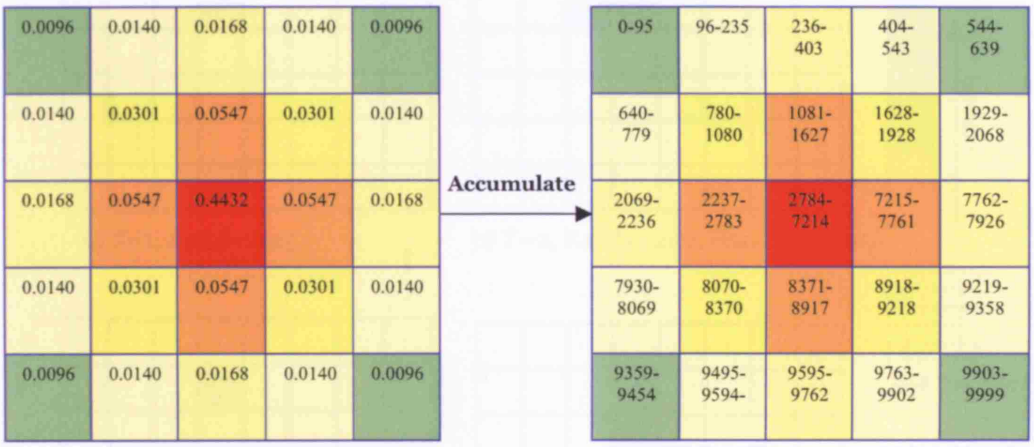
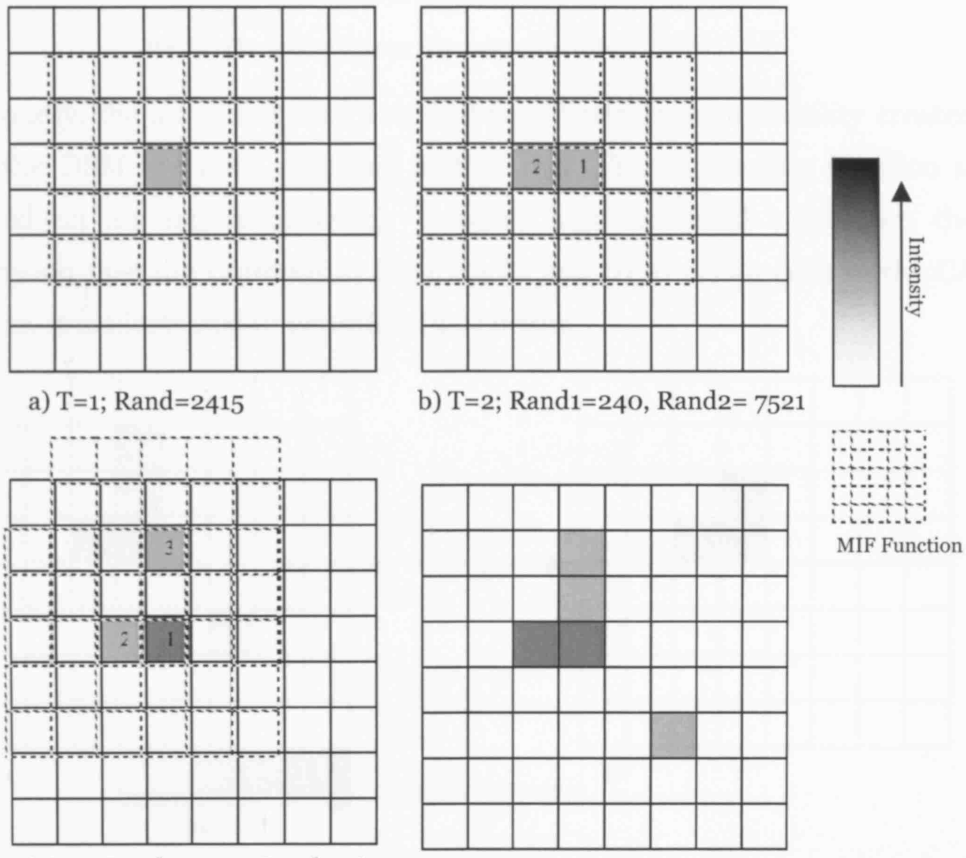


Figure 3.6 The Hägerstand Mean Information Field (MIF)

Figure 3.6 illustrates graphically how the Hägerstand Mean Information Field (MIF) operates. To create one potential cell, a random number ranging from 0-9999 is chosen. For example, a random number generator calls a number, say, 2248; a new cell is then placed to the left hand-side of the central cell. The diffusion module is also controlled by other variables such as time and intensity. Once the new cell is placed in the cellular space, the random generator chooses another number and defines a 'period' for diffusion. The diffusion module keeps track of the process by placing new cells in the diffusion space based on the central cell and MIF, until it reaches the length of the period defined. If cells are repeatedly placed in this way, the intensity of these cells will accumulate as we show in an example scenario in Figure 3.7a; the potential seed is placed on a cell in a diffusion space with a period equal to 4. A random number, 2415, is generated and evaluated from the MIF. At time T = 2, a new cell is placed on the space (number 2) as we show in Figure 3.7b. The diffusion process

then repeats the same operations on all the cells generated from the last step. At T = 3, cell number 1 is repeatedly located from the MIF evaluation in T = 2, and the intensity then increases as we show in Figure 3.7c. The final result after 4 time periods of diffusion is presented in Figure 3.7d.



a) T=1; Rand=2415

b) T=2; Rand1=240, Rand2= 7521

c) T=3; Rand1=9904,Rand2=6200 Rand3=8421

d) T=4

Figure 3.7 Example of Diffusion Process with Time and Intensity Variables

As shown in Figure 3.7, cells with a high intensity (dark grey cells) identify locations of high potential for urban growth. It is possible therefore that more than one cell can be generated as urban in the CA space during a time period. A GIS function, - a focal statistic – is then applied to calculate the probabilities of cells in the diffusion space. Equation 3.1 and Figure 3.8 conceptually represent the probability computations for cell *i,j*. Intensity values of a central cell and its neighbours are evaluated to create potential

urbanised cells. For example from Figure 3.7d, the focal function is applied to create possible urbanised cells as we show in Figures 3.8a and b.

$$PD(U_{i,j,t+1}) = f(ID_{i,j,t}, ID_{k,l}) \quad (3.1)$$

where
ID : Intensity of diffusion
i,j : Central cell
k,l : Nearest neighbours of i,j (8-neighbours)
t : Time state
PD(U) : The probability to generate a new cell

Secondly, the spreading function and the constrained probability created by the DSM are both operated at this stage. The spreading function is based on an algorithm which converts high potential cells from the diffusion (see the example in Figure 3.8a and b) to urban cells in the CA space. It is illustrated in equation (3.2) below

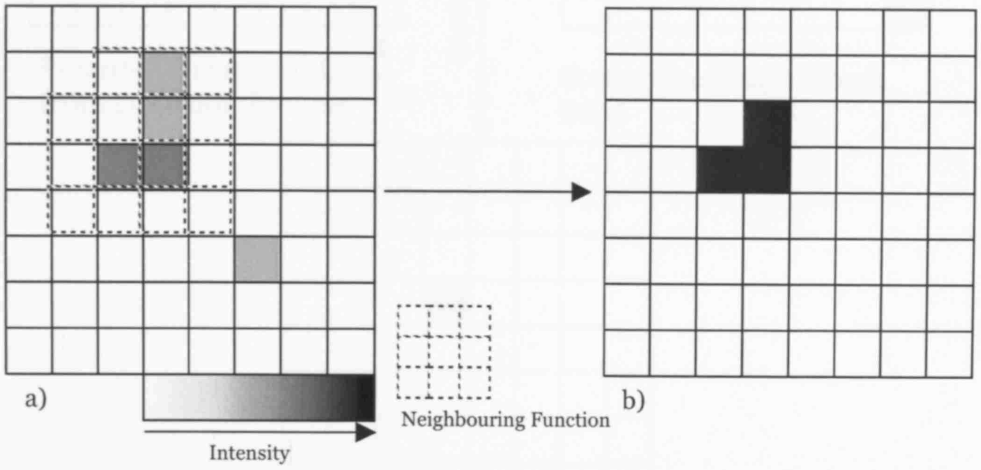


Figure 3.8 Calculation of Probability Urban Cells

$$U_{i,j,t+1} = f(CONS_{i,j},\ PD(U_{i,j,t+1}),ran) \quad (3.2)$$

where

$CONSi_{,j}$: Constrained factor calculated from cell *i* and *j* of dynamic map layers, The value is ranged 0-1
PD(U) : The probability from the diffused space
ran : Random number range 0-1

It is possible that a cell with very high potential to be urbanised will fail to become an urban cell. In addition to the constraining factors, a random number is generated and attributed to each cell in each time state; a threshold is thus defined for that cell to become urban. If the random number of any cell reaches the threshold, those cells will be entirely placed

in the CA space (see equation 3.2). Cells generated from the spreading process are thus set as seeds for 'growth' in the next stage and this is what we refer to as a **birth** cell in the CA urban space. Figure 3.9 illustrates how an urban seed in the CA space can be generated using this function of spreading.

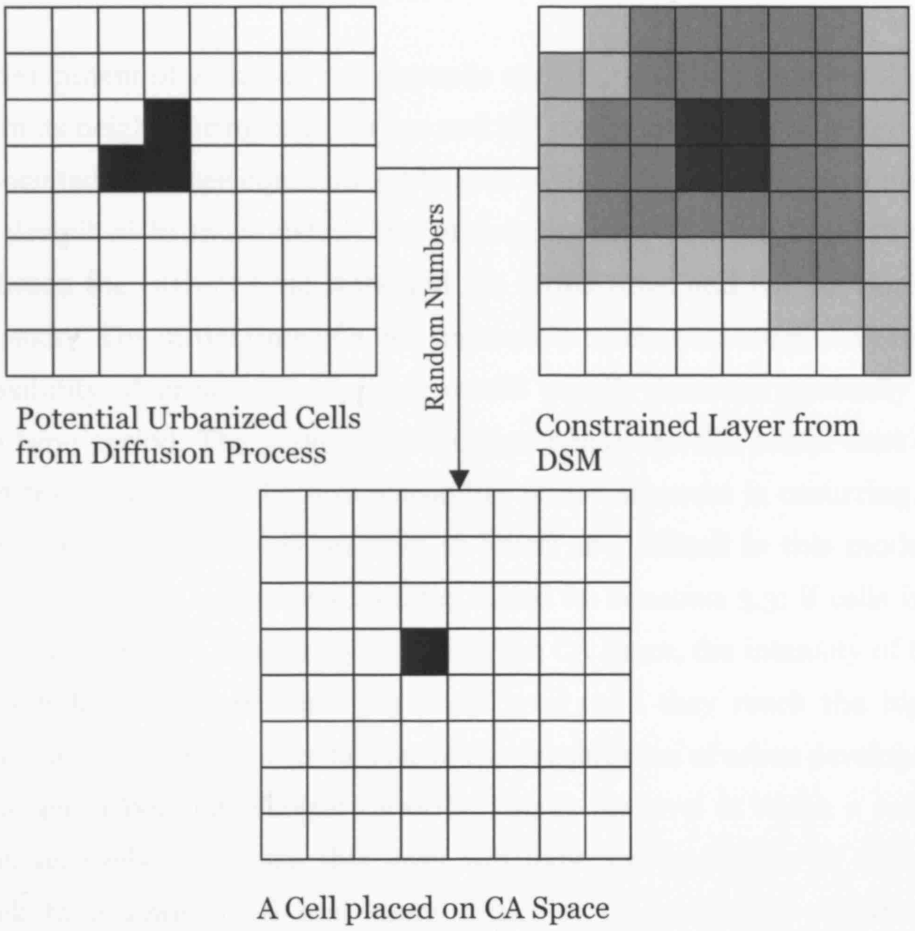


Figure 3.9 Example of the Spreading Process in One Time State

Finally, the growth algorithm is developed based on simple CA operations. In this process, there are two states of a cell which are represented as **survive** or **die**. The general conditions in this process are that: 1) birth cells generated in the previous two processes can be changed to other states; 2) cells survive depending upon the condition of the cell itself, neighbouring cells, their age and the constraining factor; 3) if cells meet the *survive* conditions, those cells will actually remain in their existing state, survive and will become more stable, and 4) if cells reach a *die*

condition defined when the development intensity of cells declines to zero, those cells will be changed to die status. These rules are all pretty simple, but in practice they are complicated by the spatial and temporal context. This provides the degree of unpredictability hinted at in the previous discussion which is a consequence of many different conditions combining with the stochastic rules for cell transition.

Development of an urban cell depends on the probability function derived from its neighbouring cells, its age and the constraint that is reflected in its associated map (see equation 3.3 below). At least two features now need to be described in more detail: the probability derived from the difference between the current time state and the initial time, and the development intensity. The initial time of a cell represents its chances for growth for the possibility of urban development would usually decrease gradually over the time period. The greater the difference between the actual time state and the initial time, the less possibility of development is occurring. The development intensity is set over 10 levels as a default in this model. At every time state, cells are generated based on equation 3.3; if cells in the last time state are repeatedly placed in the CA space, the intensity of those cells will accumulate more than one level until they reach the highest intensity level 10 (see also the idea of the densification of urban development in a cell in Wu and Webster [2000]). This is the level at which a cell will survive. Cells that reach this level will have less possibility of changing back to a lower level and more difficulty in completely reverting to becoming a dead cell. This concept will be discussed in more detail in a later chapter when we come to deal with the programming of the model.

$$UG_{i,j,t+1} = f(CONS_{i,j,t}, UG_{i,j,t}, UG_{k,l,t}, IT_{i,j,t}, ran) \qquad (3.3)$$

$CONSi,j$: Constrained factor calculated from cell i and j of dynamic map layers, The value is ranged 0-1
UG : Developing cell (grey cell)
i,j : Central cell
k,l : Nearest neighbours of i,j (8-neighbours)
t : Time state
IT : Initial time defined to cell i,j
ran : Random number range 0-1

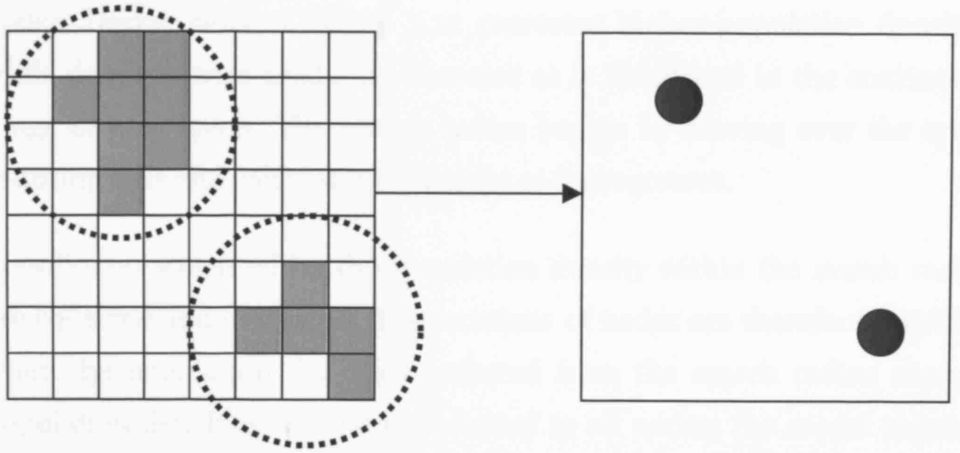
As described so far, the urban development simulation works by seeding cells which can then be transformed by chance, by transition rules, and by constraining factors which operate in the cellular space. The outcomes of this process are unpredictable in that the varieties of urban cells that emerge within the CA space cover all the possibilities of the growth process – birth, death and survivorship. In the next section, we will consider how urban growth of this kind affects the hinterland around the various seeds of growth and the various satellite cities that exist within the wider space.

Nodes and Hierarchy in the Model

Questions that are all pervasive when looking at an urban and regional area involve how to identify the centre and size of the various towns or cities that occupy the territory in question. The answers vary depending on the data available, the scale, and the shape of the urban areas defining the region. This is a very general question asked by a variety of users; for example, urban planners, real estate and facility managers and so on. In this section, we will show how such questions can be approached through the representation of **centrality** and **hierarchy** which are essential to the nodal structure of the wider region within which the sort of cellular allocation described in the previous section is addressed. Consequences of both centrality and hierarchy play two roles in the model: for visualising and measuring the morphology simulated by the model but also for defining the structure on which spatial interaction takes place which is the function of the next stage of the modelling process.

From our review in Chapter 2, a node in this model is assumed to represent the centrality of a group of urban cells. Figure 3.10 represents how we are able to define and generate nodes in a CA space. Practically, nodes can be created based on distributions of urban cells within a defined search radius as we illustrate in Figure 3.10a. A module used for counting cells within the search radius can be developed, the coordinates gathered from calculation being listed and placed in the node space as in Figure 3.10b. The two circle shapes are represented as two centres which are generated from the relevant functions applied to the CA space.

Nevertheless, in preparing to calculate the hierarchical position and extent of each node and to define the node in a more reliable way, administrative boundaries from DSM module are utilised as another condition for the node generator. Figure 3.11 illustrates this comparison between two types of node generation. Figure 3.11b shows clearly how nodes can be created by constraining them by the administrative boundary map. In Figure 3.11a, only one node can be generated which is separate from two nodes shown in Figure 3.11b. Conceptually, this implies that one urban area has possibly more than one centre. This function also considers this by developing an additive algorithm which generates new nodes in a territory. The node (shown in the upper left of the node panel in Figure 3.11b) demonstrates how one area is able to have more than one centre.



a) Urban Cells on a CA Space with Moving Search Radius

b) Two Nodes generated from the Distribution of Cells within Search Radius

Figure 3.10 A Simple Concept of Node Generation in the CA Space

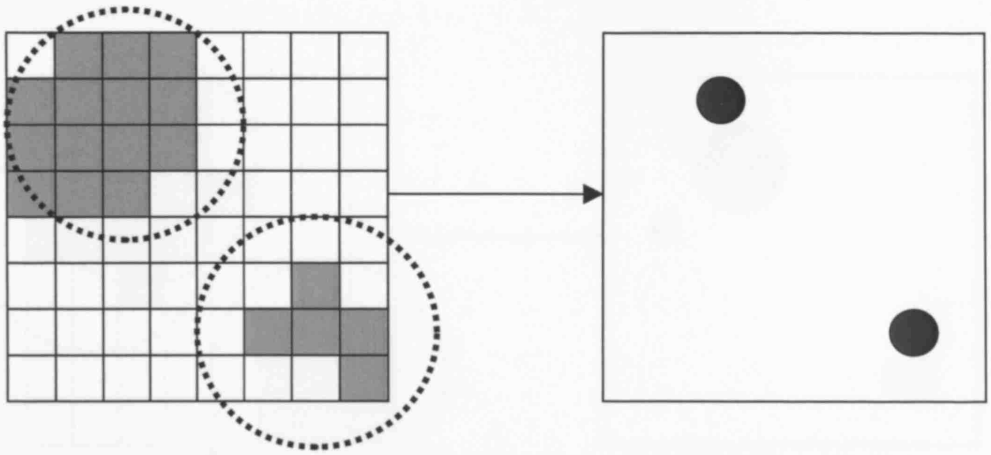
Nodes summarise and visualise the location of the centres. This can be useful in envisioning the distribution of all the centres in an urban system and its regional area. Nodes located in such a space also provide some sense of the pattern of interaction; that is the distribution of nodes. However, the nodes represented in the space are not usually sufficient for visualising the actual spatial interaction for these barely account for the two main variables of interaction as in the gravity model which are mass and distance. In fact, there are many variables that can be used to define

mass although in this study, only one variable, the population of each territory, is used to do this. Population is of course the measure of how cities grow and it is also essential to how the population can be projected into the future. In the model, a simple population projection function has been constructed from population data in each district which is first initialised in the DSM module. This acts as a thematic layer of population numbers which is then converted to raster format. By extending the node generation, the hierarchy is defined based on population density, the number of cells and the search radius. In the case of population density, this will be discussed in the next section where we deal with the dasymetric mapping method.

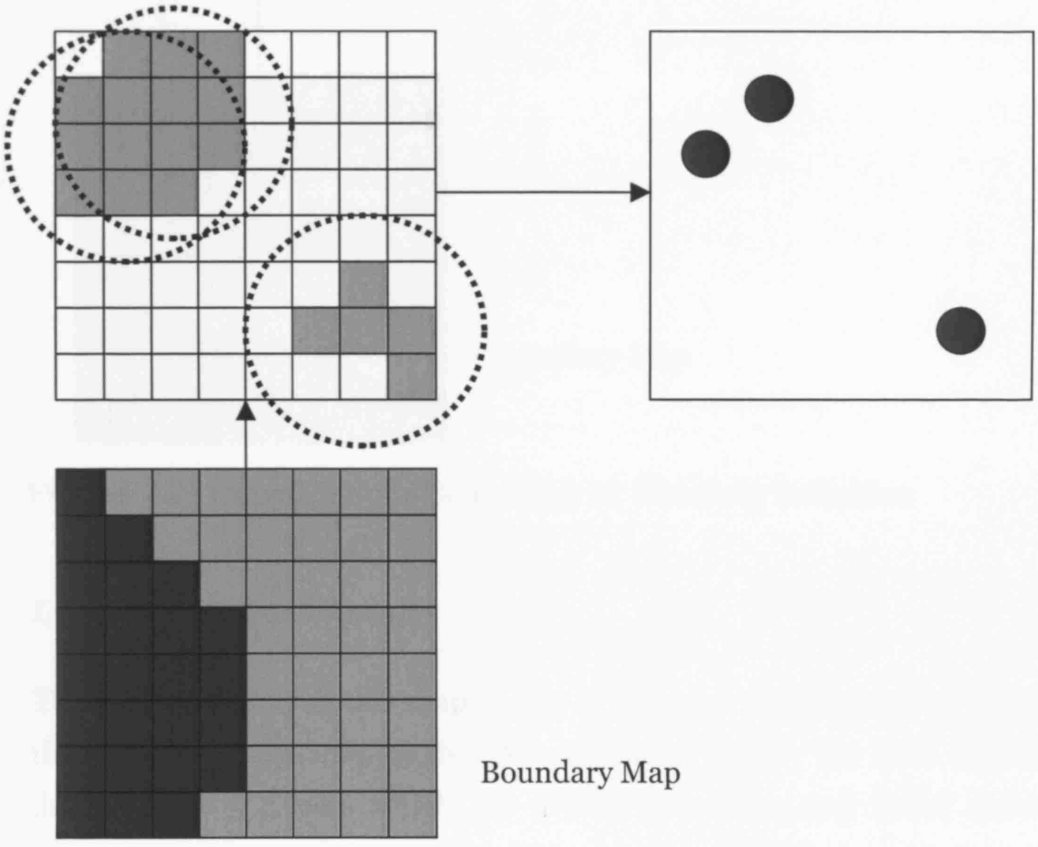
The population density of cells is used to calculate locations and sizes of all nodes. Darker cells in Figure 3.12 represent higher population densities while density levels gradually decrease as is illustrated in the continuous range of grey levels. The search radius begins by moving over the space counting cells and calculating densities as it progresses.

Coordinates weighted by the population density within the search radius are collected and evaluated. The locations of nodes are therefore modified when the population densities gathered from the search radius change. Population densities are then attributed to all nodes; the model requires the density in order to rescale to a range which identifies node sizes. Nodes are then converted graphically to the different orders as illustrated in Figure 3.12.

The output from this process not only represents the spatial distribution of nodes and their hierarchy in the urban-regional space, but it is also essential in being able to visualise the patterns of spatial interaction. We will discuss these issues more fully in a later section where we deal directly with how spatial interaction is specified.



a) Node Generation without Administrative Boundary Conditions



b) Node Generation with Administrative Boundary Conditions

Figure 3.11 Simple and Advanced Node Generation in the CA Space

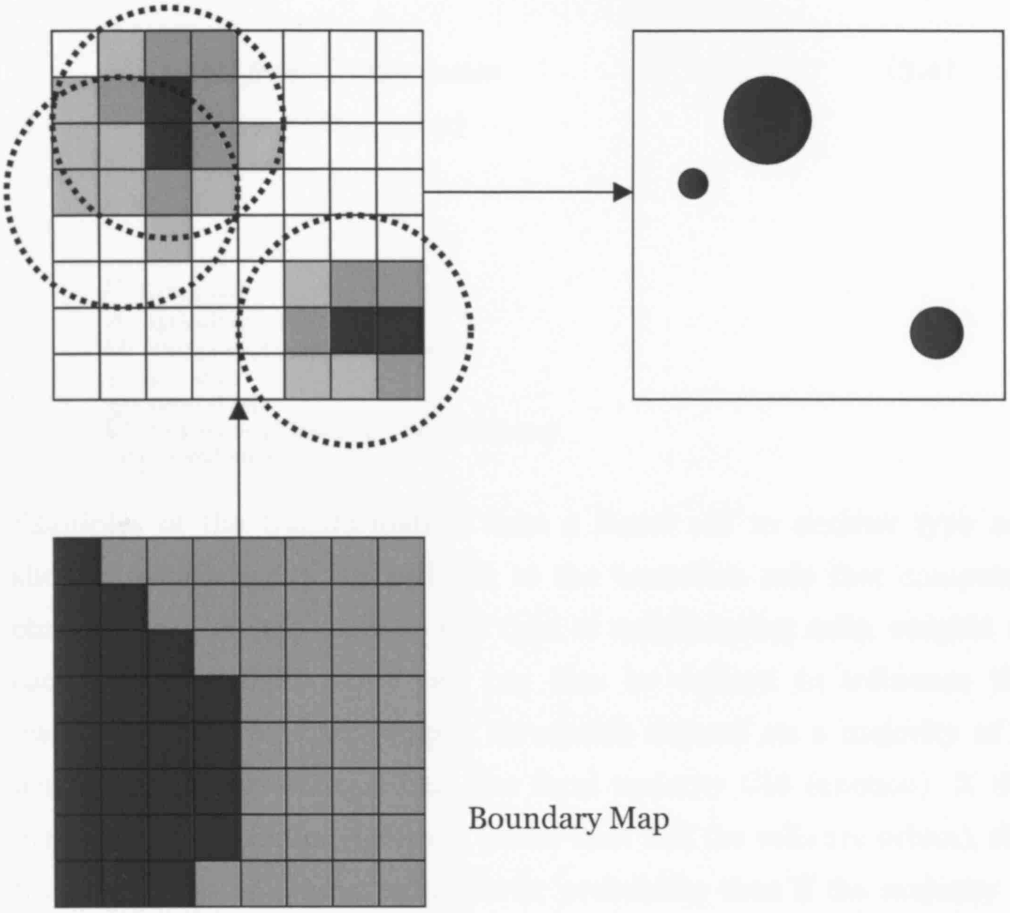


Figure 3.12 Extended Node Generation for Hierarchy Definition

Land Use /Cover Changes

The land use/cover (LUC) map is one of the most important map layers that drives the dynamic model. As mentioned earlier, the LUC map is defined over 3 classes which are urban, agriculture and forest areas. Essentially, urban cells are updated from the CA and diffusion simulation. In this section, we will show how the other two classes, agriculture and forest, can be modified over the time period.

Equation 3.4 represents changes in the cell states to urban and agricultural cells. The model assumes that the forest area can only change its state to agriculture, and it cannot directly change to an urban use. This is because

forest land use is far from the urban areas and that any transition would be to agriculture before urban use occurred.

$$U_{i,j,t+1} = \textit{Update from CA simulation} \qquad (3.4)$$
$$A_{i,j,t+1} = f(LU_{i,j,t}\ ,\ LU_{k,l,t},\ ran)$$

where

U : Urban cell
A : Agriculture cell
LU : Cell in land use/cover layer
t : Time state
i,j : Central cell
k,l : Nearest neighbours of i,j (8-neighbours)
ran : Random number range 0-1

Examples of the transformation from a forest cell to another type are shown in Figure 3.13. In addition to the transition rule that computes change based on the number and type of neighbouring cells, weights of each type of neighbouring cell can also be defined to influence the transition. The rules are simple; thresholds depend on a majority of 8 neighbouring cells being urban (the focal majority GIS function). If the majority of neighbours are urban (more than half the cells are urban), the transformation will occur with greater probability than if the majority is agriculture. In Figure 3.13 for example, it is reasonable to approximate these probabilities by thresholds which run across the lower integer numbers, 1, 2, and so on.

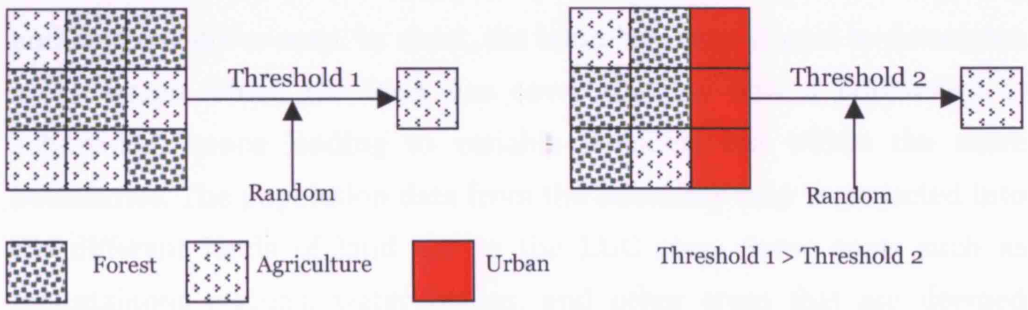


Figure 3.13 Examples of a Change of Forest to Agriculture

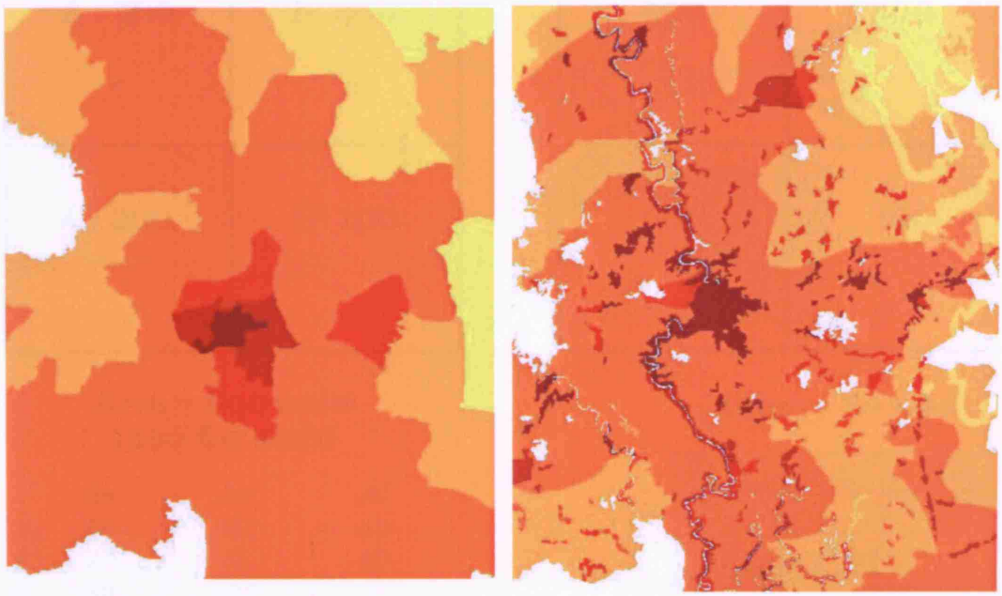
The CA simulation generates change of urban cells in the LUC map and such changes in urban cells affect other types of land use/cover. In turn, these changes in urban and other types of LUC cause transformations in other map layers within DSM, a kind of cascade occurring which marks the results of the positive feedbacks in the model. In this way, the DSM then outputs the decision rule map which thence constrains cells from taking on certain types of development within the CA allocation process.

Dasymetric Mapping

This element is a part of the DSM module, but it is important to understand how it works in the general model. Dasymetric mapping is a hybrid technique which classifies quantitative data within a map which is represented as a surface. In this study, this technique is applied to maps of population density. A population map can be created as a number of classes in an area map based on some administrative areas such as provinces, districts or states, with the largest scale of district map characterising the population in the most detail. In Thailand, based on the political boundary and population data available, the most detailed map is the sub-district boundary map where the political boundary of Thailand is divided as a country into provinces (Changwat), districts (Amphoe), sub-districts (Tambon), and villages (Mooban)). Here, the sub-district boundary map determines the population base map.

The purpose of dasymetric mapping is to modify the classified map on the boundary base to a new classified map based on both the boundary map and land use-cover map. In short, the boundary map is used to determine areas within which the land use cover dictates how a population is subdivided, hence leading to variable densities but within the same boundaries. The population data from the boundary map is projected into the different kinds of land use in the LUC map. Some areas such as mountainous regions, water bodies, and other areas that are deemed unsuitable for human occupation are excluded from the map. The population data are projected only to those areas that can be inhabited with respect to the conditions defined in creating the map. For instance,

people will not live in areas that have topographic slopes greater than 1 in 10 (10 percent), not in densely forested areas and so on. Not only the physical condition, but also a weighting for concentration of the population can be defined in this way. For instance, people living in the urban area might be, say, approximately 85 percent of those in the region with the other 15 percent living in rural agricultural areas. The results of this process will then reveal where people live as Figure 3.14 demonstrates. Two different population density maps based on the same population data and the same area but within different land cover regions illustrate how different partitions of the population from a density surface are made.



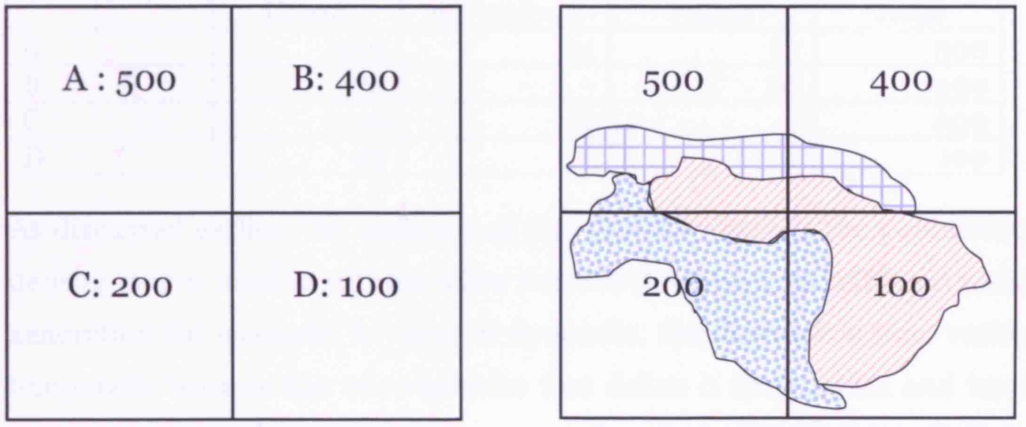
a) Choropleth Map b) Dasymetric Map

Figure 3.14 Traditional and Hybrid Population Density Map

A practical question is how to make such a dasymetric map. Maps required in the process include boundary maps with population attributes, LUC maps and other condition maps such as those dealing with water bodies and physical slopes. These would first be superimposed in creating a dasymetric-base map.

Suppose the boundary map includes areas A, B, C and D as shown in Figure 3.15. Each area has different amount of population which is shown in each territory. The base map created from the overlay method must be

in three classes for these are the three land cover types of urban, agriculture and forest. The table in Figure 3.15 describes the area of each class in each boundary. *Ru*, *Ra* and *Rf* represent relative densities of each land cover type: 85 percent of the population live in the urban areas, 10 percent in the agricultural areas and 5 percent live in the forest areas. The relative density can be adjusted depending on which area the method applies to; the percentage of people living in the urban area can be less than 85 if the areas are less urbanised for example.



Area = 800 units (200 for each)

	Au	Aa	Af
A	40	10	40
B	20	0	40
C	30	100	0
D	100	60	0

Figure 3.15 An Example of Calculating a Dasymetric Map

$$P_{ni} = N \frac{(R_{ni} A_{ni})}{\sum_i (R_{ni} A_{ni})} \qquad (3.5)$$

where

P : Estimated population in area unit
R_n : Relative density
A_n : Area of mapping unit
N : Actual population
i : Boundary area
n : Land use /cover unit (in this case: urban, agriculture and forest)

Equation 3.5 is used for calculating population areas in the dasymetric map. From the example in Figure 3.15, new populations in each area are calculated as represented in Table 3.1. Note that the total population in each area is preserved and that the amount of population in each area will not change after calculation. Lastly, the area of each land use type is measured and this is used for calculating the population density associated with the distinct land use cover types which define areas within the map.

Table 3.1 Population computed from the Dasymetric Mapping Method

	Urban	Agriculture	Forest	Total
A	459	14	27	500
B	370	-	30	400
C	126	74	-	200
D	93	7	-	100

As discussed earlier, the outcome of this process which is the population density map is used in various other parts of the model, in DSM and node generation for example. In term of dynamics, the dasymetric map varies temporally because the two variables that define it (population and land use-cover) are always changing over the time. Population changes according to the rates of population growth which are defined in a structured but random fashion over the time. The land use/cover is modified using the CA urban simulation and land use/cover transition rules. This technique has been chosen because the choropleth map represents much less detail concerning the population than the dasymetric map as Figure 3.14 shows. Some of our sequential modules that use density information such as the node generation clearly need detailed population information, hence the need for this kind of modelling.

Accessibility

A critical issue is the distance that urban cells are able to exist from road location which usually represent the infrastructure which guides urban development. This question is central to explaining real urban growth phenomena. Obviously, when we look at the map showing an urban or city settlement pattern, the picture that we frequently find is a star-like

shape and/or a linear form, with tentacles of growth emanating from its various hubs like the spokes of a wheel. If there are main roads existing in the map, we may often find that the urban areas tend to grow up along these roads, as ribbon development, from their central nodes into their surrounding areas. As a result, it might be hypothesised that roads have an extremely important influence on the way such cities actually grow.

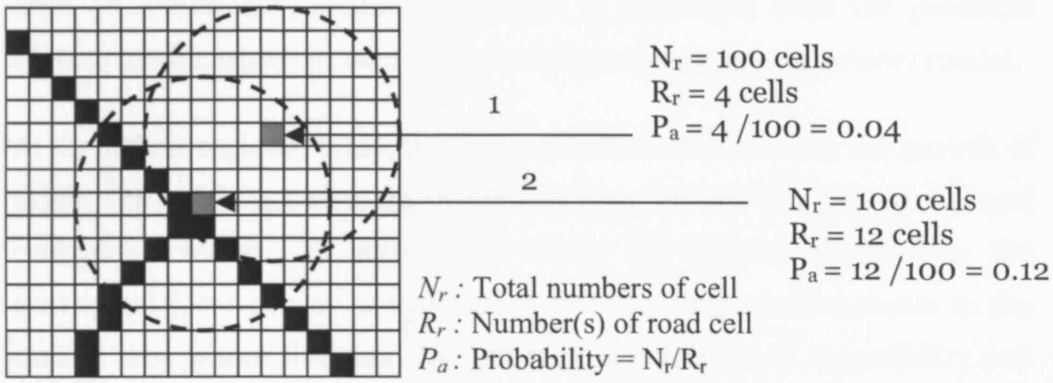


Figure 3.16 An Example of the Accessibility Probability Calculation

For the model development, accessibility was set as one layer in the DSM module. The probability function for accessibility is based on a search radius, the total cells within this search radius, and the amount of road cells captured within this search radius. Figure 3.16 demonstrates an example of how the probability of accessibility is calculated. Suppose that the total number of cells (N_r) in a search radius is 100 cells. When the search radius moves around the image, the number of road cells (R_r) will also be counted. The probability can then be calculated using the equation shown in the figure. The accessibility probability of cell 1 (grey cell) is 0.04, while cell 2 is 0.12. It is reasonable to assume then that cell 2 has a greater opportunity to become an urban cell than cell 1.

In practice, this probability value is not actually used directly in the CA model, but in the DSM module as mentioned earlier, we set up this map as a layer which calculates the overall constraints that are then imposed on the cellular automata. In this way the influence of road infrastructure is taken into account as a constraining effect.

Spatial Interaction and Urban Forces

Spatial interaction is a class of models that is able to both explain and simulate how two or more objects (nodes in this case) interact together. In this model, we represent spatial interaction effects as a surface map where all cells in the space have their own interaction attributes. Again, this is the way we enable such influences to be communicated to the DSM module later on. Practically, spatial interaction is developed from the potential surface model, which in turn is derived from the traditional gravity model.

At the urban-regional scale, there are two forces concerning the growth of urban areas. We noted these in the last chapter as being centrifugal and centripetal forces. To summarise from our earlier discussion, the centripetal force causes centralization in attracting establishments to the central area where they may benefit from advantages of accessibility and agglomeration economies. On the other hand, the centrifugal force causes decentralization and urban sprawl as it pushes dwellings and businesses away from congested, expensive inner city areas towards the suburbs. What we have done is to adapt the gravity model to embrace these two opposing forces and although force is an abstract concept that, like utility, is invisible and immeasurable, we will define force with respect to the distance function as action-at-a-distance. Then, it is possible to adapt several different distance functions to illustrate the way these two opposite forces can be integrated into a single model formulation.

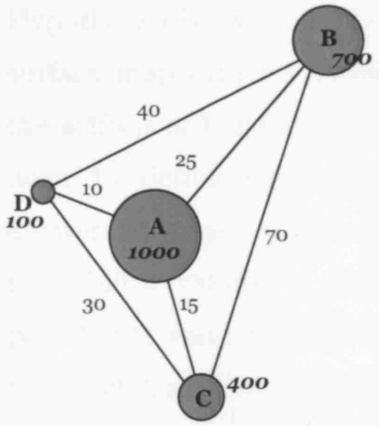
The node and hierarchy map as reviewed earlier in this chapter is the source for calculating the interaction map as a potential surface map. Equation 3.6 represents the general function which computes this potential. In this case, population density is attributed to each node and is defined as the mass (M) while the distance (D) between two points is assumed to be Euclidean distance. Generally, the exponent of distance can be set as a constant such as 1 or 2. This can make a substantial difference in the operation of the model (which is based on many potential interactions) in terms of both calculation time and the display of results.

$$P_i = \sum_{j=1}^{k} \frac{M_j}{D_{ij}^e} \tag{3.6}$$

where

P_i : Potential of node i
M_j : Mass of node j (j=1 to k)
D_{ij} : Distance between i and j
k : Number of neighbouring nodes around i
e : Exponent of distance (usually set as 2)

Figure 3.17 shows how to compute the potential of example nodes. Note that all the self-distances used in calculating a distance from a node to itself are approximated by half of the minimum distance to all other nodes: for example, the minimum distance measured from node A is from A to D equals 10, hence the distance used to calculate the self-distance is 5.



$P_A = (700/25^2)+(400/15^2)+(100/10^2)+(1000/5^2) = 43.90$
$P_B = (1000/25^2)+(400/70^2)+(100/40^2)+(700/12.5^2) = 6.22$
$P_C = (1000/15^2)+(700/70^2)+(100/30^2)+(400/7.5^2) = 11.81$
$P_D = (1000/10^2)+(700/40^2)+(400/30^2)+(100/5^2) = 14.88$

Figure 3.17 Examples of the Potential Calculation for each Node with **e** =2

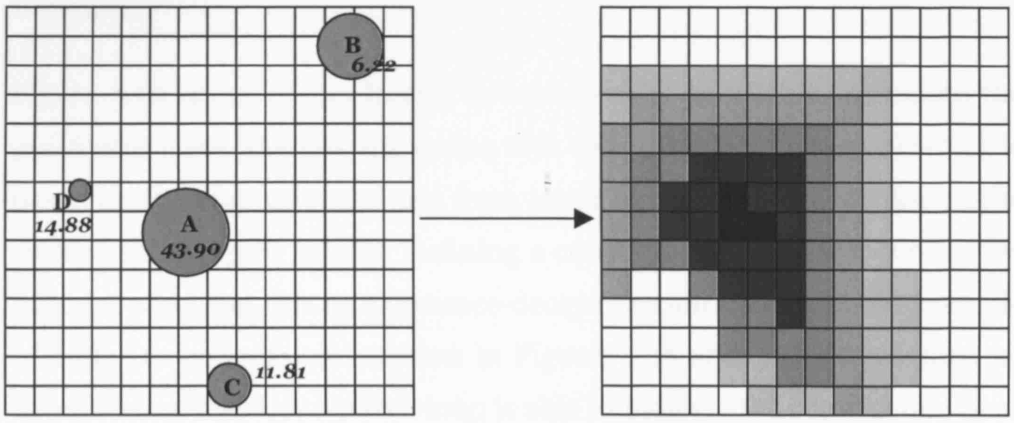


Figure 3.18 An Example of the Potential Surface Map

The values in the example in Figure 3.17 clearly illustrate how a central concentration of potential is identified and built up. The biggest node (highest in the hierarchy), A, has the highest potential value. The smallest node, D, with the lowest mass value, however, has a potential value which is bigger than node B which has a higher mass. This is of course due to the effects of distance in that it is not only mass that influences the potential, but also the distance measured to the centre. Visually, the potential surface can be easily created using several interpolation techniques such as the IDW and Spline methods noted earlier in this thesis. The results can be visualised as the surface map shown in Figure 3.18. All cells in the surface map have their own values, and this is essential in treating spatial interaction as yet another layer in the DSM module.

Hypothetically, we ask the question as to whether or not the potential surface map with a suitable distance function has the ability to visualise the actions of these two forces – the centripetal and centrifugal. If so, we need to define appropriate distance functions. Prior to anticipating answers to these two questions, the forces have to be assumed in terms of probability values to make them a little more concrete. A higher probability value usually identifies higher influence. Considering Figures 3.17 and 3.18, the general potential surface model yields the result that clearly represents concentration towards the centre (node A). The surface map in Figure 3.18 provides even more useful information where the highest potential cells are and how the spatial trend of potential is orientated.

Figure 3.19 below shows how it is conceptually possible to associate the pull forces from surrounding nodes with the centre (Figure 3.19a) which is based on the gradual transition from lower to higher concentration in the surface map (Figure 3.19b). Defining a constant exponential value for the distance identifies this as a distance-decay function as seen in the example of node D's potential calculation in Figures 3.17 and 3.18. It can then be hypothesised that this surface map is able to indicate the centripetal forces that illustrate gradual concentration about this centre.

Push factors are normally significant effects in big cities such as congestion, nuisance, high land values and so on. These factors influence the urban form in quite the opposite way to centripetal forces. To modify the potential surface to visualise these push forces, the exponent used in the equation 3.7 can be altered. As discussed earlier, the constant exponential value (term **e**) represents the distance decay function. This can be modified by applying a linear or polynomial function in order to solve for **e**. Because our model is a simulation, it is necessary to trial the linear or polynomial function to modify the exponent in order to yield a different surface map which is consistent with the push factors defined above.

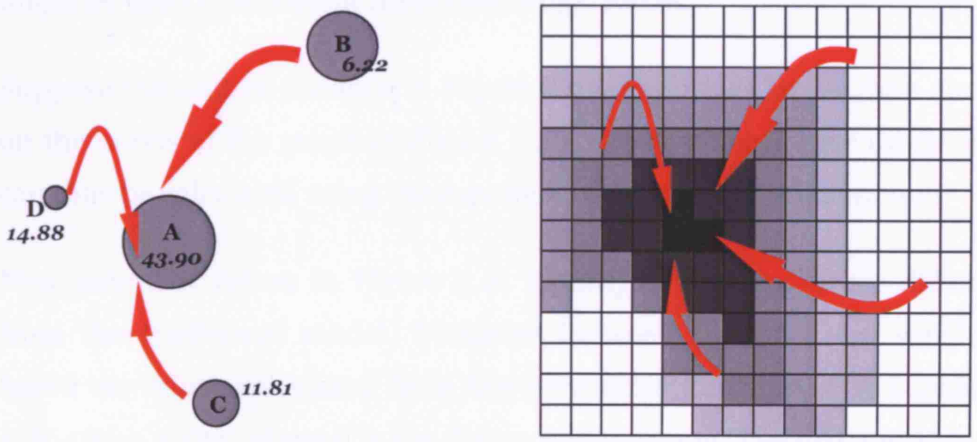


Figure 3.19 Pull Forces interpreted from Node and Potential Surface Maps

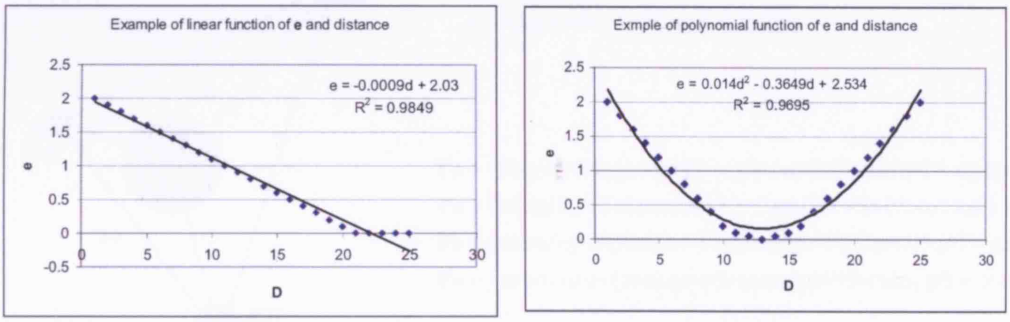
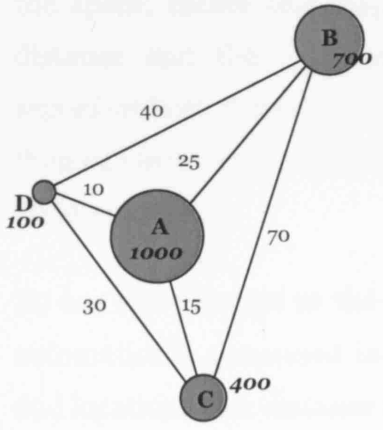


Figure 3.20 Examples of Linear and Polynomial Functions relevant to the Computation of Centrifugal Force

The functions in the examples in Figure 3.20 can be used to transform **e** in the potential surface model in the following way. When the e value is a negative weight in the potential model, a large value of e identifies a low potential. Therefore, the graph shown above might be referred to as the *frictional factor* in urban growth. The frictional factor functions as a way of moderating the distance effect in terms of these exponential values. For instance in Figure 3.20b, this can be interpreted as there being a lower probability of urban growth in locations too close to the centre. If the distance from the centre is from around 7 to 20 units, there is greater probability for urbanization. And as this distance increases, the probability of urban growth decreases. In this way, there are a variety of functions that might be used to represent these centrifugal forces.

Suppose the unit of distance in Figure 3.21 is 10 times larger than shown on the x-axis of the graph in Figure 3.20. A new potential for every node can thus be calculated using the exponents derived from this function.

New potential values in Figure 3.21 identify something rather different from the traditional model. Distances between all nodes are weighted based on values calculated from the polynomial function, $e = 0.014d^2 - 0.3649d + 2.534$. where d is the distance in question. The distribution of e values would thus look like the graph shown in Figure 3.20b.

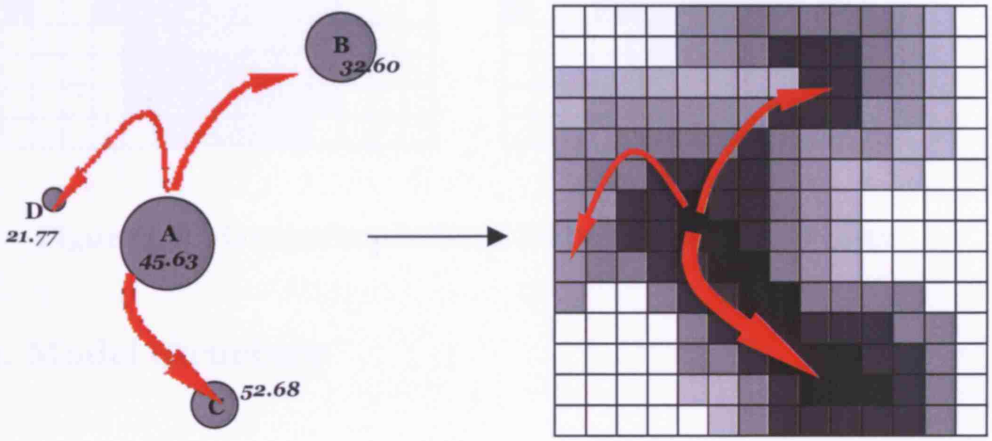


Polynomial function used to calculate the exponent as
$e = 0.014d^2 - 0.3649d + 2.534$

$P_A = (700/25^{1.71})+(400/15^2)+(100/10^2)+(1000/5^2) = 45.63$
$P_B = (1000/25^{1.71})+(400/70^{0.67})+(100/40^{1.30})+(700/12.5^2) = 32.60$
$P_C = (1000/15^2)+(700/70^{0.67})+(100/30^{1.56})+(400/7.5^2) = 52.68$
$P_D = (1000/10^2)+(700/40^{1.30})+(400/30^{1.56})+(100/5^2) = 21.77$

Figure 3.21 Examples of the Potential Calculation for each Node with Derived **e**

Once again, a new surface map which is that in Figure 3.22 can be created. Node A is still the most central node in this space based on its hierarchical position but the potential intensity now changes to another node. The highest potential now changes to node C, 52.68 and the potential surface map is suitably modified as in Figure 3.22b. The arrows over the surface map show in abstract fashion how the centrifugal forces spread from the centre to surrounding areas.



a) b)

Figure 3.22 Push Forces interpreted from the Node and Potential Surface Maps

Finally, Figure 3.23 shows the two surface maps for comparison. The potential values are not distributed in the same way in every direction in the space; rather this disperses to the directions associated with mass, distance and the exponential weights. So far, the two surface maps represent both forces in the form of two different potentials and these can then be employed in the simulation model which in turn is managed by the DSM module.

To use this concept in the model as we have discussed before, nodes are automatically generated in the model by dynamic change in their masses and locations (for distance is also changed by location). Thus over the time while the model is running, spatial interaction between cells is modified dynamically through interaction potential. Finally, the potential in each

cell gets reclassified in terms of the probabilities used as part of the cellular allocation process.

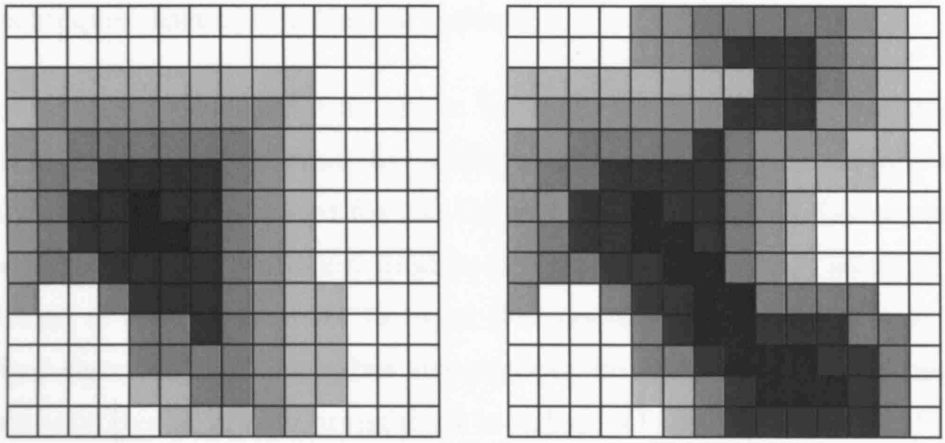


Figure 3.23 Comparison between Pull and Push Surface Maps

3. Model Structure

The conceptual framework described in last section provides some of the key details in the design of the model. In this section, we will describe how all these concepts can be integrated and implemented.

The structure of DSSM is shown in Figure 3.24. There are 4 main modules driving the model which are based on 1) the cellular automata and diffusion module, 2) the dynamic map analysis module, 3) the node and hierarchy generation module and 4) and the urban forces module.

Primarily, the model needs variables which are input in order to set up the initial state of the simulation. The data required include the urban seed set which is the initial state of the CA-diffusion module, and the urban classification which appears in the LUC map. This is a series of layers which are the river and water bodies used as a masking area; the transportation layer; the boundaries map that represents administrative territories; the population map classified according to the boundaries base map; and the topographic feature or slope map. Other initial map layers are the population density map (based on the dasymetric mapping

technique) and the urban land demand map (based on the zonal statistic technique). Moreover, the primary state of the node-hierarchical map and the potential surface maps are calculated and displayed based on the techniques discussed in the last section.

Technically, ten algorithms are to be constructed and these are derived from the conceptual framework illustrated in Figure 3.24. First, three algorithms are developed for the CA and diffusion module (as consistent with the CA and diffusion models described in an earlier section). The fourth is created in order to input the modified spatial features at each time state, and this algorithm also outputs the factors which constrain the probabilities of growth transitions to urban cells in the CA and diffusion module. The fifth algorithm is developed for generating the centre or node of an urban area. The sixth and seventh algorithms involve the spatial interaction models with the sixth for calculating the potential values of all nodes, and the seventh for interpolating the surface maps. There are two sub modules for the potential intensity calculation which involves changing some functions in calculating the different centrifugal and centripetal forces (see details in the spatial interaction and urban forces sections earlier). The last three algorithms concern the graphic representations of the output as well as the writing of map layers to files over definable time states. Some additional algorithms are used for writing outputs to text files, such as the amount of population in each time state, the size and amount of each urban cell, the amount and area of all land use /cover etc.

All these modules have been developed based on the concepts discussed above. The model structure illustrated in the chart in Figure 3.24 overviews flows between all these variables: both spatial and non-spatial, static and dynamic. It also represents the sequential processes which run throughout the model. Nevertheless, to make all this much clearer and to show how these ideas can be translated into an operational model, the programming approach will be discussed in the next chapter.

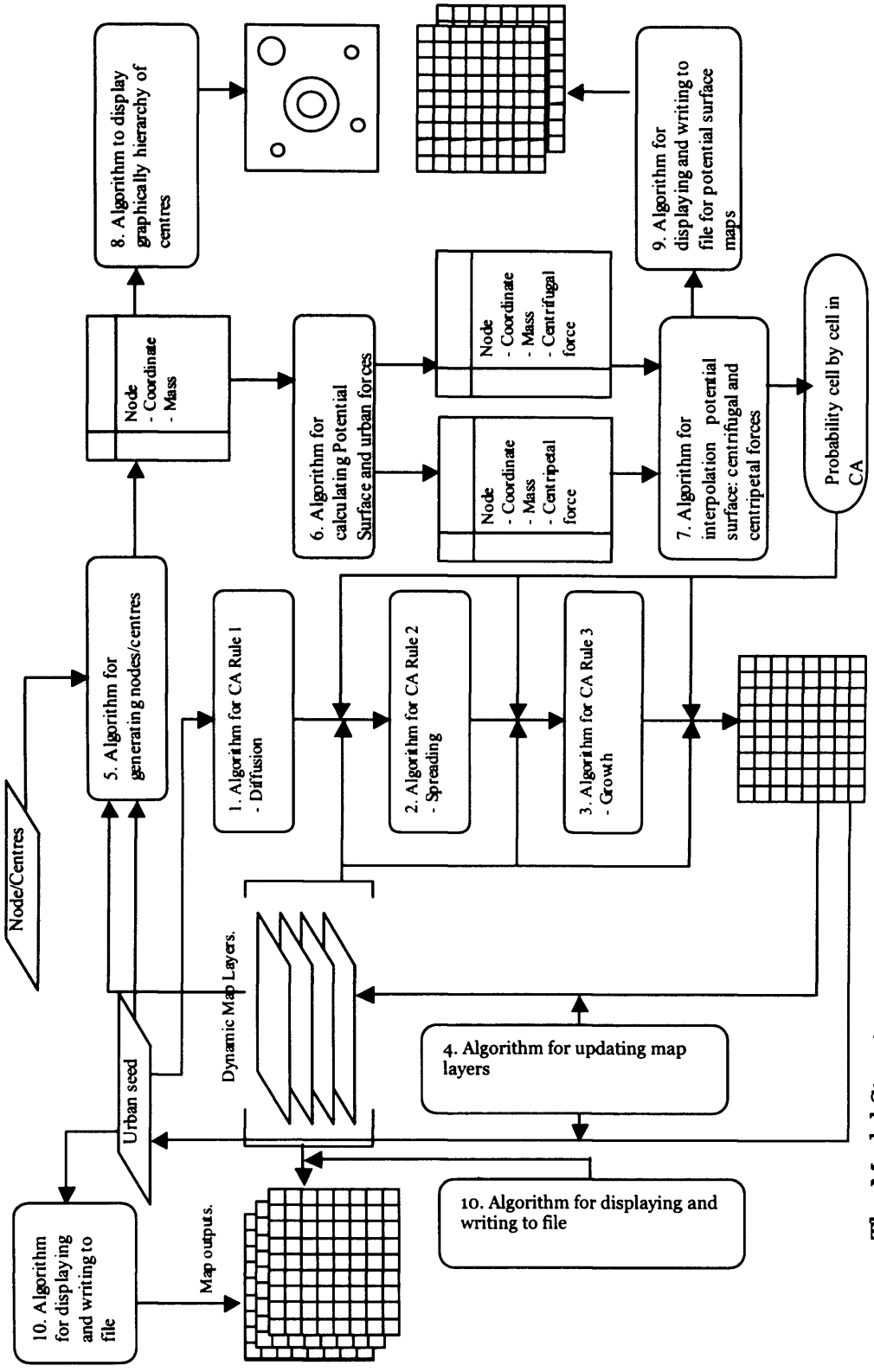


Figure 3.24 The Model Structure

CHAPTER FOUR
Programming the Model

1. Introduction

One of the most important parts of this study is to explain how the DSSM can be implemented as a set of computer programs. To achieve this objective, we need to describe various details of the program so that the reader is aware of how the various parts of the model as discussed in the previous chapters are assembled. The various theories and techniques used cannot be easily demonstrated without creating a computer model for the only way to show how they work together is by running the model as a computer program and analysing and visualising the results. The programming codes described in this chapter ensure that all these concepts are correctly applied to the model. Moreover, the description of this model can also be used to guide other urban modellers who may wish to integrate such spatial techniques within an urban simulation model in future applications.

In the previous chapter, we rigorously specified all the core concepts of the model as well as turning them into an explicit structure. This structure details all these processes as separate algorithms but indicates how the various model variables link to each other in providing flows between the different components. The three main modules based on cellular automata, the map dynamics, and the node generator supported by other elements such as those involving data input were discussed theoretically in the last chapter, and these elements are central to the exposition here where we will deepen the details of how they are programmed.

Thus in this context, we will expand some key portions of the model structure into sets of programmable algorithms. Before we do this however and in the next two sections, we will describe some of the technical approaches used; in particular, the idea of the Object-Oriented Program (OOP), and the Unified Modelling Language (UML) which enables us to

read and interpret the model structure and its algorithms through diagrams and code in later parts of this chapter. Ultimately, the chapter provides all the major details of the class diagrams and their simulated codes so that we can conceptually represent all the main steps of the model processing.

2. The Object-Oriented Programming Approach

In the last 40 years, there have been several programming languages that programmers have been able to choose in developing their software; for instance: Assembly, Pascal, C, C++, Basic and so on. These programming languages can be classified into four types: 1) unstructured programming, 2) procedural programming, 3) modular programming, and 4) object-oriented programming. In some senses, these four types of language represent the chronology of the way programming has developed with object-oriented programming representing the most recent development. It is worth saying a little on each in turn.

1) Unstructured programming: this approach is the least complicated of all programming methods. It is usually based only on a main program that includes all variables or data. The procedure to compile source codes is in linear form. All functions developed inside the main program have then to be copied to any parts of the main program that need these functions to be processed.

2) Procedural programming: The flow of control in procedural programming is still in linear form, but any function created within the main program can be determined as a 'procedure', sometimes called functions or subroutines. While the main program is being compiled, the program is able to call any procedures to calculate data at any time. Compared to unstructured programming, this method makes it easier to find errors in the program. A program written with this method is usually easier to interpret and correct.

3) Modular programming: extending the procedural programming approach, modular programming changes the way some of the procedures perform by combining one or more procedures to make smaller modules within the main program. This enables separation of variables in the program into global and local variables. Each module has one or more procedures and all processes will finish when the program compiles all codes within the module. Then the compiler returns to other code in the main program.

4) Object-Oriented Programming: OOP is an innovative programming language that takes some advantages of both procedural and modular programming. Moreover, OOP has different properties such as inheritance, encapsulation and interface etc. that make the program more efficient than other approaches. Recently, most of the modern applications used in Windows operating system have been developed using the object-oriented programming approach. For this reason, OOP has become the most popular approach for the new generation of programming languages.

The DSSM model has been developed based on the OOP approach, and the rest of this section therefore depicts in detail some of fundamental concepts used in developing an OOP application. We will first develop the general context of the OOP using common examples that are easier to understand and observe in every day life. Subsequently, we will then apply these to the particular perspective used in creating DSSM.

Object

To understand object-oriented programming, objects are the key for our prior consideration. In everyday life, we look around and see many objects: a cat, a table, a computer, a chair, a book, a car etc. All real-world objects can be said to normally have two features: a *state* and *behaviour*. For instance, cats have states: name, colour, hungry; and behaviour: running, jumping. Cars have states (four wheels, number of gears, number of doors, steering-wheel) and behaviour (braking, accelerating, changing gears).

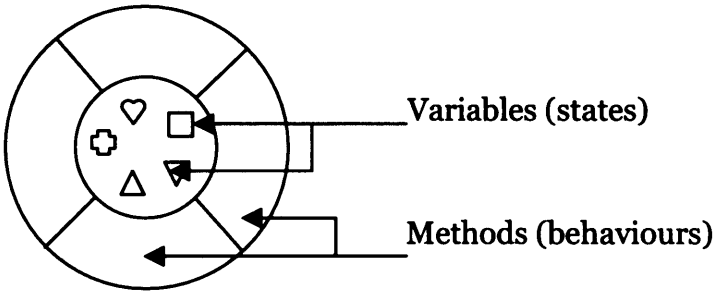


Figure 4.1 Object Diagram

Software objects are modelled using the same concepts as real-world objects in that they too have states and behaviour. A software object displays its state in one or more *variables or attributes.* Like other programming languages, a variable is a piece of data named together with its identification. A software object realizes its behaviour with *methods.* A method is a function related to an object. Figure 4.1 illustrates a graphic representation of such a software object.

Real world objects generally can be represented as software objects. For instance, 'real-world' cats are emulated as software objects in order to create an animation program which can be used to study their activities; 'real-world' cars are imitated as software objects in a program used to simulate a driving lesson, for example.

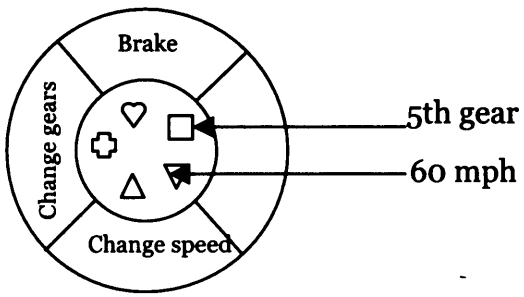


Figure 4.2 A Real-World Car as a Software Object

Figure 4.2 shows how a real-world car can be modelled as a software object. The current state of the car object is identified by its variables: its speed is 60 mph; its current gear is 5th gear. These variables are called

instance variables, because they contain the state for the particular car object. In addition, the software car would also have methods to brake, change the speed and change gears. These methods are also called *instance methods*, which change the state for a particular car instance.

The object diagram above illustrates all the variables within the centre with the methods surrounding these variables in the outer circle segments. In particular, only the object's methods can be accessed though variables at the centre of the circle. This property is called *encapsulation*. This ensures that the variables are unreachable from other objects' methods. The only way to reach and manipulate variables inside is for programmers to prepare methods which are called *accessor* and *mutator methods*. However, it is possible that some objects may often wish to be exposed to some of the variables directly associated with other objects. The OOP has some flexible alternatives to manage this kind of property and these will be discussed later in this chapter.

Class

A class is a blueprint, or prototype, that defines common variables and methods for all objects. Many real-world objects often share the same kind of variables and methods, and these can thus be classified together. For example, as there are many cars in the world, a software car object in the example is called an instance of the class of the object known as 'cars'. Cars have some specific state: four wheels, current gear, one steering wheel etc. Cars also have their own behaviours: brake and change speed etc. Even they share their states, but each car's state is independent.

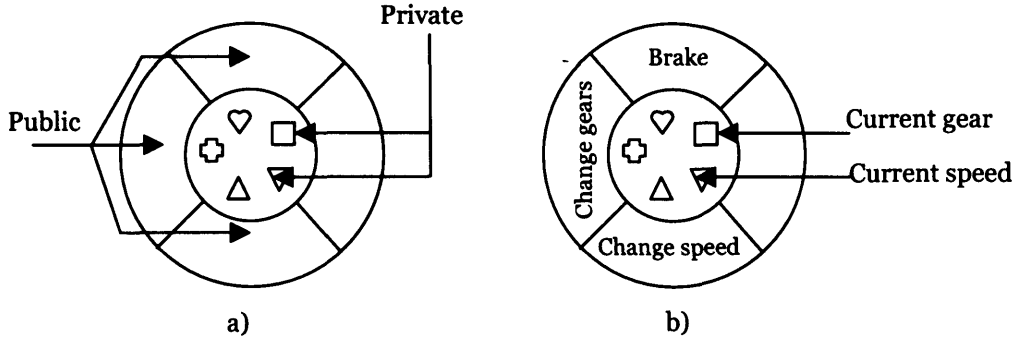


Figure 4.3 Components of a Class and a Car Class Example

As mentioned earlier, the main thing in constructing an OO program is that users need to program classes or blueprints of an object rather than program each object directly. To program car objects for example, the only thing that programmers need to do is to write a blueprint for a car object that shares the same states and behaviours with any other car object. Consequently, in developing the application, programmers are then able to create car objects from the blueprint at any time during the operation of any procedure that needs to act on the car object.

Figure 4.3 represents components of a blueprint of objects. The nucleus of the object diagram in Figure 4.3a illustrates the *private* state; these variables would be encapsulated and they would be controlled by *public* methods. Public methods are an interface to enable contact with other class objects. Figure 4.3b illustrates an example of the car class we have constructed. There are three public methods: braking, changing the speed and changing gears; and two private variables are also set: current gear and current speed.

Any number of car objects from the class can be created. Once an instance of a class is created, the computer system allocates enough memory for the object with all its instance variables. Then the object is created with its own instance variables the same as its blueprint. Cars A and B objects in Figures 4.4 a and b indicate how objects can be created from the class in Figure 4.3.

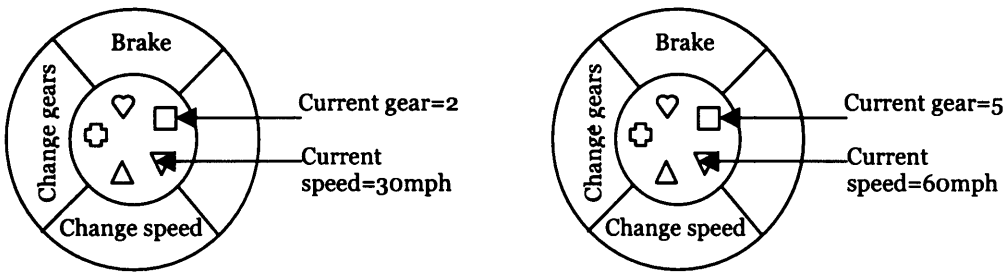


Figure 4.4 Two Objects constructed from the Car Class

Classes can be created in the case of all objects that share the same number of states by defining *class variables* which contain information that is used by all instances of the class. Suppose cars in an application have the maximum speed of 100 mph. Whereas a car object is being created from the class, the maximum speed, 100 mph, should be automatically defined for the car object. In this case, the maximum speed, 100 mph, can be primarily defined as a class variable within the car class. Figure 4.5 below represents an example of a car object with a maximum speed, 100 mph.

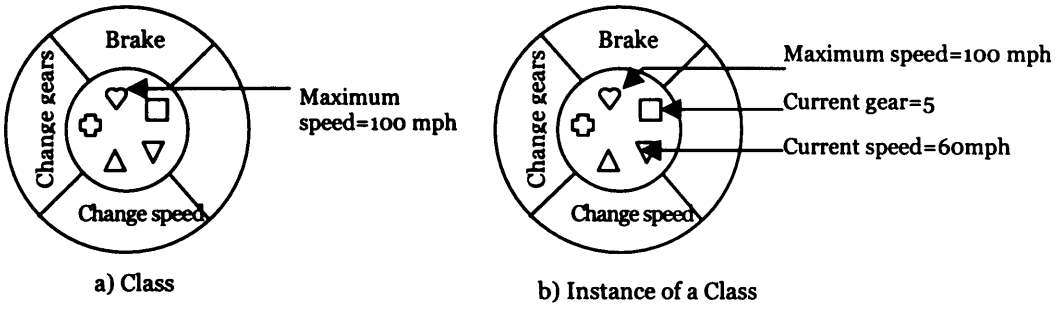


Figure 4.5 An Example of a Class Variable and Its Object

Messages

Most OOP applications cannot be developed using only one object. An object usually is a component of an application that holds many objects inside; and sometimes they need to cooperate with each other. The *message* is the way OOP enables an object to interact with any of the other objects. For instance, object A sends a message (message 3) to object C to achieve some function (see Figure 4.6), object B posts some messages to both object A and C to get some information from the variables within both objects, and so on.

Real-world cars are not movable by themselves, even though they have all the features necessary to move on a street. A car needs a driver to control its steering wheel, change gears, accelerate it and so on. In this case, a driver is another object, and it is able to control the car object. Figure 4.7 represents the situation where a driver is sending a message 'ChangeGear' to the car object when the state, 'currentGear', is in.

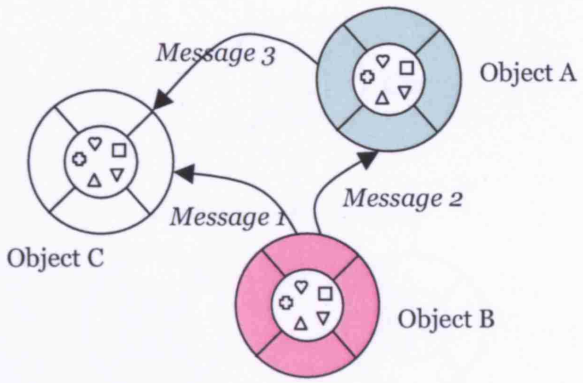


Figure 4.6 Messages sent between Objects

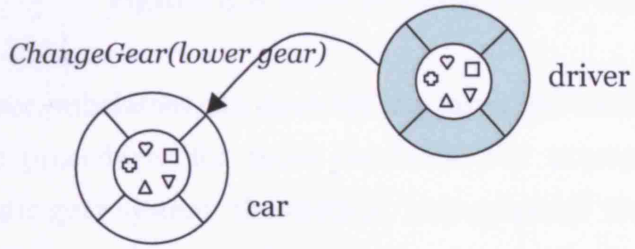


Figure 4.7 An Interaction between a Car and a Driver

Inheritance

OOP has developed this property in order to create new classes from other classes. For example, there are many kinds of cars such as trucks, sedans and vans etc. These cars might be called, using the terminology of object-oriented systems, *subclasses*. Likewise, the car class is the so-called *superclass* for types of cars above. Figure 4.8 below illustrates two classes: truck and sedan are inherited from the generic car class.

Each subclass inherits variables and methods from the superclass. All subclasses share some states: wheels, steering wheel, gears etc.; and they also share some behaviours: braking, changing speed and changing gears and so on.

Nevertheless, even though all subclasses have all variables and methods from their superclass, OOP allows them to add more variables and methods that are different from the other subclasses inherited from the same superclass.

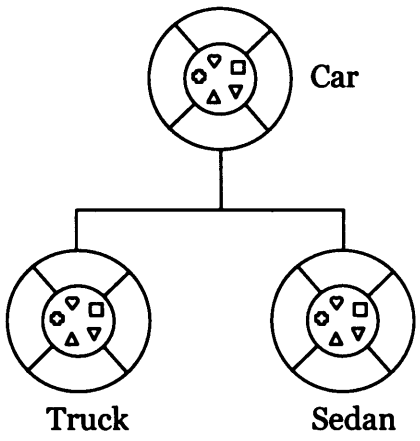


Figure 4.8 Inheritances of the Car Class

Moreover, subclasses can *override* any inherited methods and define more specific procedures for those methods. For example, sedans have the automatic gear system; the method 'change gears' would be overridden in order to properly control the gear system. There is no limitation on inheriting a subclass for more than one *hierarchy*. Levels further down a class are gradually represented by more specialised behaviour.

Many Nodes, Many Objects

When designing an object-oriented program, the main thing is to focus on are the objects. As DSSM has been designed using the OOP concept, then it is worth noting how OOP has been applied in this model and we will thus adapt the same kind of illustrations we have been using for the simulation model. Generally, objects created in this model are not so similar to the car object examples described above; they are more abstract and less tangible. For example, we cannot visibly see what 'nodes' look like or what states and behaviours 'urban' objects should contain. These objects are theoretically designed, based on their properties and how they need to interrelate to other objects. Nodes, for example, are objects in this model. To define a node object, we have to consider what properties that the model needs from its nodes.

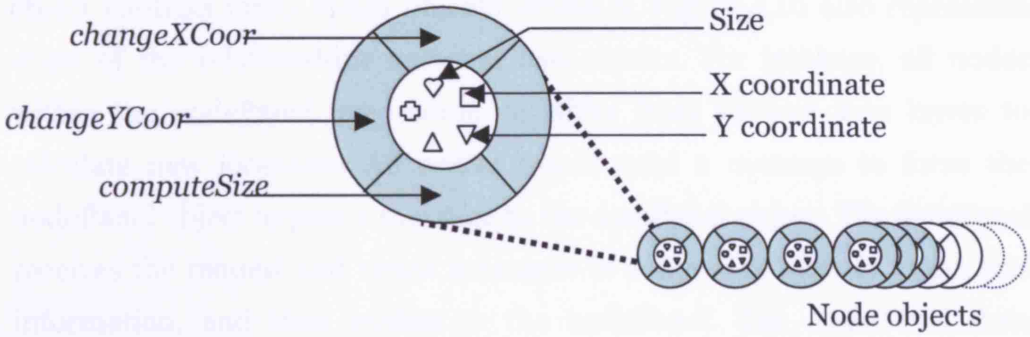


Figure 4.9 Some States and Behaviours of Node Objects

Figure 4.9 represents some instances of a node object. As discussed in the last chapter, nodes can be relocated and resized based on population densities and the arrangements of the urban cells (see Chapter 3 in Node and Hierarchy Generation). Therefore, the node object should have states such as X and Y coordinates and size. Its behaviours for example would involve changing the X and Y coordinates and computing a new size (see Figure 4.9). As nodes need some parameters from other objects such as 'population density' and 'urban cells' to sort out the variables, the node object would then interactively send messages to other objects and receive some parameters back in the process of changing its state.

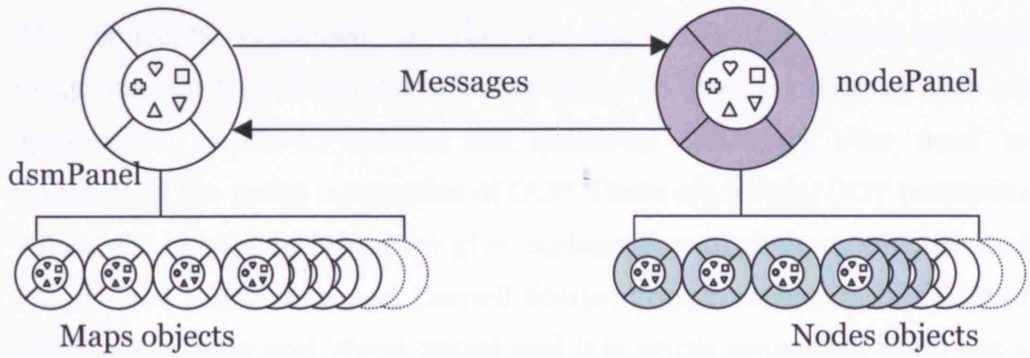


Figure 4.10 Examples of Objects and their Interactions in DSSM

In Figure 4.10, the 'dsmPanel' object contains many map layer objects such as the river, road, LUC maps and so on. On the other hand, the 'nodePanel' object controls many nodes objects within it. Figure 4.10 also represents some of the relationships between two objects. For instance, all nodes within the nodePanel need some variables from various map layers to calculate new locations. All nodes would send a message to force the nodePanel object to post a message to the dsmPanel object. The dsmPanel receives the request and sends messages to some map objects to get this information, and then replies to the nodePanel. The nodePanel then applies the parameters to a set of methods and returns other parameters, sending these as parameters which change the states of all nodes.

Figure 4.10, however, demonstrates only two objects and a single level of interrelationships. Practically in an application, there are many more than two objects which interact together. In case of DSSM, more than twenty classes have been developed; some classes create more than one hundred objects running in the program at all times. Almost every object sends messages to obtain some parameters and receives messages for operating the methods which manipulate their states continuously. It is too intricate to design an application displaying all associations using the object diagram which we have used in Figure 4.10. Thus to cope with this difficulty, another efficient technique called the *Unified Modelling Language* (UML) is employed to illustrate relationships between all classes. The UML approach will be described in the next section.

The concepts explained so far are the basis for object-oriented programming that illustrates a broad sense of how to program such an application. However, this is not sufficient for users who need to understand the entire conception of OOP. There are several OOP textbooks which are helpful resources to give explanations of their more profound details (see Horstmann and Cornell 2005a; 2005b, Barker 2003, Wiener and Pinson 2000 and Wood 2002) and it is worth consulting these but a more comprehensive way of examining the entire conception of an application is to use the UML and this we will explain in the following section.

3. The Unified Modelling Language (UML)

> *"Unified Modelling Language (UML) is a general-purpose notational language for specifying and visualizing complex software, especially large, object-oriented projects". (http://www.webopedia.com/TERM/U/UML.htm 2004)*

For a large application developed with OOP, there are many classes and objects that need to be created. As mentioned previously, UML is a graphical language for assisting programmers to understand a program structure. The main purpose of designing an application with a UML diagram is therefore to communicate structures of OOP-classes with programmers (see an example of UML diagram in Figure 4.11 which is based on the concepts introduced in Chapter 3). Once all classes are designed, programmers can use the diagram to write codes in any OOP-based programming languages. Most of UML software nowadays has the ability to automatically generate programming codes from the UML diagram in several languages such as Visual Basic, C++ and JAVA etc. This makes it essential as well as interesting to learn about the structure of UML before programmers begin to write their code.

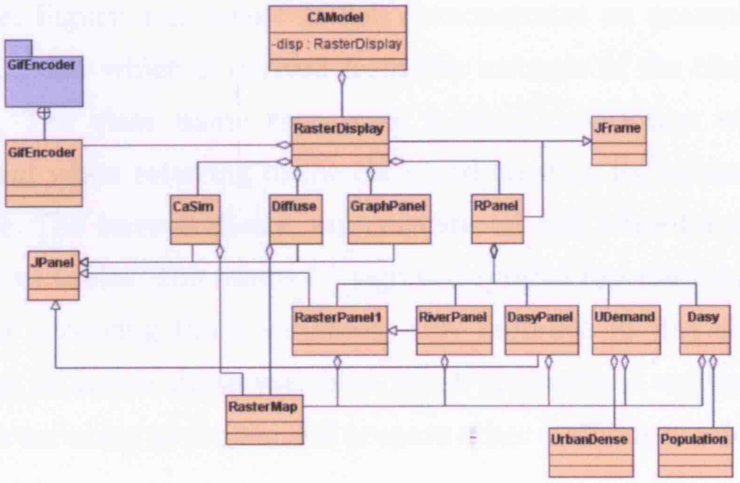


Figure 4.11 An Example of a UML Diagram for DSSM

The structure of the DSSM illustrated as a UML diagram will be discussed in great detail later in this chapter, so that it is valuable to introduce a basic overview of UML concept in this section. The concept will be only partially explained as a limited set of symbols and relationships used to

represent the model will be discussed here. Additional useful contexts about UML can be found in the numerous textbooks that are available (see Booch et al. 1998, Schader and Korthaus 1998, Rumbaugh et al. 1999, Siau and Halpin 2001, and Alhir 2003), and readers are referred to these for more detail.

Class Diagrams

Class diagrams represent the detail of the important information associated with classes. A class diagram is drawn in order to communicate to other users and for convenient coding of the variables and methods in the program. Programmers then need to design the variables and methods that are to be contained in the class, or to at least provide the class with the possibility those such variables and methods might at some point be contained within it.

To draw a class diagram, there are three components that should be considered: 1) Class name, 2) Variables or states and 3) Methods or behaviours. All are important information that programmers would require when writing a program as we illustrated above in the software car example. Figure 4.12 below in fact demonstrates an example of the car class diagram which is derived from the instance of the class in the last section. The class name represents a short description which is very important when referring to the class and creating its instances, **Car**, for example. The **currentGear**, **currentSpeed** and **wheel** are represented as class variables. The minus (-) sign is a symbol representing the *private level* for accessing those variables; only methods in this class have the privilege to access those variables. Most applications written using OOP set class variables as private and prepare other methods to refer to them.

On the other hand, the plus (+) sign defines everything behind the class at the *public level.* Other classes that relate to this class can reach all public attributes and methods. The public level mainly tends to be assigned to class methods. Although, OOP allows programmers to assign variables both in the private or public levels, defining all class variables as public

might lead to difficulties in long run. When a program is growing more complex as it is expanded, the public class variables can become uncontrollable. Instead of accessing and manipulating directly these class variables, it is more efficient to constrain the program so that other parts of the program can use such public methods to make changes to variables of the given class.

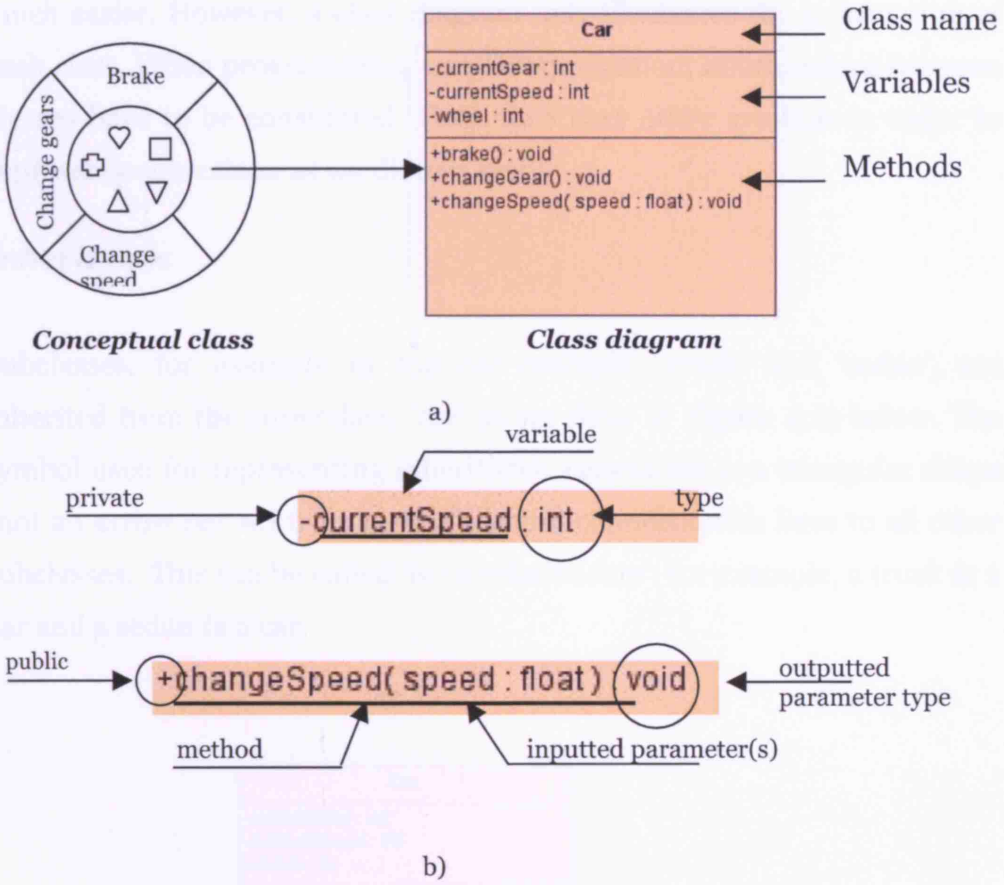


Figure 4.12 An Example of a Class Diagram

The class diagram also represents input and output types of class variables which have the form *integer*, *float*, *double*, *string* and so on. These would be defined according to the actual characteristics of variables; for instance, the variable, currentGear, can be the integer for manual-gear-system and a string type for an automatic-gear-system. The Output parameter type shown in currentSpeed method in Figure 4.12b characterises what type of variable the method would return to the program after processing has been completed. A method can be set to return every primary type of variable

such as integer, float, double and so on; however, in some cases, a method can return variables in the form of an object as well. In the case of Figure 4.12b, the method returns *void* which means the method returns a null or empty value to the program.

Class diagrams clearly represent details of each class and programmers can design many details within a class diagram which makes programming much easier. However, a class diagram only illustrates the information of each class. When programming an OOP application, relationships between classes have to be considered. UML thus uses other symbols in order to represent connections as we discuss below.

Inheritance

Subclasses, for example in the car example, 'truck' and 'sedan', are inherited from the superclass, 'car' as we show in Figure 4.13 below. The symbol used for representing inheritance association is a triangular shape (not an arrow *per se*) from superclass and connects with lines to all other subclasses. This can be called 'is –a relationship': for example, a truck ***is*** a car and a sedan ***is*** a car.

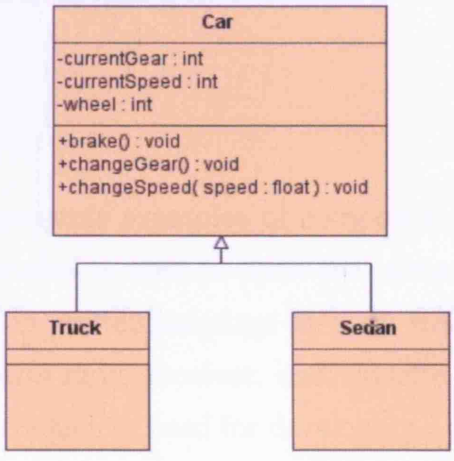


Figure 4.13 Examples of Inheritance Associations

Composition

Composition is called 'has – a relationship'. The composition relationship is used when a class is composed of other classes, for example, a car object is composed of an engine, a stereo, doors etc. At another level, the stereo object is composed of radio receiver and cassette player and so on.

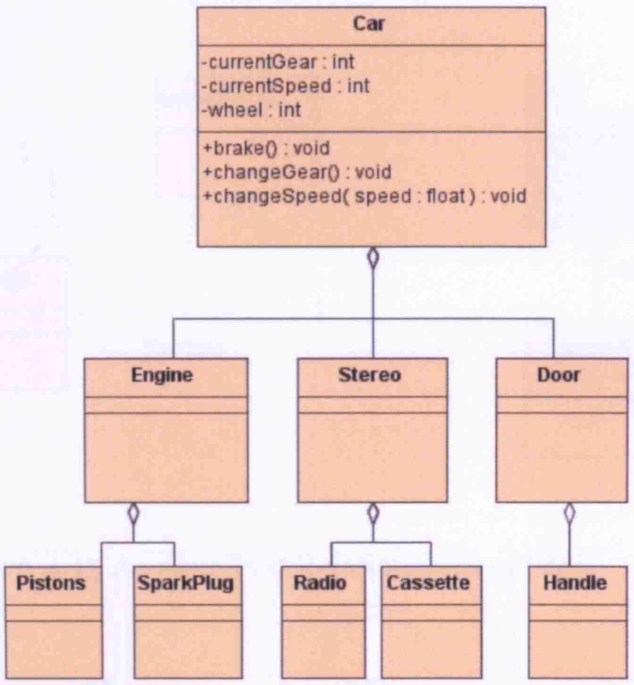


Figure 4.14 Examples of Composition Relationships

Figure 4.14 represents examples of composition of the car object (where the diamond symbol is used to show the composition relation). This may be represented in written language as 'a car ***has*** engine, stereo, and door' and 'a stereo ***has*** radio receiver, and cassette tape player'. This kind of relationship is frequently used for developing a program. All classes can be created independently, but those can be programmed together using this relationship.

In addition, Figure 4.15 illustrates other examples of associations in UML diagrams that will be frequently used to represent elements of the model in this chapter. Interfaces are special notations that have only names of methods but no procedures within. A class that realises an interface needs to rigorously implement all of the methods within the interface. This ensures that the class has all the necessary methods to perform an application.

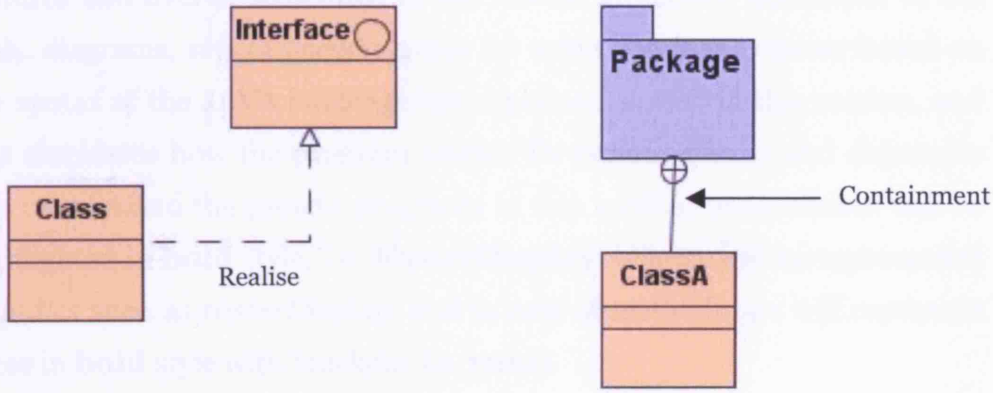


Figure 4.15 Another Two Relationships of UML

Packages are similar to shelves in a library which locate and identify a group of classes. Those classes can be requested from the packages to handle some function in the main application. It is useful to arrange many programming codes within one package, and thus retrieve them for other applications when they are needed for use. Moreover, as many applications have been broadly developed using OOP technology, some of these are available as open source codes or libraries. Programmers can thus conveniently pick some of these libraries in the form of ***packages*** which can be used immediately within their applications.

4. Programming the Model

In this study, the DSSM has been fully developed using the JAVA programming language that is built using the object-oriented programming approach. As discussed in an earlier chapter, the conceptual framework is applied to design the model structure (see Figure 3.24). According to this model structure, this chapter will symbolise some major classes and their relationships as UML diagrams that represent key features and overall structures in the model. Moreover, additional to the UML diagrams, sets of pseudo codes for some dominant classes based on the syntax of the JAVA language are explained as well in this section, and this elucidates how the program works. To explain classes and objects in this context and the pseudo programs in this section, main classes will be highlighted in **bold** style, i.e. **RasterDisplay**, objects will be represented in *italics* such as *rasterDisplay*, and in case of methods, we will represent these in **bold** style with brackets, i.e. **run**().

As explained so far in terms of OOP and UML, those concepts are essential in interpreting the diagram shown in Figure 4.16 below. The diagram clearly represents the overall structure of the model: 21 classes are implemented; 2 program packages, **GifEncoder** and **Jama**, are used; and more than 50 relationships are built within these. The 21 classes which we implement have their own functions which perform in the model as we have summarised in Table 4.1 below. The **GifEncoder** library contains classes that are used to create image files in GIF format; **Jama** is a matrix library that holds some useful classes for performing complicated matrix operations. Some other classes shown in the diagram are the classes in the JAVA original libraries such as **Jframe**, **JPanel**, **Vector**, **Image** and **Graphics** as well as some interfaces: **Runnable**, **MouseListener** and **ActionListener**.

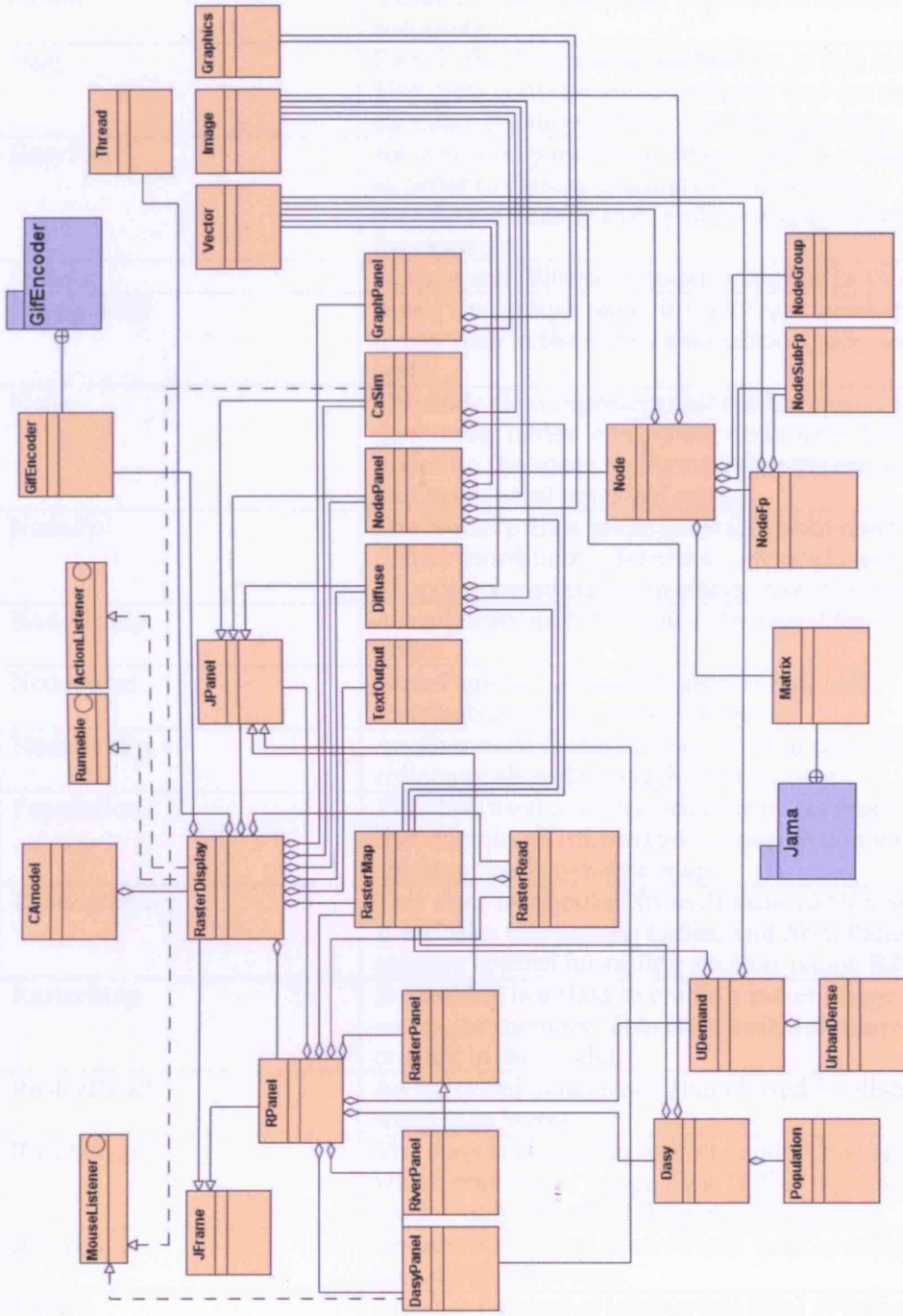


Figure 4.16 A UML Diagram of the Dynamic Settlement Simulation Model (DSSM)

Table 4.1 Classes Description

Class	Description
CAmodel	The main DSSM program
CaSim	CaSim is created in order to perform cellular automata.
Dasy	Dasy is the Dasymetric application in this model. This class contains some methods that are used to calculate the dasymetric population density.
DasyPanel	An inherited panel from JPanel (JAVA swing) used in order to display graduated colour maps: population density map and potential surface map, for example.
Diffuse	Hägerstand diffusion model is applied in this class.
GraphPanel	A real time graph panel is built from numbers of urban cells in the urban simulation space and time state.
Node	The node class represents all modules used for generating nodes, computing node sizes, locating nodes on the space and generating the centrifugal and centripetal potential maps.
NodeFp	The NodeFp class is the node footprint used to collect coordinates, densities, potentials and sizes of nodes according to administrative territories.
NodeGroup	A temporary node footprint class used for grouping nodes.
NodePanel	NodePanel is a JPanel created to represent distributions of nodes on the space.
NodeSubFp	Another node footprint class used in cases of collecting all nodes and their attributes.
Population	The class used as a blueprint for population objects that contain all information of population after creating the dasymetric map.
RasterDisplay	This class is inherited from JFrame (JAVA Swing). It includes two panels: CaSim and NodePanel, and another control for calling another panel: RPanel.
RasterMap	RasterMap is a class to create a raster image in computer memory. This class builds all maps created in the model.
RasterPanel	An inherited panel from JPanel used for displaying some map layers.
RasterRead	The class is created in order to collect methods which read raster image files. In this version of model, only the ASCII format is recognised.
RiverPanel	An inherited panel from JPanel used specifically to display the river map.
Rpanel	Another frame that is inherited from JFrame. All dynamic and static map layers would be shown in this frame.
TextOutput	This class is for writing output parameters to textfiles.
UDemand	UDemand stands for urban land demand. Functions for calculating the demand for urban

Class	Description
	land are constructed here.
UrbanDense	Attributes required by UDemand class are obtained in this class.

The Main Program

In a JAVA application, the main program is central and is required to link users to all modules in the program. The main program connects to certain classes which chain to all other classes in the application. The **CAmodel** is a class containing the main program within, **main**(**String** args[]), (see Figure 4.17 below). **CAmodel** class creates a raster map object named *oRaster* and passes it to the **RasterDisplay** class as a parameter in order to perform various methods. Notably, the main class, **CAmodel**, has not that many attributes and methods within it; it contains certain necessary attributes and the main module which runs the program. All other processes are passed to other classes in the model according to the relationships depicted in Figure 4.16 above. Some of these important processes will be discussed in this section.

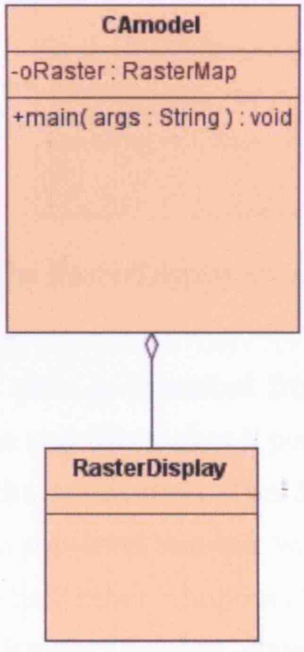


Figure 4.17 Associated Classes for the Main Program

In the process of constructing the **CAmodel** class, a new *rasterDisplay* object is created. Considering the classes in Figure 4.16, the relationships among the **RasterDisplay** class and others are complicated. **RasterDisplay** is conceptually designed for a variety of tasks which we can list as 1) to pass and retrieve variables to and from other associated classes; 2) to drive dynamically all processes though time; 3) to display two important maps: the hierarchical node map and urban structure map; 4) to control the simulation process; 5) to save maps to image files; and 6) to write parameters required to text files over time step. **RasterDisplay** is therefore really the main controller for the DSSM.

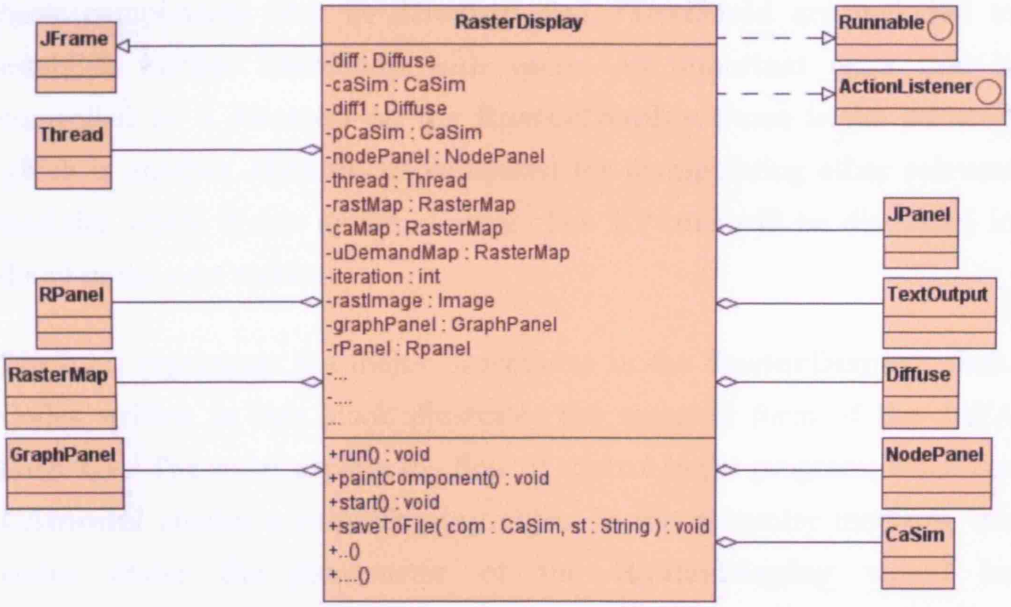


Figure 4.18 The RasterDisplay Class and Its Relationships

The **RasterDisplay** class is inherited from a class in the JAVA swing library called **JFrame** and this makes it possible to develop a graphic user interface (GUI) for the application. The **JFrame** is inherited from the **Frame** class that is a top-level window with a title and a border. This is the main window to which other components can be added later on. Figure 4.18 illustrates the **RasterDisplay** class diagram which shows some crucial attributes and methods and its major connections. It is worth noting that there are so many details in terms of the programming that it is

not possible to represent and describe all attributes and methods here but those that are shown are the main ones.

The **RasterDisplay** realises two interfaces: **Runnable** and **ActionListener**. The **Runnable** interface concerns dynamic processes in the model. The **ActionListener** involves interactions with the GUI; users are able to instruct or control the graphical interface inside the program. On the **RasterDisplay** frame, there are three panels which are inherited from **JPanel**: **CaSim**, **NodePanel** and **GraphPanel**. Moreover, another panel component is processed in the hidden background: **Diffuse**. In fact these classes will be discussed in more detail a little later. Finally, other basic components such as **JButton** and **JTextField** are included to establish further interaction with users. An important class that is controlled by a **JButton** on the **RasterDisplay** frame is the **RPanel** which is another **JFrame** implemented for manipulating other relevant modules which create dynamic maps. The **RPanel** will be discussed in detail in the next section.

Block 4.1 represents the major procedures in the **RasterDisplay** class. Codes written in this block illustrates the essential form of the JAVA language. The codes explain the flow of control in the program; when the **CAmodel** creates a *rasterDisplay* object in the computer memory, the codes under the constructor of the **RasterDisplay** would be automatically performed. After compiling and executing the program, a main window that composes control-buttons, a text field and two panels, will appear. All necessary class variables are declared and many of the program's objects are then created in subsequent operations.

Important objects include *diffuse, caSim,-rPanel, nodePanel, graphPanel* and other controls such as *buttons, textField.* Once all these objects have been prepared, other methods in the model are read for the various procedures. An important method in **RasterDisplay** that needs to be explained is **run**(). The **run**() method links to the dynamic processes of the **Thread** class in the JAVA library. The **Thread** class is used practically for controlling time in the simulation. When a *thread* object is created and

started, the **run()** method in the **RasterDisplay** will begin to perform by first passing some messages to other associated objects: *diffuse*, *caSim* and *nodePanel* which need to be executed within their appropriate modules. The process will not stop until the *thread* object is paused. These objects as mentioned above are the major drivers of this simulation; all of these will be discussed in more detail in the next stage.

Additionally, a *graphPanel* object is constructed and some methods called from **RasterDisplay** to draw real time line graphs showing the number of urban cells in the space. Finally, image and text file writers are also implemented here. A variable, *timeStamp*, can be defined which synchronises with *iterations*, the image and text files being written according to the value of this *timeStamp*.

```
Class RasterDisplay extends JFrame implements Runnable,
ActionListener
{
    Declare all necessary class variables from:
    RasterMaps, CaSim, Diffuse, NodePanel, etc.


    RasterDisplay(RasterMap rasterMap) //---The Constructor
    {
        create a diffuse object from Diffuse(rasterMap);
        create a caSim object from CaSim();
        create a nodePanel object from NodePanel();
        create a rPanel object from RPanel();
        create button objects from JButton(): run, stop, clear, reset, seed, layer,
        update, saveImage, saveText and genNode;
        create a textField object from JtextField();
        create graphPanel object from GraphPanel();
        create thread object from Tread();
        add all panels and components to the RasterDisplay;
        display the window;
    }

    /*Prior to run the program, it is necessarily to calls rPanel and set all map
    layers for gathering all map parameters.*/
    void run() //when users click 'run' button
    {
        while(status == true){ /*when users click button 'run', status is set as
                                    true*/
            diffuse.run(iteration); //operate method run(int iteration) in~
                                        Diffuse class
            Gathering all maps from rPanel object;
            caSim.caRun(variables from rPanel, iteration);
```

```
                    //operate method caRun() in CaSim class
                nodePanel.update(node variable from rPanel);
                    /*when rPanel is ready nodePanel gathers node variable~
                      from rPanel to update the space and click button~
                      'genNode' */
                graphPanel.drawGraph(liveCell);
                    /*liveCell is number of urban cells gathered from CaSim*/
                Define a timeStamp to write image and text files.
                Show iteration in textField;
                if (iteration == timeStamp) then write image and text files;
        }//--end of while loop
    }
}//--end of class
```

Block 4.1 Pseudo Codes for RasterDisplay Class

Classes and Algorithms for Diffusion

The diffusion process is a module in the model which is used for defining groups of cells which determine the urban space (see details in Chapter 3). The **Diffuse** panel is inherited from **JPanel**, but its processes are only run in the background of the model. Some *rasterMap* objects are created in this class as input and temporary maps. The **Vector** class implements an extendable array of objects. Like an array, it contains components that can be accessed using an integer index. In this case, a vector object is created for collecting *randomSeed* objects created from the **RandomSeed** class; the *randomSeed* objects contain x and y locations and *periods* of diffusion. Each iteration, the random generator module will generate a random number to define numbers of seeds in the diffusion space. The other three random numbers: x, y and *period*, are determined to create objects (see **RandomSeed** class diagram in Figure 4.19) according to the numbers of seeds defined. Objects are created and collected in the vector object. When the *rasterDisplay* object calls *diffuse*.**run**() (see the codes in Block 4.1), all parameters in the vector are retrieved using the relevant accessor methods **getX**(), **getY**() and **getPeriod**(); at this point, cells in the space begin to diffuse.

Block 4.2 conceptually represents sets of important code for the **Diffuse** class. The major method, **run**(), is used for processing the spatial diffusion according to the Mean Information Field (MIF) concept. When the *rasterDisplay* has created *diffuse* object, the constructor of **Diffuse** class

will create several raster maps: *oRaster*, *nRaster*, *rRaster*, *ageRaster* and *iterTemp* and a vector object, *seedVec*, in memory. While the *diffuse*.**run**() in *rasterDisplay* is being processed, the program will process something similar to the function **run**() in Block 4.2 below.

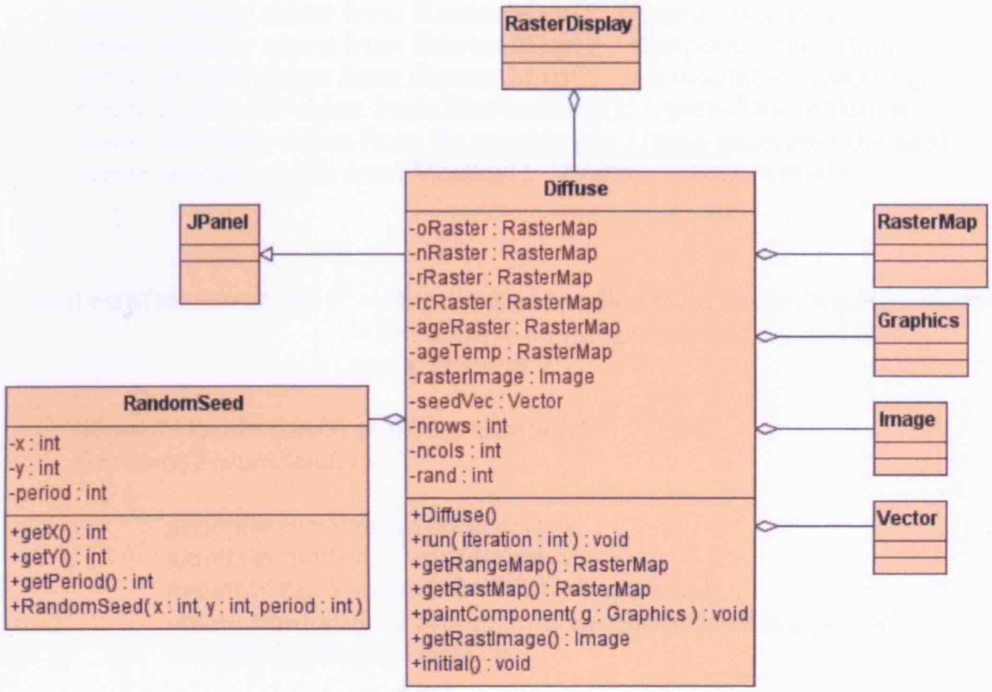


Figure 4.19 The Diffuse Class Diagram and Its Relationships

In terms of Hägerstand's Mean Information Field (MIF) and the additional idea of a diffusing period (see details of the diffusion concept in Chapter 3), the algorithms below enable seeds to be randomly defined on the MIF and how the seeds are controlled in dispersal according to the dynamics of the diffusion period parameters. The *rRaster* object is an outcome generated after processing **run**() in each iteration state. The *rRaster* is a *rasterMap* object which contains cells with different intensities. Values of these cells are used to calculate probabilities of urbanised cells in the next stage.

```
Class Diffuse extends JPanel
{
    Declare all necessary class variables from:
    RasterMaps, Image, Vector
    int nrows, ncols, ran;

    Diffuse() //---The constructor
    {
        create oRaster object from RasterMap(); //input rastermap
        create nRaster object from RasterMap(); //temporary rastermap
        create rRaster object from RasterMap(); //accumulated rastermap
        create ageRaster object from RasterMap(); //period for diffusion
        create iterTemp object from RasterMap(); //time generated the seed
        create seedVec object from Vector(); //Vector to collect seeds

    }

    void run(int iteration) /* -- this method is called from rasterDisplay object~
                               -- iteration is an integer number passed from~
                               rasterDisplay*/
    {
        generate random seed number, numSeed;
        for (i=0; i<numSeed; i++)
            {
                generate random X coordinate, x;
                generate random Y coordinate, y;
                generate random period of diffusing, period;
                create randomSeed object from RandomSeed(x,y,period);
                add randomSeed to seedVec;
            }
        define oRaster with 1 according to x and y coordinates from seedVec;
        define all seeds' period from seedVec to ageRaster;
        remove all objects in seedVec;
        if (oRaster(i,j) == 1) then define current iterations to iterTemp (i,j) = ~
        iteration; /* i and j are cell in row i and column j */
        //--Next is a loop for defining diffusion cells
        for(i=0; i<nrows; i++){
                for(j=0;j<ncols;j++){
                        if (oRaster (i,j) == 1) then{
                                generate random number range from 0-9999 to ~
                                ran;
                                if (ran belongs to which cell in MIF) then{
                                        //see Chapter 3
                                        set attribute nRaster (MIF cell) = 1;
                                        set attribute ageRaster (MIF cell) = ~
                                        ageRaster(i,j);
                                        set attribute iterTemp (MIF cell) = ~
                                        iterTemp(i,j);
                                }
                        }
                }
            }
        //--Next is a loop for accumulating diffused intensity
        for(i=0; i<nrows; i++){
```

```
                for(j=0; j<ncols; j++){
                if ((iterTemp(i,j)>0)&&(iteration–
                iterTemp(i,j)<=ageRaster(i,j))) then{
                        if (nRaster(i,j)==oRaster(i,j)==1) then{
                                rRaster(i,j) = rRaster(i,j) + 1; //--maximum = 10;
                        }
                }
                }
            }
        replace oRaster with nRaster;
    } //--end of run()
} //-- end of class
```

Block 4.2 Pseudo Codes for Diffuse Class

Classes and Algorithms for Cellular Automata

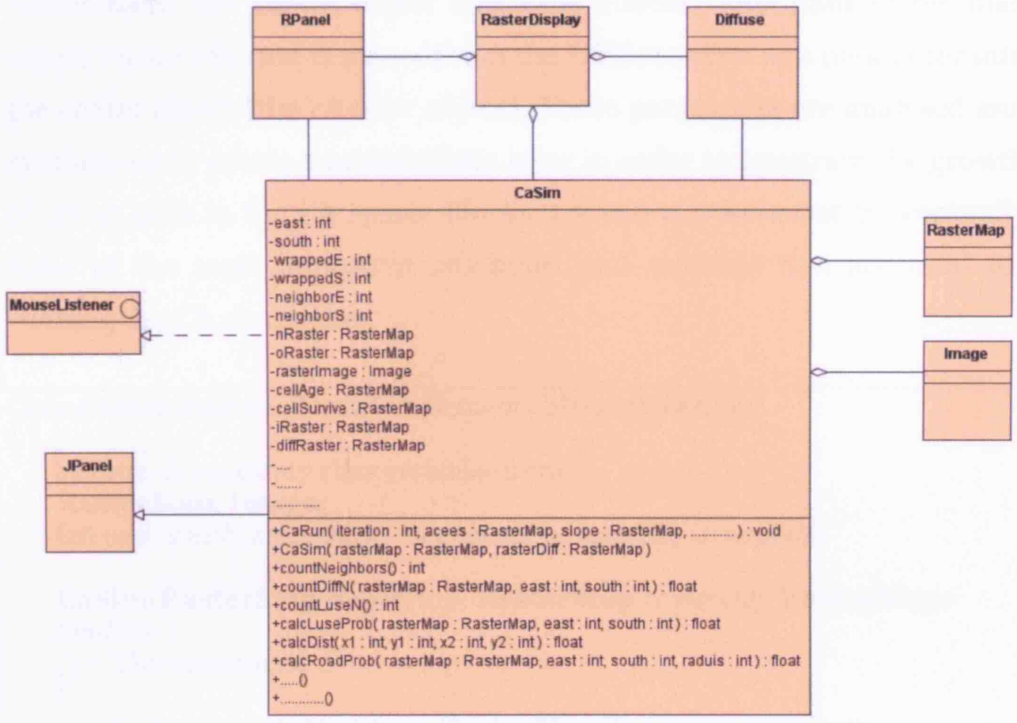


Figure 4.20 The CaSim Class and Its Relationships

As mentioned in the last chapter, results from the diffusion are used to derive the probabilities for generating urbanised cells in the CA space. The **CaSim** class is relevant to this issue. The **CaSim** is a **JPanel** which is based on the cellular automata concept so that the program can obtain all the necessary modules for urban simulation in cellular automata fashion which are able to display changes in the urban space dynamically. Figure

4.20 represents a number of important attributes and methods used in the **CaSim** class. As shown in Figure 4.20, the **RPanel** and the **Diffuse** classes are both aggregated from the **RasterDisplay**, with those classes also associating with the **CaSim**. This shows that there are several objects from both classes involved in the **CaSim**. Additionally, **CaSim** class implements an abstract class, **MouseListener**, in order to interact with the graphical interface; users are therefore able to count urban cells in the CA space for example, using this class.

CaSim class relates to many objects in the program as seen in the codes in Block 4.1 and Figure 4.20, while the parameters gathered from the *rPanel* are required to run the *caSim* object (created from **CaSim** class). Additionally, the *caSim* object also calls another important raster map object, *rasterDiff* that is passed from the **Diffuse** class as a parameter into the *caSim* object (the *rRaster* object). These parameters are analysed and synthesised to create a probabilistic layer in order to constrain the growth of urban cells in the CA space. Blocks 4.3 and 4.4 represent conceptually some of the most important attributes and methods that are used for running the CA simulation.

```
Class CaSim extends JPanel implements MouseListener
{
    Declare all necessary class variables from:
    RasterMap, Image;
    int east, south, neighborE, nieghborS, wrappedE, wrappedS;

    CaSim(RasterMap rasterMap, RasterMap rasterDiff, RasterMap~
    landUse )
    //---The constructor
    {
        create oRaster object from RasterMap(); //input rastermap
        create nRaster object from RasterMap(); //input rastermap
        create lRaster object from RasterMap(); //temporary rastermap
        create lacRaster object from RasterMap(); //accumulated rastermap
        create ageRaster object from RasterMap(); //period for diffusion
        create survRaster object from RasterMap(); //time generated the seed
        create lUseTemp object from RasterMap(); //Temporary landuse
        oRaster = rasterMap; initialise default state of oRaster
        nRaster = oRaster; initialise nRaster as oRaster
        lUseTemp = landuse;

    }
```

```
int countNeighbors (RasterMap rasterMap, int e, int s)
{
    int neighbors = 0;
    int nE, nS;
    int wE, wS;

    for(nE=(e-1); nE<(e+1); nE++){
            wE = nE; //in each row, collecting neighbouring cell to wE
            for(nS=(s-1); nS<(s+1); nS++){
                    wS = nS; // in each column, collecting neighbouring cell ~
                            to wS
                    if (oRaster(wE, wS) != 0) then neighbors = neighbors + 1;
            }
    }
    /*--The algorithm to count neighbouring cells also considers the edge of
    raster map. Default for this program set cells that  are out of the raster
    map boundary as zero. */
    return neighbors;
}

float countDiffN (RasterMap rasterMap, int e, int s)
{
    float diffNeighbors = 0;
    int nE, nS;
    int wE, wS;

    for(nE=(e-1); nE<(e+1); nE++){
            wE = nE; //in each row, collecting neighbouring cell to wE
            for(nS=(s-1); nS<(s+1); nS++){
                    wS = nS; // in each column, collecting neighbouring cell~
                            to wS
                    diffNeighbors = diffNeighbor + rasterMap(wE,wS);
            }
    }
    /*--The algorithm to count neighbouring cells also considers the edge of
    raster map. Default for this program set cells that are out of the raster
    map boundary as zero. */
    return diffNeighbors;
}

float countLuseN (RasterMap rasterMap, int e, int s)
{
    int lUseMajor = 0;
    int nE, nS;
    int wE, wS;
    int urbanCell, agriCell, forestCell = 0;

    for(nE=(e-1); nE<(e+1); nE++){
            wE = nE; //in each row, collecting neighbouring cell to wE
            for(nS=(s-1); nS<(s+1); nS++){
                    wS = nS; // in each column, collecting neighbouring cell~
                            to wS
                    if (rasterMap(wE, wS) == urban) then urbanCell++;
                    else if (rasterMap(wE, wS) == agriculture) then
                    agriCell++;
```

```
                    else if (rasterMap(wE, wS) == forest) then forsetCell++;
            }
        }
        /*--The algorithm to count neighbouring cells also considers the edge of
        raster map. Default for this program set cells that are out of the raster
        map boundary as zero. */

        lUseMajor = Majority of land use /cover in neighbouring cells
        //--urbanCell – agriCell – forestCell : 1 – 2 – 3 respectively.

        return lUseMajor;
    }

    float calcLuseProb (RasterMap rasterMap, int e, int s)
    {
        float lUseProb = 0;
        int nE, nS;
        int wE, wS;
        int urbanCell, agriCell, forestCell = 0;
        int urbanW = 0.5; int agriW = 0.4; int forestW = 0.1;
        int maxP = urbanW * 9; int minP = forestW * 9;
        // weights for land use / cover defined.

        for(nE=(e-1); nE<(e+1); nE++){
                wE = nE; //in each row, collecting neighbouring cell to wE
                for(nS=(s-1); nS<(s+1); nS++){
                        wS = nS; // in each column, collecting neighbouring cell~
                                to wS
                        if (rasterMap(wE, wS) == urban) then urbanCell++;
                        else if (rasterMap(wE, wS) == agriculture) then
                        agriCell++;
                        else if (rasterMap(wE, wS) == forest) then forsetCell++;
                }
        }
        /*--The algorithm to count neighbouring cells also considers the edge of
        raster map. Default for this program set cells that are out of the raster
        map boundary as zero. */

        lUseProb =
        ((urbanCell*urbanW)+(agriCell*agriW)+(forestCell*forestW))-minP~
        /(maxP - minP);

        return lUseProb;
    }

    float calcRoadProb (RasterMap rasterMap, int e, int s, int radius)
    {
        float roadProb = 0;
        int sum = 0;
        int roadCell = 0;

        //--Loop for counting possible number of cells in a search radius
        for(int nE=(e-radius); nE<(e+radius); nE++){
                wE = nE; //in each row, collecting neighbouring cell to wE
                for(int nS=(s-radius); nS<(s+radius); nS++){
```

```
                    wS = nS; // in each column, collecting neighbouring ~
                                    cell to wS
                    sum++;
                }
        }

        //--Loop for counting number of road cells in a search radius
        for(int nE=(e-radius); nE<(e+radius); nE++){
                int wE = nE; //in each row, collecting neighbouring cell to wE
                for(int nS=(s-radius); nS<(s+radius); nS++){
                        wS = nS; // in each column, collecting neighbouring ~
                                        cell to wS
                        if (rasterMap(wE, wS) == 1) then
                                roadCell++;
                }
        }
        /*--The algorithm to count cells in search radius also consider the edge of
        raster map. Default for this program does not count cells that are out of
        the raster map boundary.*/

        roadProb = roadCell / sum;
        return roadProb;
    }

    void caRun(.....){
    .....
    }// see Block 4.4 in detail
}//--end of class
```

Block 4.3 Pseudo Codes for the CaSim Class

Block 4.3 above illustrates how the **CaSim** class can be implemented. In the class constructor, several objects, especially the number of *rasterMaps*, are created for the simulation processes. All other methods appearing in Block 4.3 are implemented to perform some calculations between cells and their neighbours. The **countNeighbors**() method, for example, requires certain parameters such as those from a *rasterMap* object, which are numbers representing the position of a central cell (*e* and *s* in this case). The **calcRoadProb**() is another example of the neighbourhood function using a search radius instead of the 8-neighbours so that probability levels of the cells based on the existence of existing roads can be computed.

These functions are executed within the most important module of this class, **caRun**(), as explained in Block 4.4. The **caRun**() calls all of these

methods to count or calculate neighbouring cells by passing coordinates of the central cell; these functions return various parameters and all of these are then used for computation in the next process of the **caRun**().

```
void caRun(int iteration, all variables gathering from rPanel){
    /*--Variables or objects gathering from rPanel are: accessibility,
    rivermask, slope, uDemand, pSurface, uForce--*/
    int radius = 300; //--search radius in meters scale – adjustable.

    /*--Define weights for probability for each layer—These weights are
    adjustable – ranges are from 0 – 10: summation of weight = 10 --*/
    float landUseW = 0.75;
    float petalFW = 2;
    float fugalFW = 2;
    float  roadProbW = 3;
    float slopeW = 2;
    float uDemandW = 0.25;
    for(i=0; i<nrows; i++){
        for(j=0; j<ncols; j++){
            int neighbors = countNeighbors(oRaster,i,j);
            int lUseMajor = countLuseN(landuse,i,j);
            float diffNeighbors = countDiffN(rasterDiff,i,j);
            float diffProb = rescaling diffNeighbors to range 0-1;
            float lUseProb = calcLuseProb(landUse,i,j);
            float roadProb = calcRoadProb(accessibility,i,j,radius);
            float mask = riverMask(i,j);
            float petalF = pSurface(i,j);
            float fugalF = uForce(i,j);
            float slopeP = rescaling slope to range 0-1;
            float uDemand =  rescaling uDemand to range 0-1;
            //--constrained probability--
            float const = ((lUseProb*landUseW)+(petalF*petalFW)~
            +(fugalF*fugalFW)+(roadProb*roadProbW)+~
            (slope*slopeW)+ (uDemand*uDemandW)) / 10;
            float luChangeT1 = 0.7;// adjustable threshold for ~
            landuse  change 1
            float luChangeT2 = 0.9; //adjustable threshold for ~
            landuse change 2
            float caThreshold1 = 0.6;//adjustable CA threshold 1
            float caThreshold2 = 0.6;//adjustable CA threshold 2
            float caThreshold3 = 0.95;//adjustable CA threshold 3
            float caThreshold4 = 0.95;//adjustable CA threshold 4
            float diffThreshold = 0.5;//adjustable Diffuse threshold
            int nbThreshold1 = 4; //adjustable threshold for ~
                                        neighbouring  cells
            int nbThreshold2 = 4; //adjustable threshold for ~
                                        neighbouring  cells
            int timeThreshold1 = 4; //adjustable threshold for time
            int timeThreshold2 = 4; //adjustable threshold for time
            float acThreshold = 0.9; //adjustable accessibility~
                                            threshold

            //--Land use change---
```

```
if (landUse(i,j) == 3) then {// number 3 = forest cell
        if ((lUseMajor ==1)&&(random > luChangeT1))
        then
                lUseTemp(i,j) = 2; //number 2 = ~
                agriculture cell
        else if ((lUseMajor ==2)&&(random > ~
        luChangeT2)) then
                lUseTemp(i,j) = 2;
         else
                lUseTemp(i,j) = 3;
}

//--CA simulation begins here
/*--all urban cells are not developed on masking area
 -- Then, all conditions after this will consider each
cell belongs to riverMask*/
switch (oRaster(i,j)){
        case (0){ //--empty cells on CA space
        if (const >= caThreshold1)then{
                if (diffProb >= diffThreshold) then{
                        nRaster(i,j) = 1;
                        ageRaster(i,j) = iteration;}
                if(neighbors >2) then{
                        if (((neighbors-1)/8)<random<~
                        neighbors/8)) then{
                                nRaster(i,j) = 1;
                                ageRaster(i,j) = iteration;}
        }
        break;
        }
        case (1){ //--developing cells on CA space
        if (const >= caThreshold2)then{
                if(neighbors <= nbThreshold)then{
                if((survRaster(i,j)!=1)&&(iteration- ~
                ageRaster)>0)then{
                        if(random<=(1 / iteration- ~
                        ageRaster(i,j)))then
                                nRaster(i,j) = 1;
                        else {
                                 nRaster(i,j) = 0;
                                ageRaster(i,j) = 0;
                                 survRaster(i,j) = 0;}
                }
                if((random>caThreshold3)||(iteration- ~
                ageRaster(i,j)>timeThreshold1)||~
                (roadProb(i,j)> acThreshold)) then{
                        nRaster(i,j) = 10; //highest level of~
                                                cells
                        survRaster(i,j) = 1; //cell survive}
                }
        }
        break;
        }
        case (10){
        if(neighbors <= nbThreshold2)then{
```

```
                                    if((survRaster(i,j)!=1)&&(iteration- ~
                                    ageRaster)>0)then{
                                        if(random<=(1 / iteration- ~
                                        ageRaster(i,j)))then
                                            nRaster(i,j) = 1;
                                        else {
                                            nRaster(i,j) = 0;
                                            ageRaster(i,j) = 0;
                                            survRaster(i,j) = 0;}
                                    }
                                    if((random>caThreshold4)||(iteration- ~
                                    ageRaster(i,j)>timeThreshold2) then{
                                        nRaster(i,j) = 10; //highest level of~
                                                           cells
                                        survRaster(i,j) = 1; //cell survive}
                                    }
                                }
                                break;
                                }
            }//end for j
        }//end for i
        for(i=0; i<nrows; i++){
            for(j=0;j<ncols;j++){
                if((nRaster(i,j) == 1)&&(nRaster(i,j)==oRaster(i,j))) ~
                then
                    lRaster(i,j)++; //--maximum = 10 levels
                else if(nRaster(i,j) == 10)&&(nRaster(i,j)==oRaster(i,j)))
                then
                    lRaster(i,j) = 10;
                else if(nRaster(i,j) == 0) then
                    lRaster(i,j)--; //--minimum = 0
            }//--end for j
        }//--end for i

        - rescale lRaster to lacRaster to display
        - landuse = lUseTemp;
        - oRaster = nRaster;
    }
```

Block 4.4 The caRun() Function in the CaSim Class

The **caRun**() method in Block 4.4 is implemented to deal with the CA simulation. The *rPanel* object passes a set of encapsulated attributes to **caRun**(). All are raster map layers, and are significant in calculating the probabilities that constrain the growth process in the CA simulation. The raster maps include *accessibility, rivermask, slope, uDemand, pSurface* and *uForce*. **caRun**() directly retrieves parameters from the map layers and requests certain neighbouring functions to manipulate some of these attributes: **countNeighbors**() and **calcRoadProb**(), for example. A set of codes in the block also represents definable weights in each map layer.

This is similar to multi-criteria spatial analysis as in traditional GIS as these layers are superimposed using their own weights, and thus standardised into the probability layer, *const*, ranging from *zero* to *one*.

For implementation, several adjustable thresholds *caThreshold1*, *2* and *3*, *diffThreshold*, *luChangeT1* and *2* etc. are predefined in the program. All these are controllers which are used to set the thresholds associated with the generated random numbers (the term *random* in the block). Focusing on the set of codes that show the operation of CA in the block, the processes do not only depend on the value of *const*, but depend on the values of the random numbers which are generated and their thresholds.

There are three states for the cells represented in the code which conform to the conceptual framework which we outlined in the last chapter: dead cells, developing cells, and developed cells (**case** 0, 1, 10 respectively). These cells can make transitions between any of these states. The processes behind these mutations are defined as: *birth*, *develop*, *survive*, *decline* and *die*. These are the sorts of functions that are associated with models that exist in the field of *artificial life*. Cells then take on these various states within the diffusion process. They develop and decline in stochastic fashion but are also constrained by the land suitability of each cell. If cells survive during this process, this does not imply that they cannot die in future iterations (see the program code in Block 4.4).

Classes for Managing Map Layers

In this section, we will discuss the **RPanel** class in more detail. From Figure 4.21 below, this class is a window within the model that is constructed to display all the relevant map layers and it enables manipulation in terms of the input and output of those maps in other parts of the program. This class is inherited from the **JFrame**; it belongs to the **RasterDisplay** and is associated with the **CaSim** class which passes the relevant variables which create constrained map layers. The **RPanel** also realises the **ActionListener** class in order to ensure that all the user interface control methods are implemented.

There are three main tasks of this class that will be explained. These are: 1) to display maps in graphical fashion, 2) to manipulate the map layers and 3) to construct various map layers that need to be analysed by their integration with certain input map layers.

Firstly for the map display, the **RPanel** contains other classes that inherited from the **JPanel**: **RasterPanel**, **DasyPanel** and **RiverPanel**. All these are created by enclosing attributes and methods that are used to represent input maps as different styles of graduated colours, unique colours, and so on. Secondly for map manipulation, there are two kinds of method which are constructed here – **get**() and **set**() or **update**() methods. A **get**() method is used to access a private attribute in the class. On the other hand, a **set**() method is implemented in order to change an attribute. As mentioned earlier, these two methods are called *accessor* and *mutator* methods respectively. Finally for the map construction, map layers need to be created based on combinations of other map layers. **Dasy** and **UDemand** classes are involved in this task and both will be explained in more detail later (see Block 4.5).

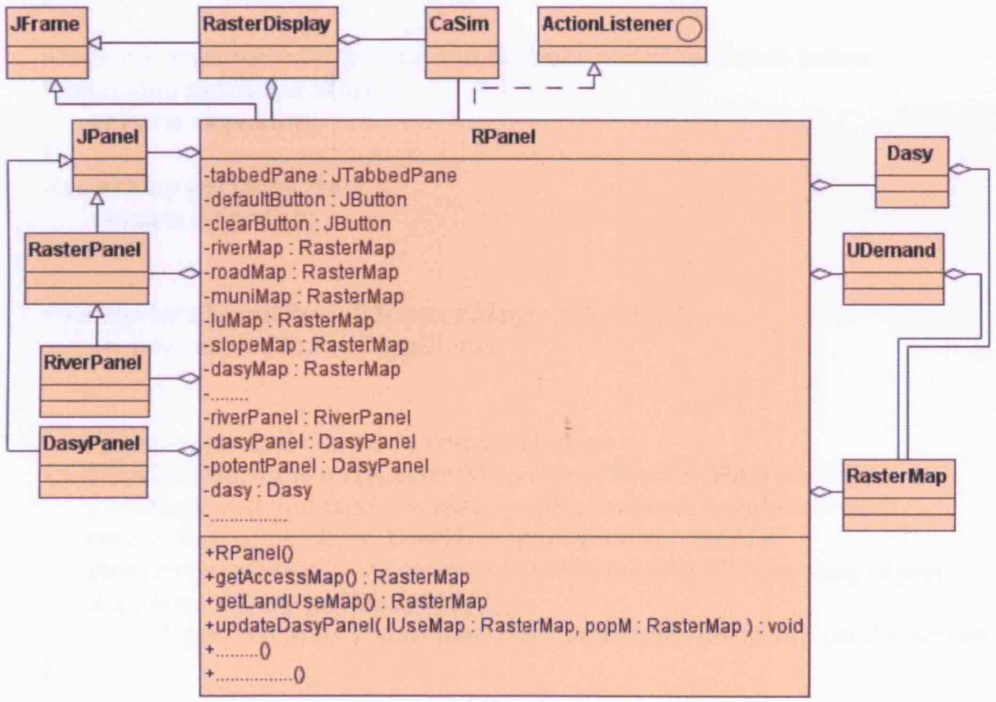


Figure 4.21 The RPanel Class Diagram

```
Class RPanel extends JPanel implements ActionListener
{
    Declare all necessary class variables from:
    RasterMaps, RiverPanel, Dasy, RasterPanels, DasyPanels,
    UDemand;

    Rpanel()
    {
        create riverMap object from RasterMap();
        create roadMap object from RasterMap();
        create muniMap object from RasterMap();
        create luMap object from RasterMap();
        create dasyMap object from RasterMap();
        create slopeMap object from RasterMap();
        create uDemandMap object from RasterMap();
        create fugalMap object from RasterMap();
        create petalMap object from RasterMap();
        create riverPanel object from RiverPanel();
        create roadPanel object from DasyPanel();
        create muniPanel object from RasterPanel();
        create luPanel object from RasterPanel();
        create dasyPanel object from DasyPanel();
        create slopePanel object from DasyPanel();
        create uDemandPanel object from DasyPanel();
        create fugalPanel object from DasyPanel();
        create petalPanel object from DasyPanel();
        create other GUI components: tabbedPane, buttons;
        add all panels and components to RPanel();
        display this window;
    }

    //--Some examples of Accessor and Mutator methods shown below
    RasterMap getSlopeMap(){
        return slopeMap;
    }
    RasterMap getDasyMap(){
        return dasyMap;
    }

    void updateFugalPanel(RasterMap rasterMap){
        set new rasterMap to fugalPanel;
    }

    //--Other concerned classes to construct maps
    void updateDasyPanel(RasterMap lMap, RasterMap popM){
        //--lMap: updated land use map, popM: updated population map
        create dasy object from Dasy(lMap, muniMap, popM);
        dasy.createDasy(); //--call a createDasy() method from dasy object
        dasyMap = dasy.getDasyMap();
        //--call getDasyMap() from dasy object redisplay dasyMap on the screen;
    }

    void updateUDemandPanel(){
        //--lMap: updated land use map, popM: updated population map
```

```
        create uDemand object from UDemand(luMap, muniMap, dasyMap,~
        popMap);
        uDemand.createUpdatedPop();
        uDemand.createUrbanDemand();
        uDemandMap = uDemand.getUrbanDemandMap();
        popMap = uDemand.getPopMap();
        redisplay dasyMap on the screen;
    }
}//--end of class
```

Block 4.5 Pseudo Codes for RPanel Class

The *rPanel* object in Block 4.5 concerns two other classes which were mentioned earlier: the **Dasy** and the **UDemand**. Both are created when the *rasterDisplay* object calls **updateDasyPanel**() and **updateUDemandPanel**() respectively in each time state. In the case of *dasy* object, the *rPanel* object passes three raster maps – land use (*lMap*), administrative boundary (*muniMap*) and population (*popM*) – to the **Dasy** class. Behind the method **createDasy**() called by the *rPanel* object, these maps are analysed by several methods which perform various GIS functions and routines which create a new map encapsulated in the **Dasy** class.

Consequently, *uDemand* object is created according to the following parameters: land use, administrative boundary, population and dasymetric population density. Moreover, *uDemand* has a set of methods (**createUpdatedPop**()) which calculate a new population map based on stochastic prediction. The new population map is returned to *rPanel* and then used in other classes (see the term *popMap* = *uDemand*.**getPopMap**()). We now describe more of the context for these two classes.

The dasymetric mapping technique has already been discussed in Chapter 3. The class implemented here composes the necessary attributes and functions for calculation. A vector object, *popVec*, is constructed from the **Vector** class and this is created in order to collect *population* objects. Much like a unique key field pointing to a database, *index*, is set as an attribute in a *population* object. Because the program requires population or other attributes from each territory, indexes are then defined to point to

or refer to each territory. As seen in the **Population** class in Figure 4.22, some attributes are fixed and some are calculated and redefined with respect to each object referred to by the indexes. Figure 4.23 clearly illustrates how the population objects can be created and associated with a vector object.

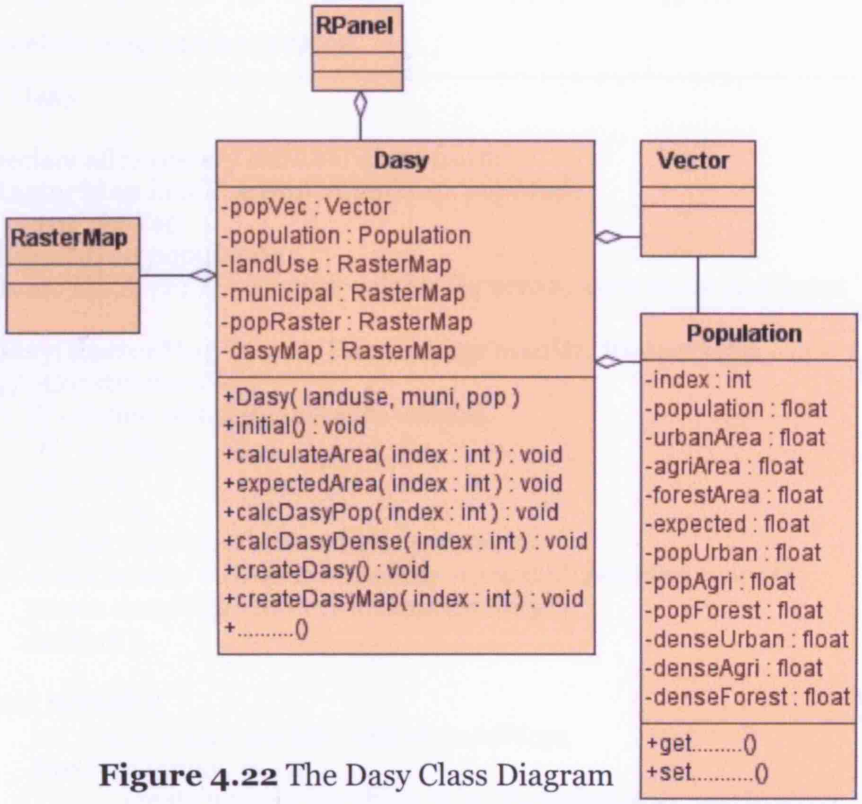


Figure 4.22 The Dasy Class Diagram

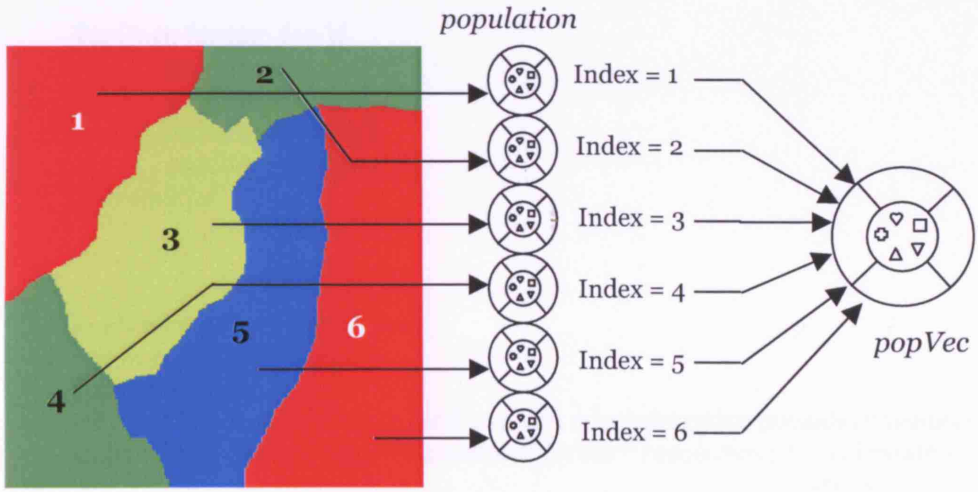


Figure 4.23 Reference Index for a Vector Object

So far, all the population, density and related parameters are separated and collected to a vector. Remaining tasks in the **Dasy** class involve computing population density in dasymetric fashion, arranging all the parameters collected from computing the vector and creating a map, and then passing this back to the *rPanel* object. Conceptually, the codes in Block 4.6 represent how all these attributes are passed and how the dasymetric map can be created.

```
Class Dasy
{
    Declare all necessary class variables from:
    RasterMap landUseMap, muniMap, popMap;
    Vector popVec;
    Population population;
    float rU, rA, rF; //--relative weights for urban, agriculture and forest

    Dasy(RasterMap lUseM, RasterMap muniM, RasterMap popM )
    {//--Constructor-----
        //--define adjustable relative weights
        rU = 0.85;
        rA = 0.1;
        rF = 0.05;
        //--set class variables as input parameter
        landUseMap = lUseM; muniMap = muniM; popMap = popM;
        create dasyMap object from RasterMap();
        initial();
    }
    void initial(){
        int max = maximum attribute of muniMap;
        for(i=0; i<max; i++){
                create population object from Population(i, popMap(i));
                //--popMap(i) population attribute of area i
                add population(i) to popVec;
        }//--end for i

        for(j=0;j<max;j++){
                calculateArea(j);
                expectedArea(j);
                calcDasyPop(j);
                calcDasyDense(j);
        }//--end for j

    }

    void calculateArea(int index){
        int urban, agri, forest;
        int resolution;
        count all cells according to land use and administrative boundary (index);
        urban * resolution; agri * resolution; forest * resolution; //--calculate~
                                                                    areas
        set all areas to popVec[index];
    }
```

```
    void expectedArea(int index){
        float e;
        e = rU * popVec[index].getUrbanArea() + rA *~
        popVec[index].getAgriArea()  + rF * popVec[index].getForestArea();
        set e to as expected in popVec[index];
    }

    void calcDesyPop(int index){
        float pU, pA, pF;
        pU = rU * popVec[index].getUrbanArea() * ~
        popVec[index].getPopulation() / popVec[index].getExpected();
        pA = rA * popVec[index].getAgriArea() * ~
        popVec[index].getPopulation() / popVec[index].getExpected();
        pF = rF * popVec[index].getForestArea() * ~
        popVec[index].getPopulation() / popVec[index].getExpected();
        set pU, pA and pF to as popUrban, popAgri and popForest  respectively~
        in popVec[index];
    }

    void calcDasyDense(int index){
        float dU, dA, dF;
        dU = popVec[index].getUrbanPop() / popVec[index].getUrbanArea();
        dA = popVec[index].getAgriPop() / popVec[index].getAgriArea();
        dF = popVec[index].getForestPop() / popVec[index].getForestArea();
        set dU, dA and dF to as denseUrban, denseAgri and denseForest~
        respectively in popVec[index];
    }

    void createDasyMap(int index){
        define denseUrban, denseAgri and denseForest in popVec[index] to ~
        dasyMap based on [index] of administrative area and land use: urban, ~
        agriculture and forest ;
    }

    void createDasy(){
        int max = maximum attribute of muniMap;
        for(j=0; j<max; j++){
                createDasyMap(j);
        }//--end for j
    }
}//--end of class
```

Block 4.6 Pseudo Codes for the Dasy Class

UDemand is a class implemented extensively in the program so that two more tasks can be achieved within the simulation model. These are: 1) population prediction and 2) urban land demand calculation. Firstly in general, there are several methods that can be used to project the population, from the most simple straightline forecast to much more complicated nonlinear methods. In this version of the model, the

population component is however implemented with a simple predictive method but in a stochastic manner. Random population growth rates for each territory are generated dynamically over a time state and used to predict new population numbers for the next time state. Secondly, the general idea of calculating demand for urban activities is to define a variable threshold of suitable population levels for each area and to compare these with current population densities. The urban demand map created in this fashion would thus represent levels of urban land requirement based on the amount of population required and the amount of land occupied.

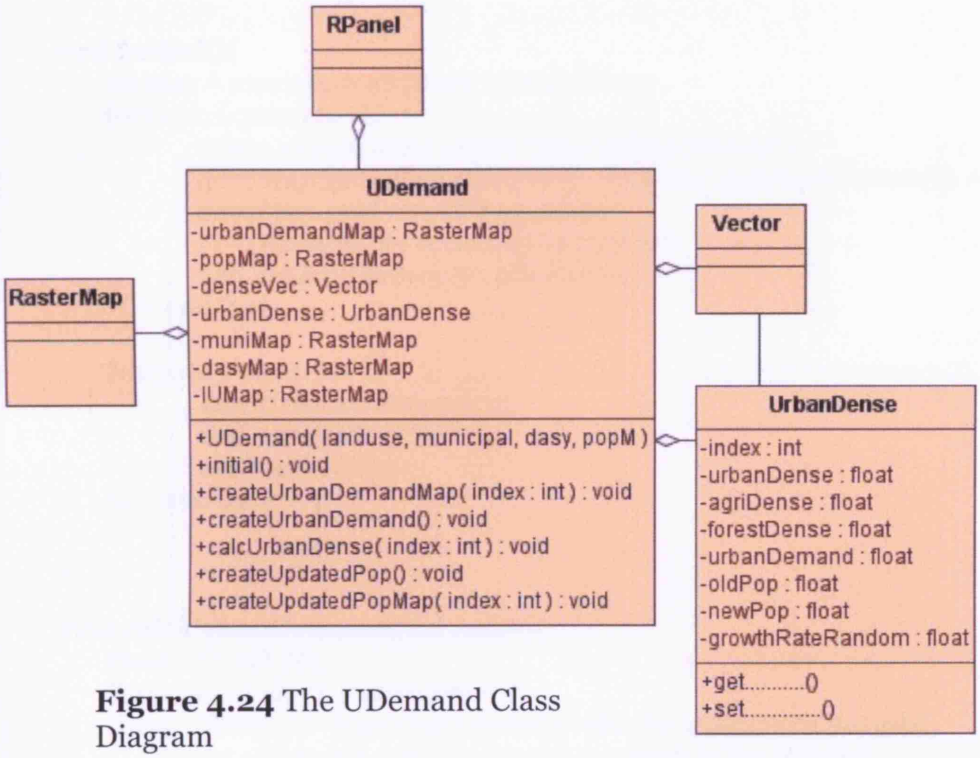


Figure 4.24 The UDemand Class Diagram

Figure 4.24 illustrates this **UDemand** class with its attributes and methods. Much like the **Dasy** class, a vector object, *denseVec*, is constructed to collect *urbandDense* objects that contain all essential attributes. Attributes in those objects would be retained as parameters which enable the program to process other subsequent operations in the *uDemand* object (see Block 4.7).

```
Class UDemand
{
    Declare all necessary class variables from:
    RasterMap landUseMap, muniMap, popMap, dasyMap, uDemandMap;
    Vector uDenseVec;
    UrbanDense urbanDense;


    UDemand(RasterMap lUseM, RasterMap muniM, RasterMap dasyM~
    , RasterMap popM )
    {//--Constructor-----
        //--set class variables as input parameter
        landUseMap = lUseM; muniMap = muniM;
        dasyMap = dasyM; popMap = popM;
        create uDemandMap object from RasterMap();
        initial();
    }


    void initial(){
        int max = maximum attribute of muniMap;
        for(i=0; i<max; i++){
                create urbanDense object from UrbanDense(i, ~
                dasyMap.getUrbanDense(i), dasyMap.getAgriDense(i), ~
                dasyMap.getForestDense(i));
                //--Densities of i classified by land use
                add urbandDense(i) to uDenseVec;
        }//--end for i

        for(j=0; j<max; j++){
                calcUrbanDemand(j);
                defineOldPop(j);
                defineRate(j);
        }//--end for j

    }

    void calcUrbanDemand(int index){
        float uD, aD, fD;
        float demand; //-- land demand
        float uTemp; //--temporary density value for computing demand
        //--define adjustable suitable density to calculate the demand
        float suitThreshold = 100;
        uD = uDenseVec[index].getUrbanDense();
        aD = uDenseVec[index].getAgriDense();
        fD = uDenseVec[index].getForestDense();
        //--all weights following are changeably
        wU = 0.8;
        wA1= 0.2; wA2 = 0.6;
        wF = 0.1;

        //--weight only on urban and agriculture if there is an urban cell in a~
        territory
        if(uD != 0) then uTemp = (wU * uD)+(wA1 * aD);
        //--if no more urban cell in a territory, weight for agriculture and forest
```

```
        else uTemp = (wA2 * aD)+(wF * fD);

        demand = (uTemp - suitThreshold) / suitThreshold;
        set demand as urbandDemand in uDenseVec[index];

    }

    void defineOldPop(int index){
        get unique attributes from popMap as variable oP;
        set oP as oldPop in uDenseVec[index];
    }

    void defineRate(int index){
        //--defining lower and upper bounds of random number of growth rates
        //--Both are amendable
        float lower = 0.01;
        float upper = 0.3;
        generate a random number, random, range from lower to upper;
        set random as growth Rate in uDenseVec[index];
    }

    void calcNewPop(int index){
        float nP;
        nP = (1+uDenseVec[index].getRate()) *uDenseVec[index].getOldPop();
        set nP as newPop in uDenseVec[index];
    }

    void createUpdatedPopMap(int index){
        define newPop in uDenseVec[index] to popMap based on [index] of ~
        administrative boundary;
    }

    void createUpdatedPop(){
        int max = maximum attribute of muniMap;
        for(j=0;j<max;j++){
                calcNewPop(j);
                createUpdatePopMap(j);
        }//--end for j
    }

    void createUrbanDemandMap(int index){
        define urbanDemand in uDenseVec[index] to uDemandMap based on~
        [index] of administrative;
    }

    void createUrbanDemand(){
        int max = maximum attribute of muniMap;
        for(j=0;j<max;j++){
                createUrbanDemandMap(j);
        }//--end for j
    }
}//--end of class
```

Block 4.7 Pseudo Codes for the UDemand Class

Classes for Node, Hierarchy and Spatial Interaction

An important part of the model is to simulate spatial interaction within each urban area and its hinterland. As defined in the last chapter, nodes are assumed to be centres around which urban areas develop and are thus measured by the distribution of urban cells in the cellular space. Each node can be classified by various types of data such as population and economic attributes, and as such, are recreated as nodes with specific positions or order within their hierarchies. Finally, the distribution of all the nodes with their identified values can be used to calculate the spatial interaction in potential terms and thus create the surface map.

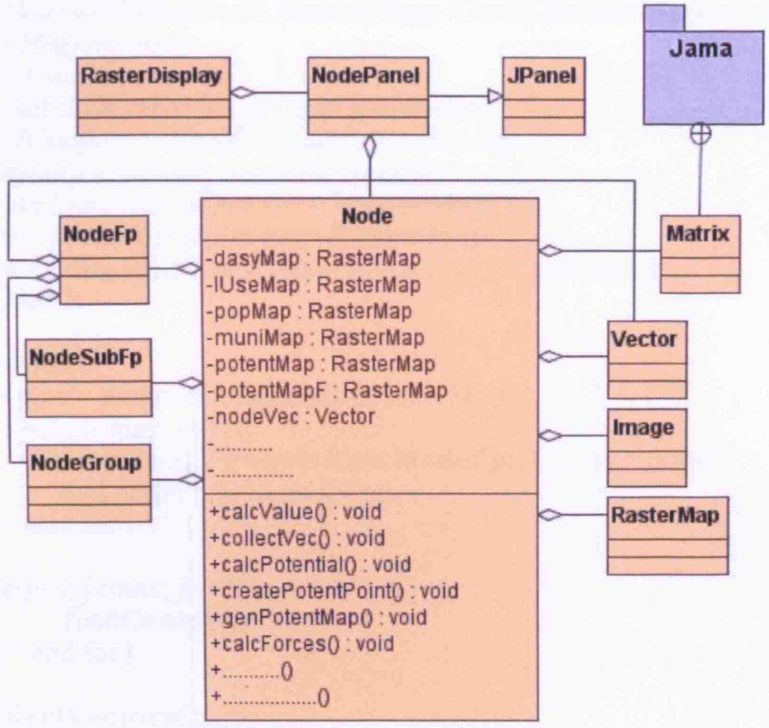


Figure 4.25 The Node Class Diagram

The class diagram in Figure 4.25 outlines the **Node** class and its associations. The **Node** class basically belongs to the **NodePanel** which is inherited from the **JPanel**. The **NodePanel** is then implemented as an interaction with the **RasterDisplay** and the **Node** class. It contains compulsory methods used for drawing and redrawing graphics on the panel with respect to the accessor and mutator methods.

Three sets of functions are developed in this class for different purposes: 1) for finding centres and generating nodes, 2) for computing hierarchies, and 3) for mapping surface maps. The **Node** class enables a number of *rasterMap* objects to be created, some of which are input from the **NodePanel** and the **RasterDisplay**; all are handled by functions inside the class (see Block 4.8).

```
Class Node
{
    Declare all necessary class variables from:
    RasterMap dasyMap, lUseMap, popMap, muniMap;
    RasterMap fugalMap, petalMap;
    Vectors nodeVec, gVect, pVect; //--other temporary vectors are declared~
                                        here

    Node(RasterMap dasyM, RasterMap lUseM, RasterMap popM , ~
    RasterMap muniM)
        {//--Constructor-----
        //--set class variables as input parameter
        landUseMap = lUseM; muniMap = muniM;
        dasyMap = dasyM; popMap = popM;
        create fugalMap object from RasterMap();
        create petalMap object from RasterMap();
        create other temporary rasterMaps used for calculation;
        initial();
    }
    void initial(){
        int max = maximum attribute of muniMap;
        for(i=0; i<max; i++){
                create nodeFp object from NodeFp(i); //-- i : index
                add nodeFp(i) to nodeVec;
        }//--end for I

        for(j=0;j<max;j++){
                findCentre(j);
        }//--end for j

        collectVectors();
        calculateSize();
        calcPetalP();
        createPetalMap();
        calcFugalP();
        createFugalMap();
    }
    .....
    .....
```

Block 4.8 Class and Constructor for Node

Firstly, to find centres and generate nodes in the simulation space, some other classes such as **NodeFp**, **NodeSubFp** and **NodeGroup** are required. We assume that at least one node created from a set of urban cells would be placed in each administrative area. However, some territories that have large amounts of urban activity are likely to have more than one node associated with their urban area.

Two classes, the **NodeFp** and the **NodeSubFp**, are constructed in order to manage the attributes of the administrative boundaries and the attributes for node generation within each area. The **Node** class holds a vector object that collects *nodeFp* objects which attributes reference numbers and population densities to each administrative territory. Once the vector object is created and all attributes are collected, the vector is now ready to store several nodes generated by the program. At some point in the simulation time, the node objects create a number of objects from the **NodeSubFp** that attribute key parameters such as x and y coordinates, population densities, and size of nodes, and then all the *nodeSubFp* objects are stored in the vector object which is referred to by its unique administrative boundary number (see Figure 4.26).

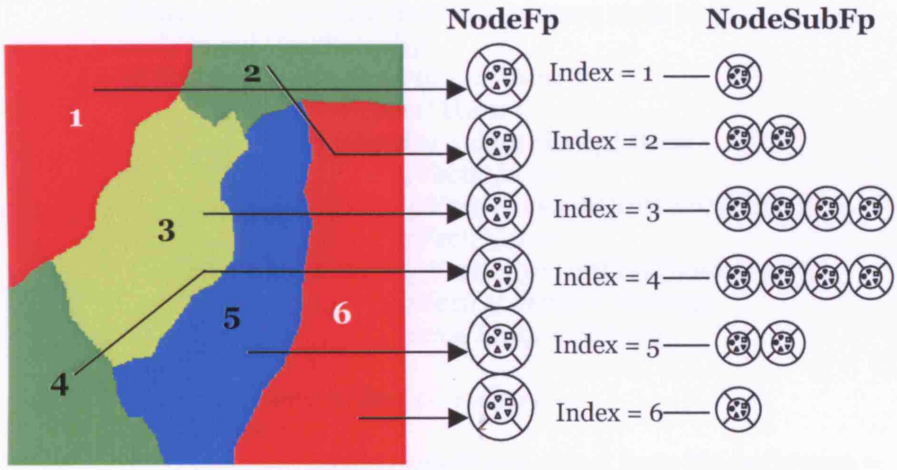


Figure 4.26 Double Vector Collections

The **Nodegroup** is constructed to create temporary objects during the simulation so as to find the centres within the urban areas. The way we find centres and generate nodes has already been explained in Chapter 3 but more details are shown in the codes in Block 4.9 below.

```
Class Node
{
    ..............//--constructor
    ..............//--other methods


    void findCentre(int index, int searchRadius, int resolution){
        int number = 1;
        float dense;
        create subVectTemp from Vector();
        create gVect from Vector();
        //--sum up urban cells within search radius for each territory
        for(i=0; i<nrows; i++){
                for(j=0;j<ncols;j++){
                        dense = sum up all urban density cells of dasyMap(i,j)~
                        within searchRadius in each administrative boundary ~
                        (index);
                        create nodeSubFp object from NodeSubFp(index,~
                        number, i, j);
                        add nodeSubFp to subVectTemp;
                        number++;
                }
        }
        //--add first nodeSubFp to gVect--
        create nodeGroup object from NodeGroup(subVectTemp[1]);
        add nodeGroup to gVect;
        denseTemp = subVectTemp[1].getDense();


        for(i=0; i<subVectemp.size(); i++){
                get density from subVectTemp[i];
                float dist = calculate distance between node in gVect[1] and ~
                node in subVectTemp[i];
                if(first time node generation) then {
                        if(dist <= radius) then {
                                if (density > denseTemp) then {
                                        gVect[1].x =~
                                        (gVect[1].x+subVectTemp[i].x)/2;
                                        gVect[1].y =~
                                        (gVect[1].y+subVectTemp[i].y)/2;
                                        gVect[1].dense = density;
                                        denseTemp = density;
                                }
                        }//--end if(dist <= radius)----
                        else {
                                create nodeGroup object from NodeGroup ~
                                (subVectTemp[i]);
                                add nodeGroup to gVect;
                        }//--end else---
                }//---end if (first node generation)---
                else {
                        for(j=0;j<gVect.size();j++){
                        if(dist <= radius) then {
                                if (density > denseTemp) then {
                                        gVect[j].x =~
```

```
                                    (gVect[j].x+subVectTemp[i].x)/2;
                                    gVect[j].y =~
                                    (gVect[i].y+subVectTemp[i].y)/2;
                                    gVect[j].dense = density;
                                    denseTemp = density;
                            }
                        }//--end if(dist <= radius)----
                        else {
                                create nodeGroup object from NodeGroup ~
                                (subVectTemp[i]);
                                add nodeGroup to gVect;
                        }//--end else---
                        }//--end for
                }//--end else----
        }//--end for I-----

        //--compare all nodes collected in gVect;
        gVectTemp = gVect;
        for(i=0; i<gVect.size(); i++){
                for(i=0; i<gVect.size(); i++){
                        calculate distance between gVect[i].node and
                        gVectTemp[i].node;
                        if (distance < radius) then remove that
                        gVectTemp[i].node;
                }—end for j
        }//--end for I
    }
    gVect = gVectTemp;
    collect all remained nodes in gVect to nodeVec[index] in form of attributes of
    nodeFp
}
```

Block 4.9 Finding Centres and Generating Nodes

Secondly, nodes only represent the places where an urban area identifies its centres and they do not illustrate interrelationships between the urban areas and its centre or between centres. Hierarchical nodes, however, are much more focussed on representing distribution and interaction in the spatial sense. The **Node** class implements all the key methods for all nodes and ranks them according to their attributes of population density. The rank order values of all nodes in *nodeVec* collected from the **findCentre**() in Block 4.8 are employed to visualise nodes in different sizes of symbol. A method, **calculateSize**(), is then simply executed over all nodes in *nodeVec*, and this uses their own attributes to compute sizes which are finally associated with the attribute *size*. The method, **calculateSize**(), is therefore extended to the Node class shown in Block 4.10 below.

```
Class Node
{
    ..............//--constructor
    ..............//--other methods

    void calculateSize(){
        float size;
        float min = nodeVec.getMinimum();
        float max = nodeVec.getMaximum();
        int lowerSize = 5; //--smallest symbol for visualisation
        int upperSize = 25; //--largest symbol for visualisation
        for(i=0; i<nodeVec.size(); i++){
                for(j=0;j<nodeVec[i].getRecord();j++){
                        float value = nodeVec[i].getDense(j);
                        size = (upperSize * ((value-min)/(max-min)))+lowerSize;
                        set size to nodeFp[j] within nodeVec[i];
                }//--end for j
        }//--end for i

    //--This method will be applied in NodePanel class
    void paint(Graphics g, int width, int height){
        some commands to draw graphics g within width and height dimensions;
    }

    }//--end calculateSize()
} //--end of class
```

Block 4.10 Calculating Sizes of Nodes and Their Display

Lastly, the class also prepares an array of methods associated with the spatial interaction model and its surface interpolation. In addition to ranking all nodes to generate the hierarchical node map, the population density parameter also plays a role as 'mass' in the spatial interaction functions. A routine distance algorithm is developed to access all the attributes of nodes and to measure distances between them. Once all the required parameters are collected, another module calculates the potential values which are returned to the nodes (see the discussion of spatial interaction in Chapter 3). Thus far, nodes contain some basic attributes created by the constructor of the class as well as now containing key parameters applied in the construction of the potential surface map. The inverse distance weight interpolation technique is consequently applied to the model based on this technique, and implemented in the set of algorithms described below.

These methods are then extended in the **Node** class to calculate the interaction between all nodes in the space. Referring again to the concepts of force described in Chapter 3 – the centrifugal and the centripetal forces, these two forces are implemented using the modified spatial interaction model. Block 4.11 which follows explains how the attributes of all nodes in the *nodeVec* created from algorithms in the previous blocks, can be manipulated to create the surface maps representing both forces.

```
import Jama.*; //--used for calculate polynomial equation

Class Node
{
    Vector pVect, pVectTemp;
    .....//--constructor
    .....//--other methods

    void collectVectors(){
        add all nodeSubFp objects from nodeVec to pVect;
    }//--end collectVector---

    //--Petal force applies original distance exponents equal to 2
    void calcPetalP(){
        pVectTemp = pVect;
        float limit = 50; // define limitation for distance to calculate potentials
        float res = 100; //resolution of raster map
        float potential = 0;
        for (j=0;j<pVect.size();j++){
                for (k=0;k<pVect.size();k++){
                        if(pVect[j] == pVect[k]) then continue;
                        else{
                                float d = calculate distance between pVect[j] and~
                                pVect[k];
                                if(d <= limit) then {
                                        p = pVect[k] / (d*res)*(d*res);
                                        potential = potential + p;
                                        minDist = find minimum distance to~
                                        minDist;
                                }//--end if--
                        }//--end else--
                }//--end for k
                //--calculate potent of node itself
                p = pVect[j] / ((minDist*res/2) * (minDist*res/2));
                potential = potential + p;
        }//--end for j----
        collect all potential from nodeVec;
    }//--end calcPetalP

    //--Apply Inverse Distance Weight (IDW) interpolation
    void createPetalMap(){
        nrows =number of row of petalMap;
        ncols =number of column of petalMap;
```

```
        for (i=0; i<nrows; i++){
                for (j=0; j<ncols; j++){
                        for (k=0;k<nodeVec.size();k++){
                                for (l=0;l<nodeVec[k].getRecord();l++){
                                        dist = calc distance between pixel[i][j]~
                                        and nodeVec[k].node(l);
                                        bT = 1 / (dist*res)*(dist*res);
                                        aT = nodeVec[k].getPotent(j)/ ~
                                        (dist*res)*(dist*res);
                                        a = a + aT;
                                        b = b + bT;
                                }//--end for l
                        }//--end for k
                        p = a / b;
                        rescale p to pR ranged from 0 – 1;
                        set attributes pR to petalMap(i,j);
                }//--end for j
        }//--end for I
    }//--end createPetalMap---

    void calcFugalP(){
        in the same fashion of calcPetalP();
    }

    void createFugalMap(){
        in the same fashion of createPetalMap();
        //--distance exponents defined by  polynomial equation ~
        //--the polynomial equation calculated using least square~
          method
        //--default equation ß = 2.3706+(-0.1798d)+(0.0041d²)
    }

} //--end of class
```

Block 4.11 Potential Surface Map Generator

5. Conclusions

Object-oriented programming represents a remarkably transparent approach to modelling systems which are composed of many objects, attributes and parameters, although it is essential to understand that the way of describing such programming is painstakingly detailed in comparison with previous generations of language. Indeed, it represents a way of thinking about the structures that merge every aspect of the way a program is to be developed using the same terminology. In agent-based modelling for instance, modellers can clearly take advantages from OOP for all agents created in such models are able to have independent

attributes with the same behaviours constructed from the same blueprint. Certainly, when the temporal dimension is added to the model, all agents can move or mutate autonomously. In our model here for example, as we associate maps with objects, all raster maps in the model are constructed using the same prototype and they are thus all independent of one another. As explained in this chapter, nodes in the model are yet another example which shows what the benefits of creating an object-based model can yield.

At this point, it is hopefully clear how we have implemented this simulation model using OOP. To summarise once again, several major classes that drive the model have been explained in detail, and these are: 1) the diffusion processes which randomly generate groups of seeds in the simulation space; 2) the cellular automata simulation which generates possibilities for transition of cells in terms of birth, development, survival, decline and death for all urban seeds in the space; 3) the dynamic map module which arranges and manages the input and output of map layers to all simulation processes; 4) the hierarchical node generator which gives the model an ability to find regional nodal centres, to rank them, and thus to create node maps; and 5) the spatial interaction module which derives some hidden node attributes with respect to their potentials and constructs potential surface maps which represent the two interactive forces. All are essential pieces in the composition of the model.

To investigate whether the model works consistently and whether or not all algorithms are usable, a trial version of the model will be constructed. In the next chapter, an experimental urban area equivalent to a small town is defined in hypothetical terms and then all the concepts described in Chapter 3 and this chapter are tested. This is a process not only of testing but also one of getting rid of various blunders in the programming syntax. As in all such modelling, the program that has been described in this chapter evolved over many months and thus what we have just described is a 'finished product'. When we test the model in the next chapter, the results of running a hypothetical model will then be used to adjust the model in various ways so that we are then able to apply it to the simulation of real urban phenomena at the next stage.

CHAPTER FIVE
Experiments with the Theoretical Model

1. Introduction

To understand how any simulation model works, it is necessary to perform experiments on the model using hypothetical data which is under the control of the model builder. Although the last two chapters have explained the concepts and procedures behind the model, we will test these concepts in this chapter by designing experiments which will be implemented on an artificial space populated with hypothetical data. The first part of this chapter discusses how the area was designed and how all the required data layers were prepared. Some characteristics of the experiments using this space will be explained which are consistent with the various concepts behind the model. At this stage, two levels of spatial resolution based on an area of 50x50 pixels and one of 250x250 pixels will be defined for different purposes. The smaller resolution space, 50x50 pixels, is advantageous for studying various flows within the model but it is a far cry from an application that is close to reality in terms of its visualisation. The 250x250 resolution space, on the other hand, is much closer to a reality but it is more difficult in this case to monitor all the procedures associated with the model's algorithms. This example associates the model with data which is much closer to the real world phenomena, and thus takes the analysis to the point where the model can be developed for actual applications.

A later part of this chapter describes the experimental context. In terms of the data preparation, two different sets of data were used to generate results from the model. Three issues will then be discussed in the rest of the chapter: 1) the parameters used in the model, 2) experimental design and then the actual experiments with the model, and lastly 3) visualisation of the results from the simulations.

2. The Test Area and Its Data

This section introduces the test area and how we define it. The first part elucidates some specific characteristics of the area that are suitable for applications of the model and the next part will illustrate the two sets of data which have been prepared at different scales.

Defining the Test Area

Because of the intricate dependencies within the data and the way this data are used in the model, unlike other simulation models that do not need predefined data sets, the DSSM requires some sort of specific initial data which represents the default state of simulation – in short, the model's initial conditions. From the data requirements posed in the last two chapters, the model requires a spatial area definition that meets the following conditions:

1) There is at least one big city in a part of the space and there are some smaller towns in the surrounding area. As the model focuses on the spatial interaction between cities and towns, to test the model, the area should include a large city with a much larger hinterland than the other towns in the same space. Moreover, these spatial definitions should accord to realistic numbers of population which are usually associated with the sizes of these different urban areas. In reality, most city systems in Thailand or indeed many cities in other parts of the world have similar hierarchical definitions in terms of the way they are spatially distributed and the way the numbers of their populations are associated with their position in their hierarchy. Thus in these experiments, we will seed the cellular space with urban areas arranged in this fashion.
2) Our hypothetical area requires administrative boundaries for it is difficult to collect and retrieve data for this model without such spatial referents. Obviously, the socio-economic and geodemographic data are based on political spatial units defined by the boundaries associated with the country, province, district or sub-district etc, while

some data is based on physical spatial reference areas such as watersheds, land cover types and so on. We therefore need to define layers in this manner to include in the model.

3) There are also quantitative indicators for each boundary area that are associated with simulating the growth of an urban area such as employment, income, and population etc. In further development of this model which will be part of future research directions, these data might be statistically combined to yield more effective indicators, but for the moment, the model has been developed to support only a single indicator which is currently population. This is a good option at this stage for population is always positively correlated with the process of urban development.
4) There are various data that are required which relate to constraints posed by rivers and/or water bodies in the landscape. To run our hypothetical simulation, we will mask certain areas as water bodies for all cities for most small towns in Thailand have some parts of their area covered by water. In fact, such constraints are a more general feature of all such areas which need to be set for simulation models.
5) The area must have existing transportation networks for these are essential to evolution in most modern cities. Transportation influences the shape of cities and in this model, it is instrumental in the way the model generates urbanisation in the wider landscape
6) The topography of the area should also be varied. Although many urban areas are located on flat lands, topography in the form of slope does influence urban development, and to show how this occurs, we will define a variable topography with different slope levels.
7) We will classify the area into three classes of land use/cover which are based on urban, agriculture, and forest areas. The model developed thus far performs mutations on just these three types of land. This is well suited to the urban-regional scale, for greater detail in the form of more land use classes would complicate the running of the model and would not be consistent with the level of

spatial resolution. This of course is somewhat independent of the actual generic structure of the model that could be adapted quite easily to finer scales of working.

All seven characteristics defined above feature in the spatial structure of most cities across the world, and thus we have some confidence that the hypothetical area we have designed provides a good example for scrutinising the various processes in the model that will be applied to real world phenomena at the next stage.

The Scales of the Trial Area

Two scales of an arbitrary city have been set to test the model in two different ways. Figure 5.1 illustrates these two scales simply in terms of the imposed river layer in the test area. It is fairly clear from Figure 5.1a that implementing the model with these rather rough-scale datasets takes us pretty far from the reality. However, this is ideal for examining how cells in the space make transitions from one state to another according to the algorithms implemented in the model. Thus all these processes can be monitored when changes are made to various thresholds and the mutation of cells shows up much more clearly than with the sorts of fine-scale data that a real application would require.

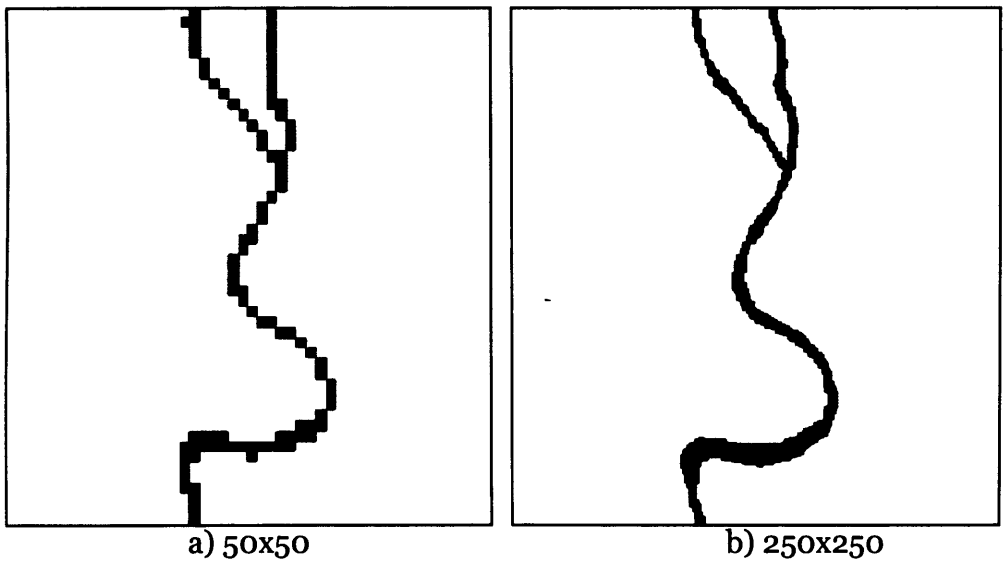


Figure 5.1 The Two Hypothetical Regions: Comparison of the Two Scales

On the other hand, to cope with limitations posed by modelling this rough-scale data, fine-scale datasets do have the advantage of connecting the model to real world phenomena as we show, but still in terms of a hypothetical data set, in Figure 5.1b. Rather than examining the cellular processes of the model, testing the model with finer-scale datasets helps us test the theories that lie behind the model involving urban-regional growth, spatial interaction, and so on.

The Trial Datasets

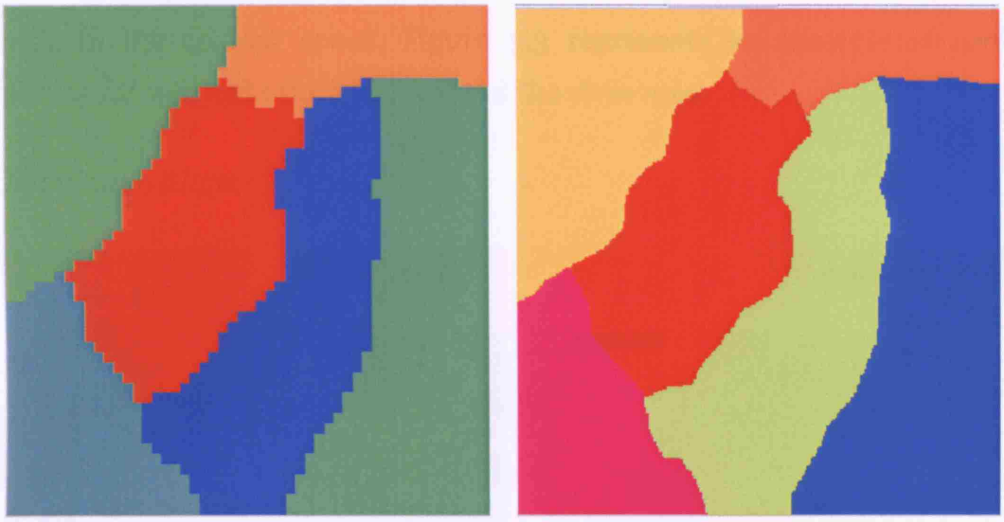
Based on the characteristics of the idealised case study area described in the last section, various trial data sets need to be prepared. There are six map layers to create which are the administrative boundary map, the river and water body map, the transportation, population, and land use/cover maps and the slope map. Five other map layers must be derived from these six basic maps and these are the hierarchical node map, population density dasymetric map, urban land demand map, and the centripetal and centrifugal surface maps. We have discussed these in some depth in the previous two chapters. and readers can familiarise themselves with their form by referring specifically to the discussion in Chapter 3.

The most preferable shape for the test area has been set as rectangular which enables us to tag each cell with appropriate values and to observe the cellular processes most clearly. All maps were prepared in ASCII format that is the easiest form in which to deal with input and output in the model. In the following figures, we visually represent all the map layers that have been reformatted to the model and displayed on the screen using the model GUI which we discussed in Chapter 4.

The Administrative Boundary Maps

The administrative boundary map represents the politically defined territories (or districts/zones) of the area. Based on the characteristics noted above, this map has been set as the base map for referencing all data and outputs used in the program. Moreover, this map has also been defined as a base for the thematic maps associated with the population and

urban land demand. In this case, there are six territories defined as shown at the two different scales in Figures 5.2a and 5.2b. Each territory is represented by a different colour which indicates its unique index. For real cases which we will deal with in later chapters, this administrative map would be divided instead by sub-district boundaries.



a) 50x50 b) 250x250

Figure 5.2 Administrative Boundary Maps

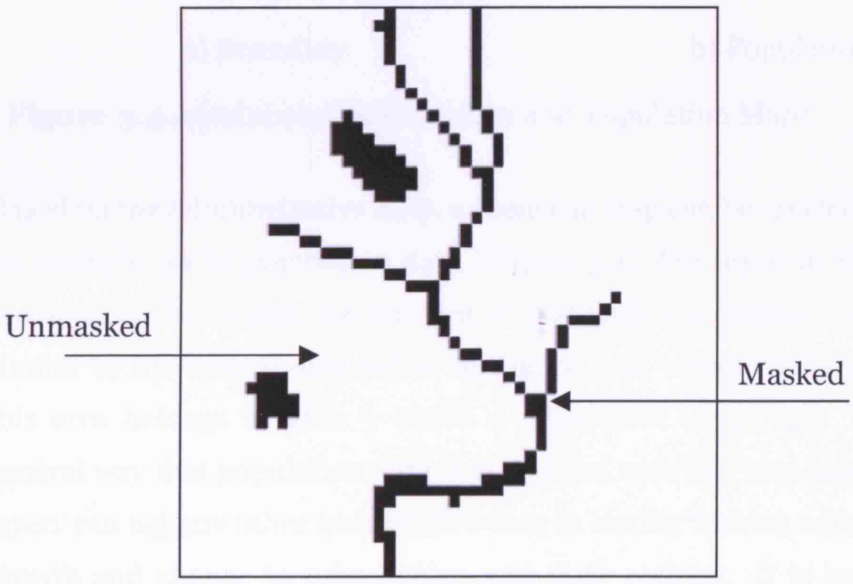
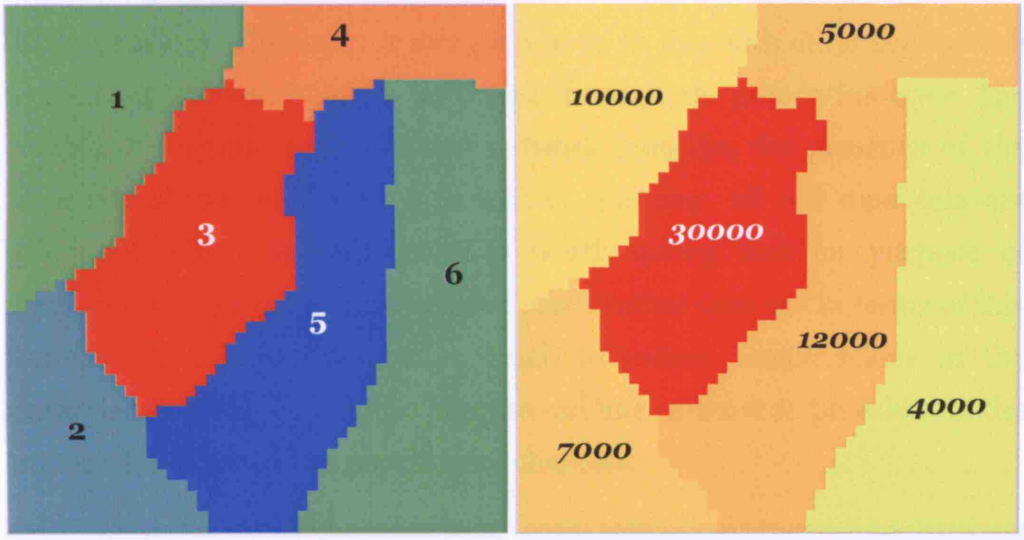


Figure 5.3 Additional Masked Cells on the River Map

River and Water Body Maps

The river map has already been shown in Figure 5.1 above. The values of the masked map are binary with the 'masked' and 'unmasked' cells in black and white respectively. In Figure 5.1, only one river and its branch have been included in the area. However, another module within the model has been developed so that the user can interactively input additional masked cells in the cellular space. Figure 5.3 represents an example of some additional masked cells which extend the river map.

Population Maps



a) Boundary b) Population

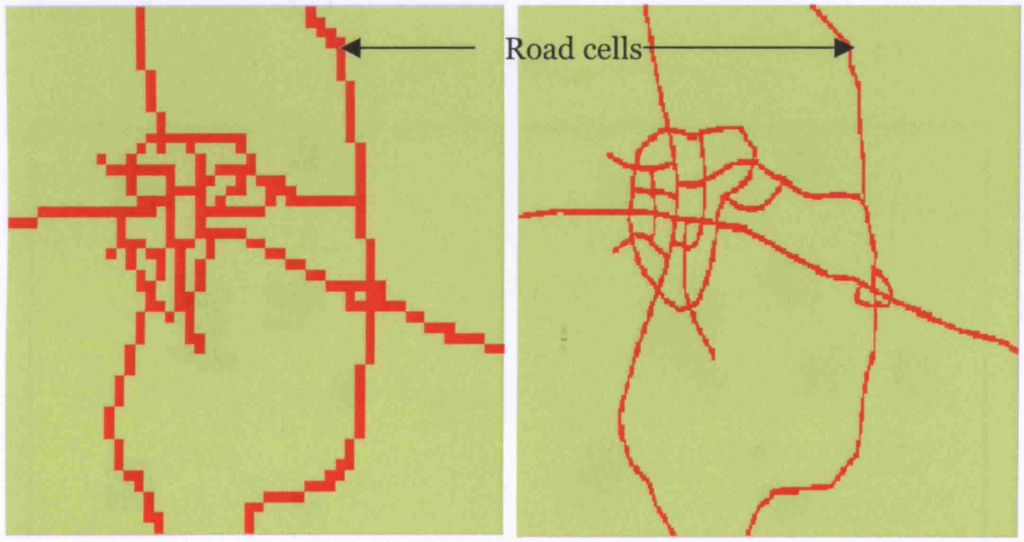
Figure 5.4 Administrative Boundary and Population Maps

Based on the administrative map, a thematic map can be created to classify an example set of population data (Figure 5.4). For input to the relevant modules in the model, the amount of population is defined in a fashion similar to the map shown below in Figure 5.4b where the largest city in this area belongs to area 3 which a population of 30,000. This is the general way that population data is associated with this map layer but such layers can tag any other quantitative data in similar fashion which involves growth and change in urban areas and their regions. It is important to note at this point that administrative boundary data is being associated

with the raster map and that later, the kinds of apportionments discussed in earlier chapters will be required when data comes to be associated with the various cells.

Transportation Maps

As we have argued and is well-known, transportation technology is one of the major reasons why urban space expands in most modern cities across the world. Most urban models developed so far build in such transportation features and this model is no exception for the transportation layer is considered to be one of the basic constraining factors. Figure 5.5 below demonstrates an example of the road map at two different scales. This layer is designed to be in line with other layers such as the river and water bodies layer and of course the population layer. The complexity of this transportation network indicates the presence of the large city shown in Figure 5.4b and in this way, all our data sets are consistent with one another. It is worth noting that for purpose of simplification, we have only included roads rather than rail in terms of this transportation infrastructure, as much to reduce complications in the experimental analysis but also because rail links are not so prevalent in the examples that we will engage in later chapters.



a) 50x50 b) 250x250

Figure 5.5 The Transportation Maps

Slope Maps

An arbitrary contour map which spatially conforms to other layers was drawn as an example DEM that is used to construct the slope map. Each cell in the slope map contains a value showing its percent change. Larger percent changes indicate greater slope with the implication that this has less potential for development. Figures 5.6 a and b demonstrate these slope maps at the two different scales.

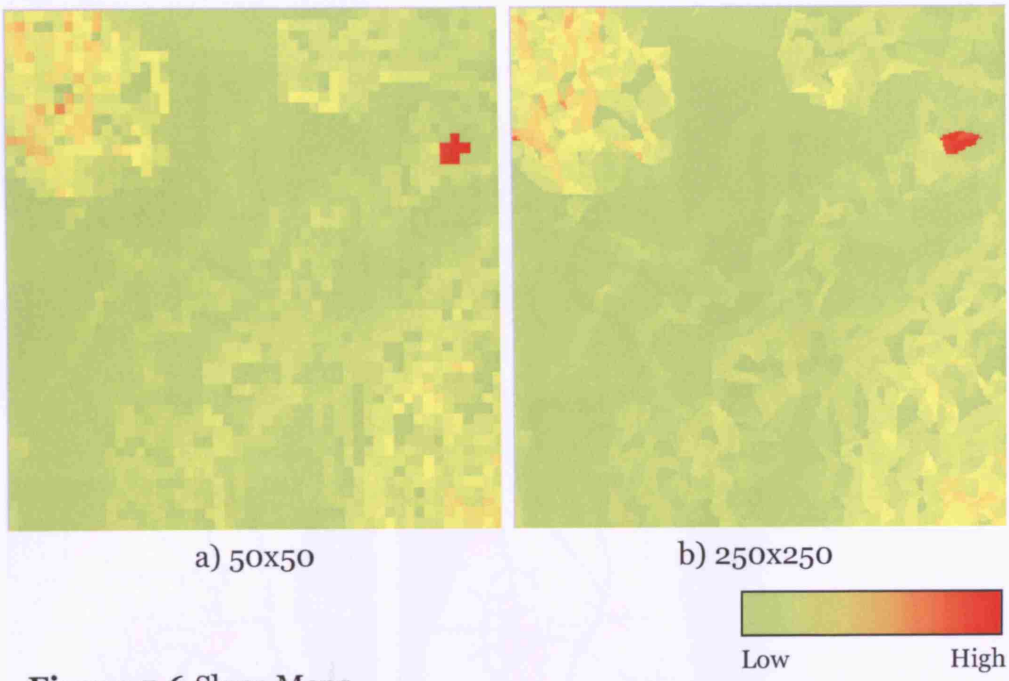


Figure 5.6 Slope Maps

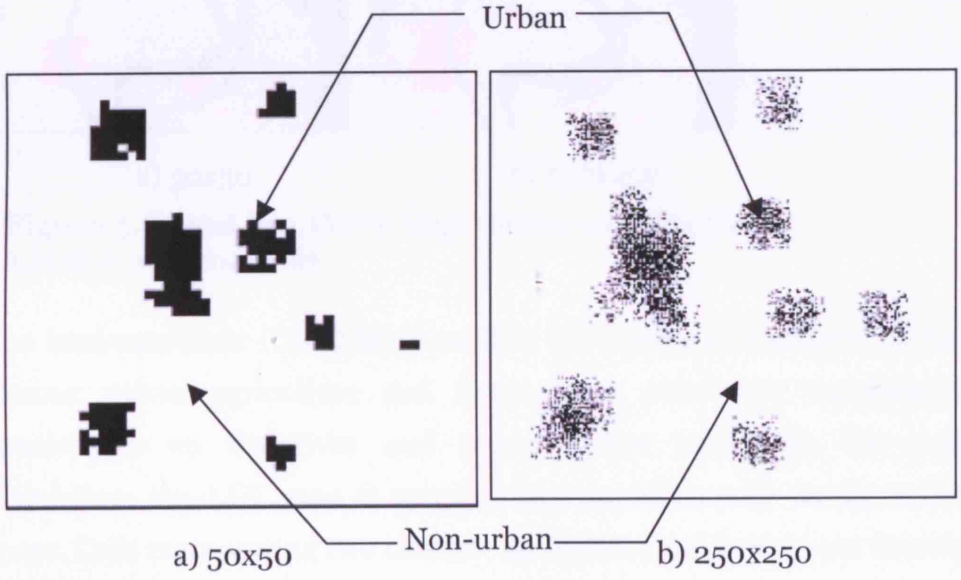
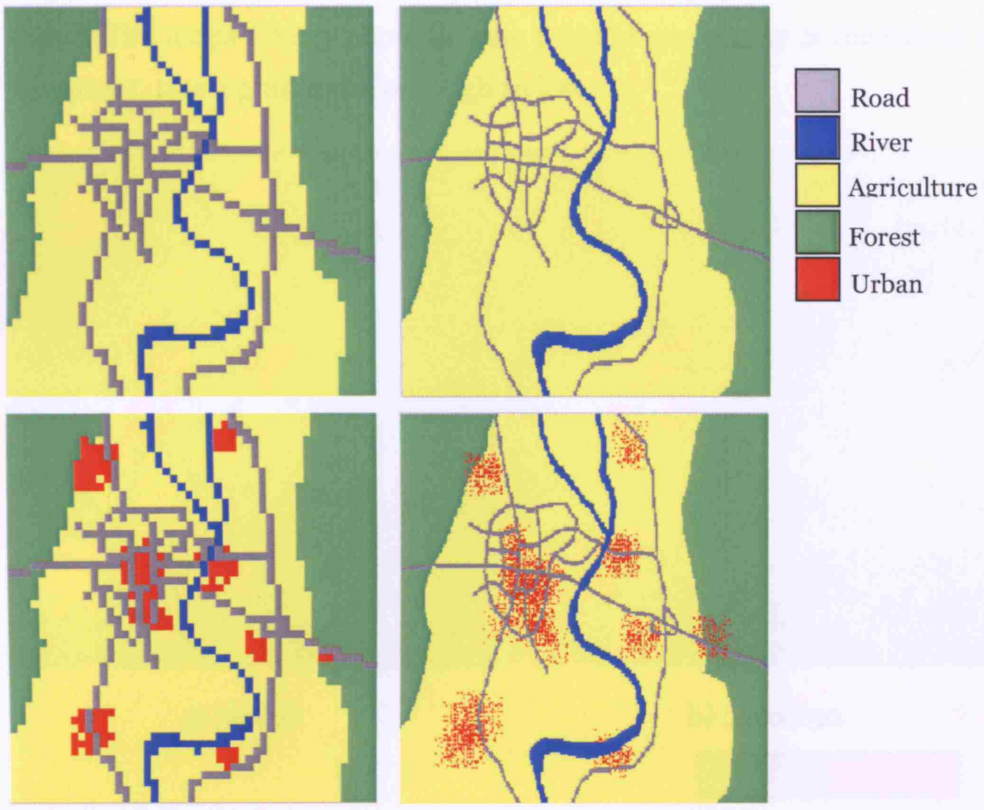


Figure 5.7 Urban Seed Maps

Urban Seed Maps

The GUI to the model allows the user to draw urban cells interactively in the graphic panel. Figure 5.7 represents examples at the two different scales showing urban cell maps used for the CA panels. This layer is mainly used in applying the CA and diffusion algorithms but this layer does coordinate synchronously with the land use/cover map as well.

The Land Use/Cover Maps



a) 50x50 b) 250x250

Figure 5.8 Land Use/Cover Maps Before and After the Addition of Urban Cells

The land use/cover (LUC) layer used in this model is organised as three classes: urban, agriculture and forest areas which are superimposed consistently on the river and transportation layers. In the model simulation, the LUC map is synchronised to urban cells in the cellular space. Cells representing two classes - agriculture and forest - are therefore defined prior to generating any other urban cells within the CA space. The

images shown in Figure 5.8 below represent these states before and after the urban areas are defined with respect to the urban map layers.

Population Density Maps

These maps represent a secondary process of map initialisation for they need to be constructed using the various dasymetric techniques detailed in Chapters 3 and 4. Using the apportionment techniques described there, the two maps are illustrated in Figure 5.9 and thus represent two scales of dasymetric map based on the spatial and non-spatial information given above. The maps clearly show in how population density is measured and visualised over a gradient from high to low.

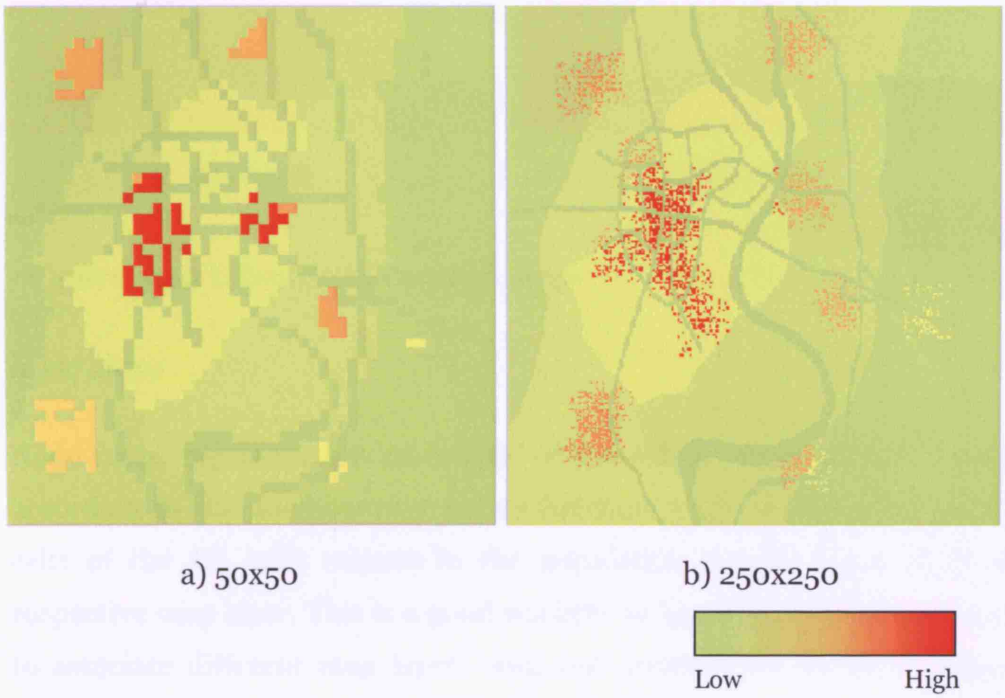


Figure 5.9 Population Density Dasymetric Maps

Urban Land Demand Maps

The urban land demand map layer is constructed around the population density map and administrative boundary map using a zonal statistic function. This thematic map represents which political territories demand more land for urbanisation, depending upon the density of population and land use/cover. The maps are created as shown in Figure 5.10 with the colours representing the gradation in levels of land demand. The dark

colour, in this case, belongs to the administrative unit 3 (see Figure 5.4 above) that was predefined as being the largest city in this hypothetical area.

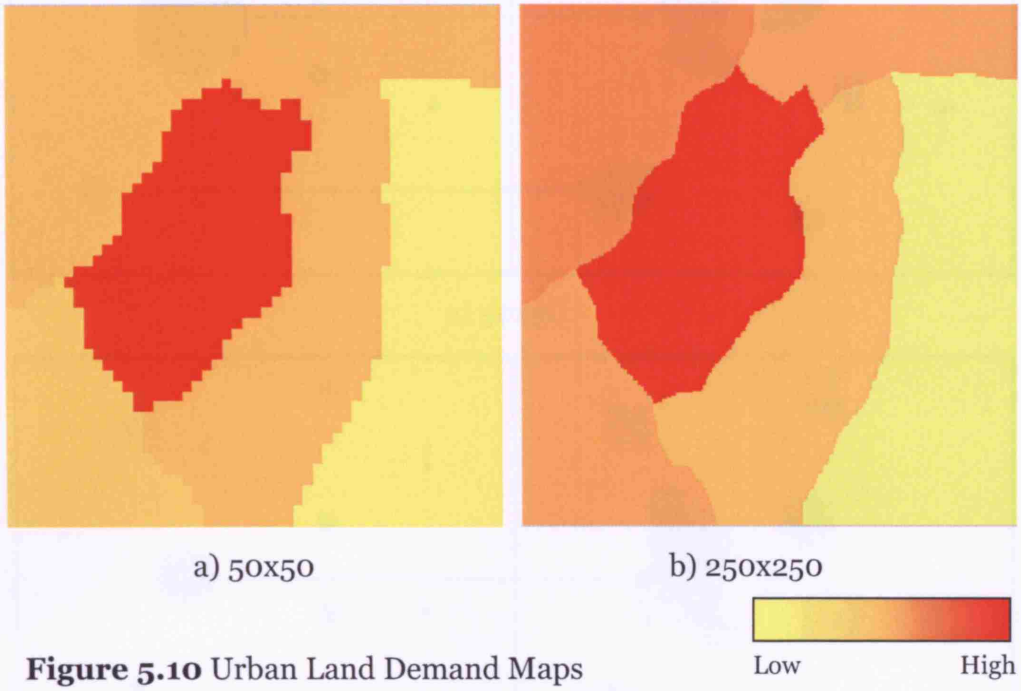
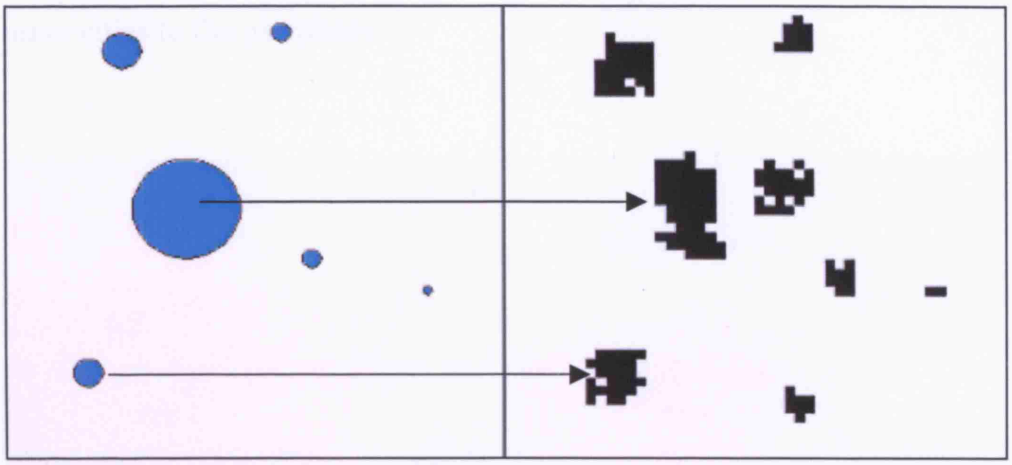


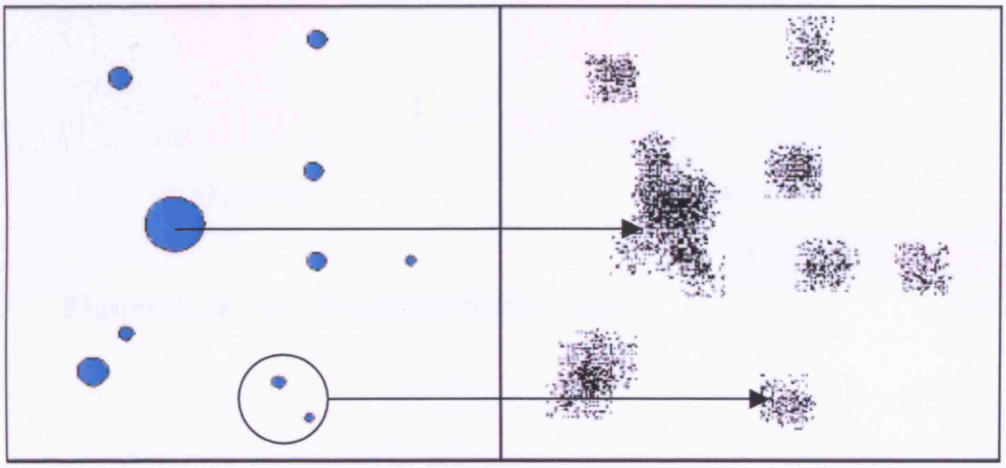
Figure 5.10 Urban Land Demand Maps

Node Maps

Node maps represent the 'centrality' of the urban areas and are created according to the neighbouring radius function which is computed on the cells of the CA with respect to the population density taken from its respective map layer. This is a good example of how the raster base is used to associate different map layers with one another. As shown in Figure 5.11, nodes are symbolised as circle shapes which imply different sizes that represent the hierarchy of the nodes defining the city system. Furthermore, the maps also demonstrate how the influences of distance and scale affect the number of nodes generated and their spatial distribution (consider the differences between the two maps in Figures 5.11a and 5.11b). It is clear that the increasingly fine scale provides a much more detailed hierarchical pattern to the system with new nodes being detected.



a) 50x50



b) 250x250

Figure 5.11 Node Maps and Urban Map Layers

The Centripetal Surface Map

The centripetal surface map is an abstract representation of the spatial surface of inward forces which are computed from the lower to higher potential nodes. The two maps in Figure 5.12 are obvious examples that demonstrate a large potential associated with the largest city surrounded by lower potential small towns. The maps indicate that vacant cells within the inner city and its encircled area have a very high probability of being transformed into an urban state. Conversely, any empty cells at remote distances from the towns and cities face difficulties in such a

transformation to urban due to the gradual decrease of potential from the inner cities to the periphery.

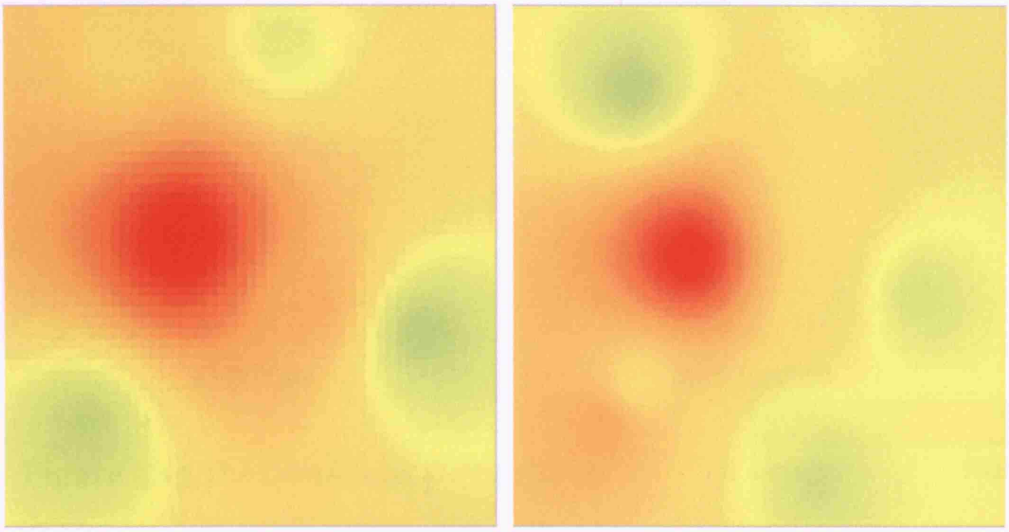


Figure 5.12 The Centripetal Surface Maps

The Centrifugal Surface Map

In similar fashion, the centrifugal surface map represents the outward forces and is somewhat opposite in its spatial pattern to the centripetal map. Push forces cause an urban area to decentralise; as seen in Figure 5.13 below compared to centripetal forces in Figure 5.12, all potential cells are reallocated and transformed outward from the centres to their hinterlands. However, cells do not spread all around the space but rather they distribute their potential to other nodes in the space, and new potential values are then recalculated. Both maps reveal that the potential intensities stretch to the most northern and southern parts of the area with the eastern part of the image still containing lower potential cells as displayed in the gradation from yellow to red in Figures 5.13a and 5.13b.

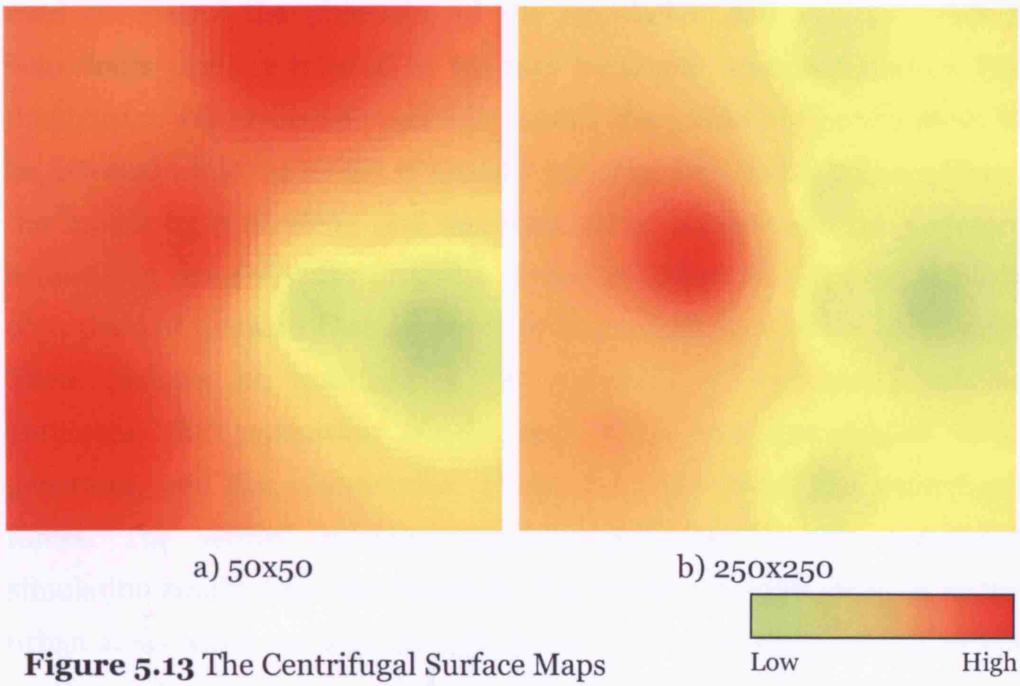


Figure 5.13 The Centrifugal Surface Maps

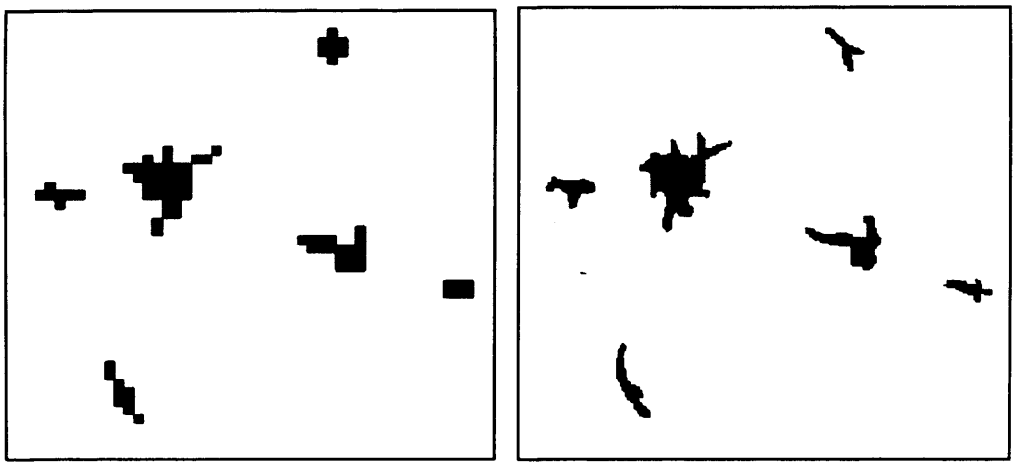
As visualised thus far, all these maps belong to the first time state of the simulation and represent the model's initial conditions. These maps will be used for the experimental testing in the next section from which various results will be extracted so that the concepts and programs in the last two chapters can be evaluated. Although all these maps have been fabricated, they do contain most of characteristics that are close to real world city systems and therefore these tests should enable the model builder to judge the adequacy of the model for real urban applications and for forecasting at a later stage.

3. Experiments with the Model

This part focuses on experiments with the model implemented on the trial datasets. Three objectives will be examined in this part: 1) some results will be produced which support the various model concepts employed, 2) the simulation results will also be used for connecting the model to real world data, and 3) some quantitative outcomes which can be used as forecasts, will be generated. As mentioned so far, the model is driven by a set of spatial data in a raster format where pre-defined parameters are

used to control the processes of the simulation and various random behaviours that are integral to the way locational transitions occur. For this reason, any errors in processing within the model are hardly likely to be detected while the model is actually running. The results gathered from the model by controlling and tweaking these parameters are therefore essential in visualising the progress of the simulation. In this case, the first objective is to ensure that all the crucial procedures operate accurately. These include the workings of the general and constrained cellular automata, the generation of dynamic maps, the hierarchical node generator, and the computation of the two centripetal and centrifugal forces. The second objective involves gathering various sequential simulation results from the model so that we can visualise changes to the urban areas which we hope should match what happens in the real world as the region begins to urbanise. Finally, the model can yield other dynamic data useful for assessing urban population trends and quantitative changes in land use/cover.

To set up the experiments, we first fix the initial state of urban areas to ensure that all results obtained are comparable. We then adjust various model parameters that control the simulation processes. All-important parameters are briefly described and this then enables us to test different objectives with respect to these simulation experiments.

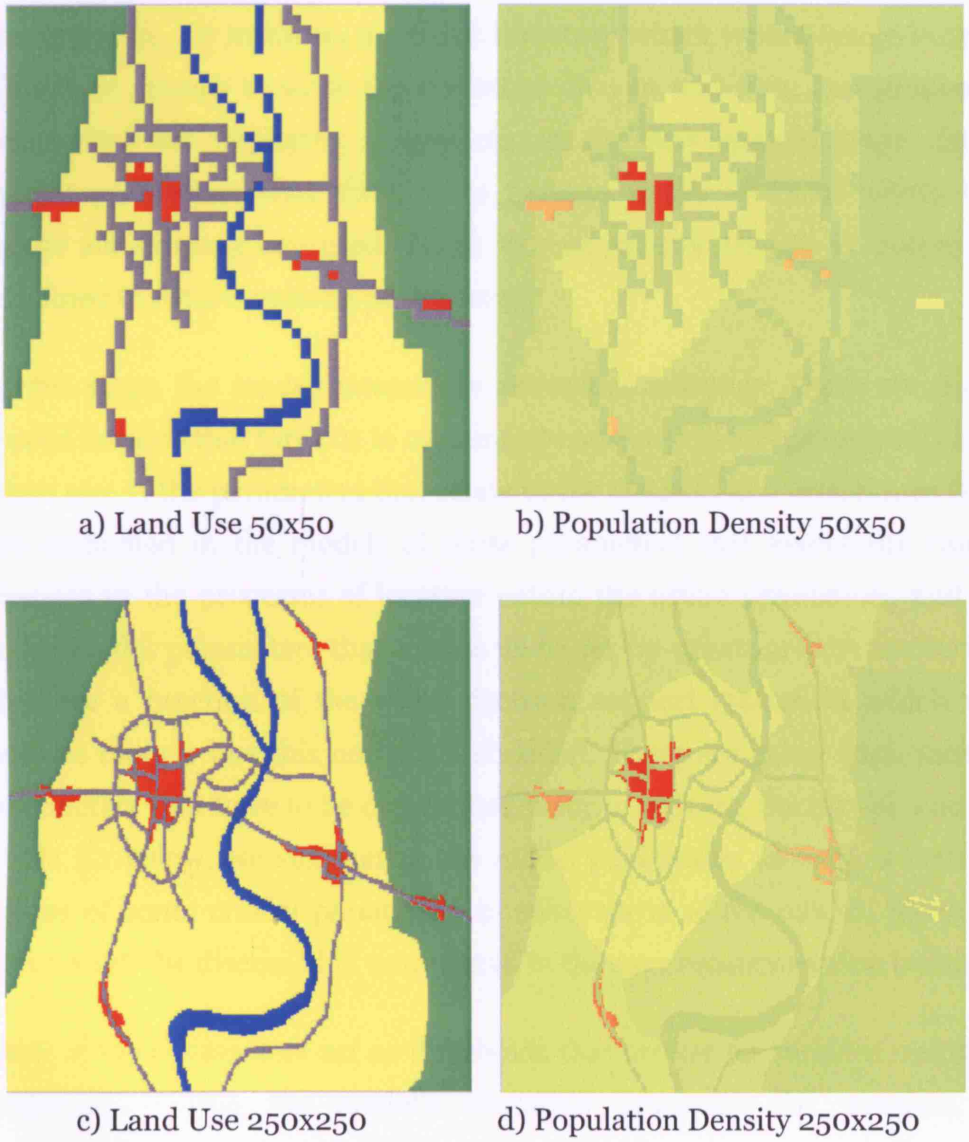


a) 50x50 b) 250x250

Figure 5.14 The Initial States at the Two Scales of Urban Area

The Initial State

The previous section described the data used in the model with most of these map layers set as part of the initial state for testing the simulation. Only the urban seed layers were defined by the user as shown in Figure 5.14 and once this was done, the various maps were generated as shown in Figure 5.15. The land use/cover and population density maps were then created as well as the node maps which are also shown in Figure 5.16 below.



a) Land Use 50x50 b) Population Density 50x50

c) Land Use 250x250 d) Population Density 250x250

Figure 5.15 The Initial State of Land Use and Population Density

According to Figure 5.15, there is one large city in the centre of the space which is surrounded by a series of small towns. This city has the highest population density compared to the surrounding towns which are shown as colour gradations in Figures 5.15b and5.15d and as high order nodes in Figures 5.16a and b.

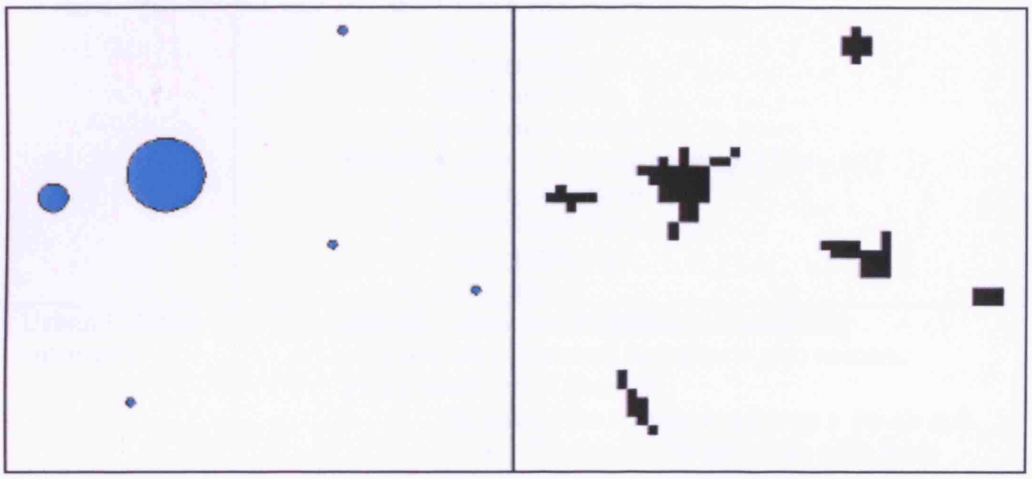
The Model Parameters

To develop the simulation, two kinds of parameter need to be considered, namely the system parameters and the model parameters. System parameters are defined in the model to control flows of information within the program, for instance, a default directory which writes image output files, time stamps to write the statistical data to text files, and graphical panel sizes for displaying images etc. In the development stage, these system parameters were fixed so as to ensure that all flows within the model are properly executed. These system parameters can of course be redefined in future versions of the model.

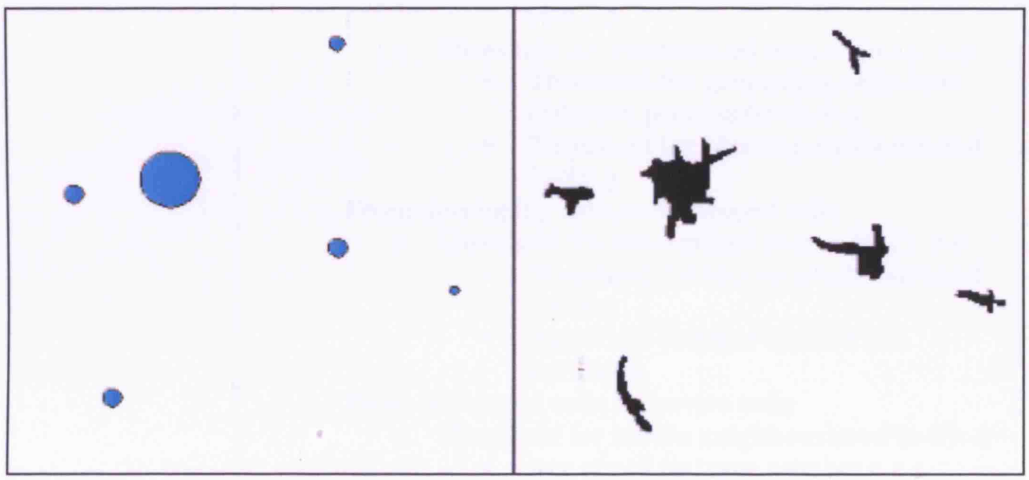
At this stage, the model parameters are rather arbitrary. There are three crucial reasons that force us to concentrate on these model parameters and these are: 1) the parameters that relate to the theories and techniques that are embodied in the model; 2) those parameters that essentially cause changes in the processes of location within the urban simulation; and 3) those model parameters that enable us to set up urban growth scenarios that are a function of the wider decision support system in which we assume models like this one are embedded. There are many such model parameters that have to be determined prior to running the model, and in Table 5.1 below, we summarise the model parameters as well as setting values of some crucial parameters in other parts of the model. Some of these would be discussed in more detail in the experiments section below.

Most of the parameters act as thresholds that are set for random number generation during the model's allocation processes; for instance, a threshold for transforming cells that do not have development (i.e. are dead) to cells that need to be developed, distance thresholds defined in

order to generate new nodes in the space, and so on. Some other important parameters are weighting numbers defined for each map layer which are involved in performing dynamic map overlay, relative densities for each land use/cover which are used to compute the dasymetric population density maps, and weights identified for creating the urban land demand map. These parameters are organised to facilitate different simulation scenarios as we discuss later in the experimental section.



a) Nodes and Urban Areas 50x50



b) Nodes and Urban Areas 250x250

Figure 5.16 The Initial States for the Node Maps Contrasted with Urban Areas

Table 5.1 Model Parameters

Model parts	Parameters
Raster Map	- Numbers of row (integer number): 50, 250 - Numbers of column (integer number): 50, 250 - Resolution (integer number): 200, 50
Diffusion	- Threshold range 0-1 to generate diffused cells on space (0-1): 0.2 - Maximum number of cells generated randomly in one time (integer number): 5 - Maximum number of periods to diffuse defined randomly (integer number): 20
Dynamic map	- Weight scores 1 for map overlay: ($\sum w = 10$) o Centripetal force: 2 o Centrifugal force: 2 o Accessibility: 3 o Topography: 2 o Land use: 0.75 o Land demand: 1 - Weight scores 2 for map overlay: ($\sum w = 10$) o Centripetal force: 2.5 o Centrifugal force: 2.5 o Topography: 3 o Land use: 2
Urban Cellular automata	- Search radius for computing accessibility probabilities (distance in meter): 300 meters. - From dead cell to live cell o Threshold for constrained map 1 (0-1): 0.6 ▪ Threshold for generate cells from diffusion process (0-1): 0.5 ▪ Threshold for Moore neighbourhood (1-8): 2 o Threshold for constrained map 2 (0-1): 0.9 ▪ Threshold for generating cells from diffusion process (0-1): 0.5 ▪ Threshold for Moore neighbourhood (1-8): 3 - From developing cells to developed cells o Threshold for constrained map 1 (0-1): 0.6 ▪ Threshold for Moore neighbourhood (1-8): 4 ▪ Age of cell (integer number): 4 iterations - From developed cells to survive cells o Threshold for Moore neighbourhood (1-8): 4 ▪ Age of cell (integer number): 4 iterations - From developed cells to developing cells o Threshold as random number that shows probability to decrease the level of development (0-1): between 0.04 to 0.3
Land Use/Cover changes: Forest and Agriculture	- Threshold if majority neighbouring cells are agriculture (0-1): 0.1 - Threshold if majority neighbouring cells are urban (0-1): 0.3

Model parts	Parameters
Node and Hierarchy	- Threshold distance for separating nodes (distance in meters): 1000 m. - Threshold distance for choosing node to interpolate surface map (distance in meters): 5000 m.
Population density	- Relative density for calculating dasymetric map: ($\sum r = 1$) o Urban: 0.85 o Agriculture: 0.1 o Forest: 0.05
Urban land demand	- Weight for calculating demand for each land use if there are urban areas existing o Urban: 0.8 o Agriculture: 0.2 - Weight for calculating demand for each land use if there are no urban areas existing o Agriculture: 0.6 o Forest: 0.1
Population growth	- Random population growth rates for each administrative area: 0 - 0.1

The Experiments

At this stage, several trial simulations were performed by assigning different sets of parameters to cover the objectives stated earlier. Firstly, experiments relate to the procedures for unconstrained cellular automata focusing on the results from small-scale images. As mentioned in Chapters 3 and 4, changes to urban cells in the urban space are normally based on the diffusion and CA processes. This first experiment concentrates on the simulation and its visualisation of how urban cells can be generated within the urban space, ignoring all the constrained layers and any diffusion process. Secondly, experiments were implemented in same fashion as in this first one, but including the diffusion process so that we might observe how cells not connected to other cells in space could be generated. Thirdly, experiments applying all the map layers to compute the constrained probability layers which constrain the CA process were developed. In this third case, the urban growth process was then input to the constrained layer which was created dynamically using weighted overlay methods. In addition, results from both the second and third experiments were then comparably represented to observe the differences between natural and constrained urban growth. Focusing on the constrained map, the updated constraint maps constructed from the dynamic map layers will be

discussed in this section. Cells of some map layers are directly obtained and updated from the CA simulation; some are constructed by collecting the mutated cells and recomputing the map layers based on the relevant map algebra. This section visualises all changes in these maps and examines the interrelationships between these maps and urban cells generated by the CA simulation.

The next two sets of experiments connect all urban cells in the model space to the generation of a unique nodal system. At the micro-scale, a central cell can interact with its neighbours on several levels, at several distance bands, for instance: with 4 neighbours, 8 neighbours, and so on with varying number of neighbours at a distance. On the other hand, at the regional scale, a city can relate to others in rather different ways. Instead of examining neighbourhoods that depend on distance, the density and size of cities is used to compute the nodal structure which is then visualised using symbols with a ranking system which can be generalised into a more abstract visual image; this map provides a much more intuitive view of the size of and distance between centres. Finally, the spatial interaction approach was explored where the more abstract centralising and decentralising urban forces were transformed into visible pictures as potential surface maps derived from weighted spatial interaction. We show the impact of dynamics with different parameters. We are now in a position to report all these experiments and we will do so in the following sections.

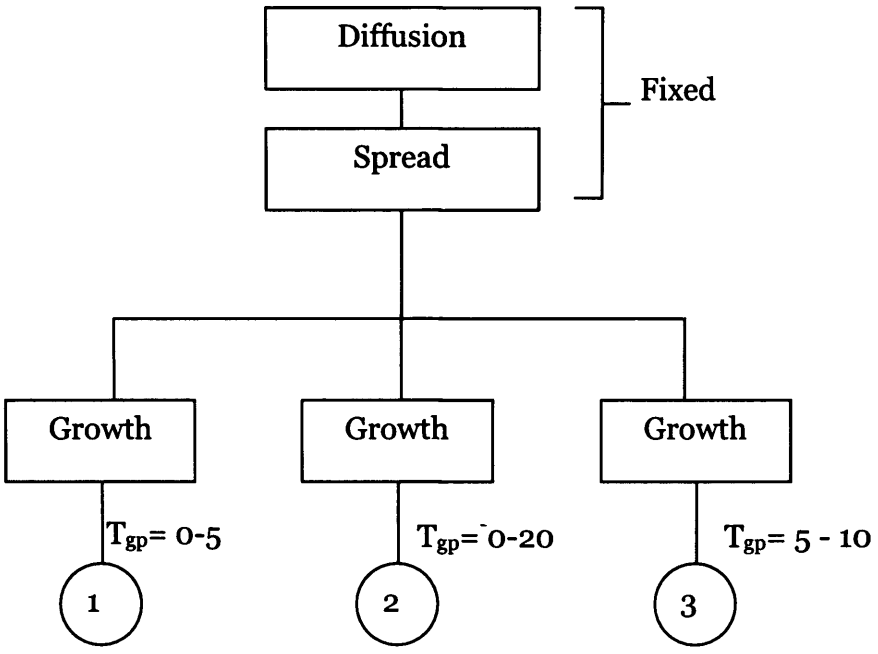
Cellular Automata Simulation without any Diffusion Process

Before getting into the full set of modules which compose the model, this first trial only focuses on how the cellular automata processes work under different parameter values. Excluding the diffusion process makes it easier to observe the undistorted operation of the CA processes conceptualised in this study. Independently, this section also discusses two different values for the model parameters associated with first the developing period of the simulation, and then the neighbourhood effect. First, the developing period is affected by only one parameter that controls the CA processes

generating all cells in the space. The second trial relates to the influence of changing rules on the number of neighbouring cells used in making transitions from non-urban cells to urban.

The Developing Period

In this case, the parameter defining the possible range for the developing period which is the time to transform a developing cell to its survival as a developed cells is fixed from 0 to 5 or from 0 to 20 and so on. The model then generates a random number based on this definable range and cells would begin to develop and ultimately survive when the period defined by this random number was reached. For trials, three different scenarios were set up defining different possible developing periods as shown in Figure 5.17: 0 – 5, 0 – 20 and 5 - 10. The number shows how long a developing cell (grey cell) would take to be transformed into a developed cell (white cell).



T_{gp} = Possible period of urban cells to be developed cells

Figure 5.17 Different Scenarios for CA Simulation

1. *The Short Developing Period: 0 – 5*

Ignoring the diffusion process and starting with a small number of seeds, cells tend to grow in circle-like fashion as shown in the images in Figure 5.18. The developing period ranges from 0 to 5, with the actual period fixed by the random number. This means that some cells generated in the urban space need very little time to grow the highest level while on the other hand, some cells need more time to grow to become developed (white) cells. Generally if there are no constraints on development or if the urban area can grow in every direction, the circle shape is the perfect form in such development. As shown in Figure 5.18, from iteration 50 to 200, the shape of the urban area grown from the seed obviously approaches a symmetric circle. Nevertheless, the urban shape is never the perfect circle because of the randomised urban generation which drives the entire CA process.

2. *The Long Developing Period: 0 – 20*

Figure 5.19 illustrates snapshots of the same initial seeds with a wider range for the developing period from 0 to 20. Changes within this range simply change the time taken for cells in the urban space to become developed with some cells the taking a longer time to develop, while some transforming into developed cells almost immediatedly.

In terms of the morphology produced, these cells also create a circle-like urban shape, but the processes of growth are slower. Comparing the images and graphs in Figures 5.18 and 5.19 at the same time state, there are clear differences between the two simulations. As can be seen in Figure 5.19, grey cells generated from dead cells by the CA transition rules were more difficult to convert into the white cells in the longer developing period than those which developed in the shorter time period. This is not so difficult to explain as it relates to greater opportunity for a cell to be converted to urban in the longer period due to the random number generation and it implies that if the uncertainty is greater and the development process longer, then it is more difficult to grow: shape is thus

influenced by a greater degree of historical accident or path dependence. The next trial give more obvious results pertaining to this issue and the further comparisons also depicted in Figure 5.21 are to be discussed later.

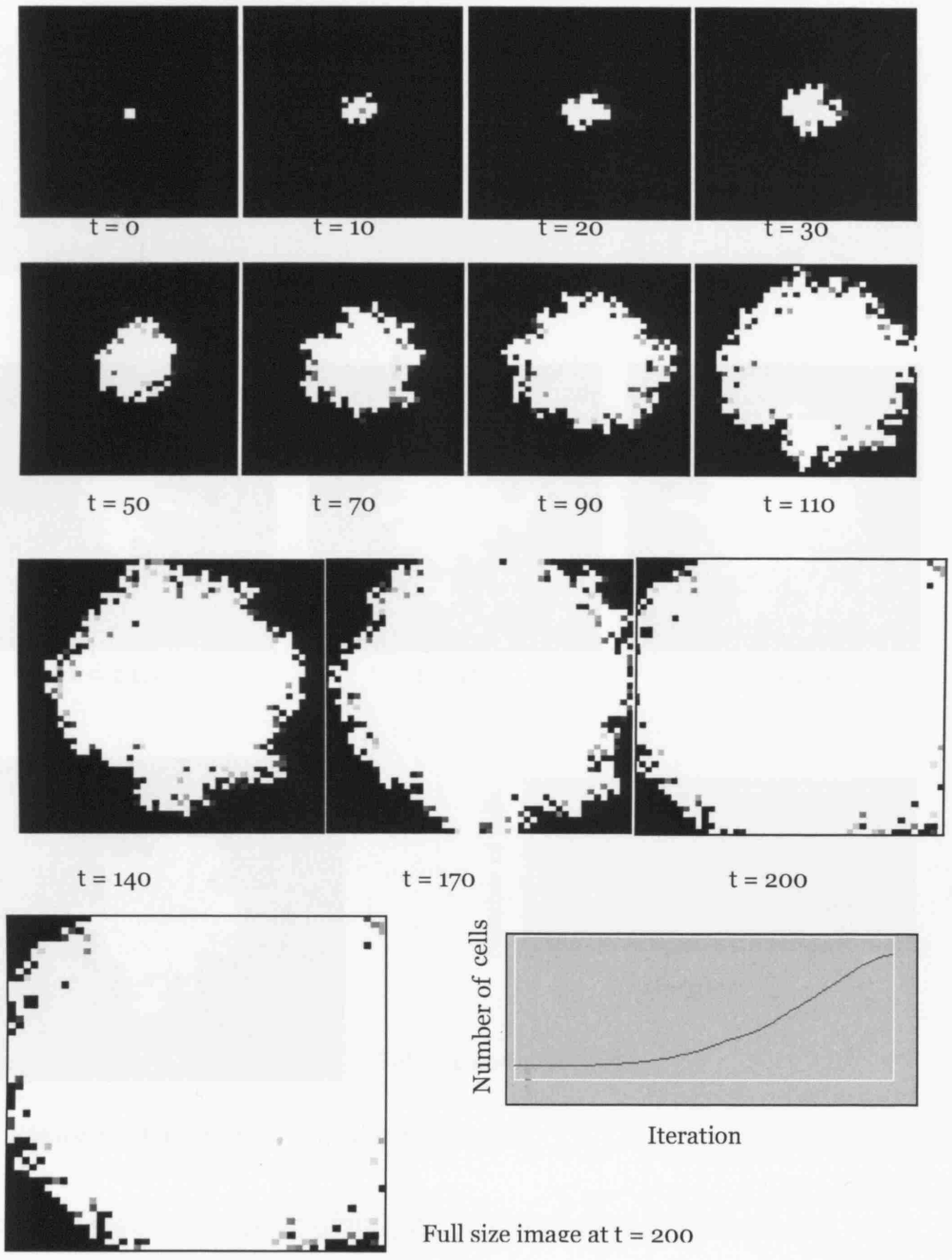


Figure 5.18 Snapshots of the Scenario 1

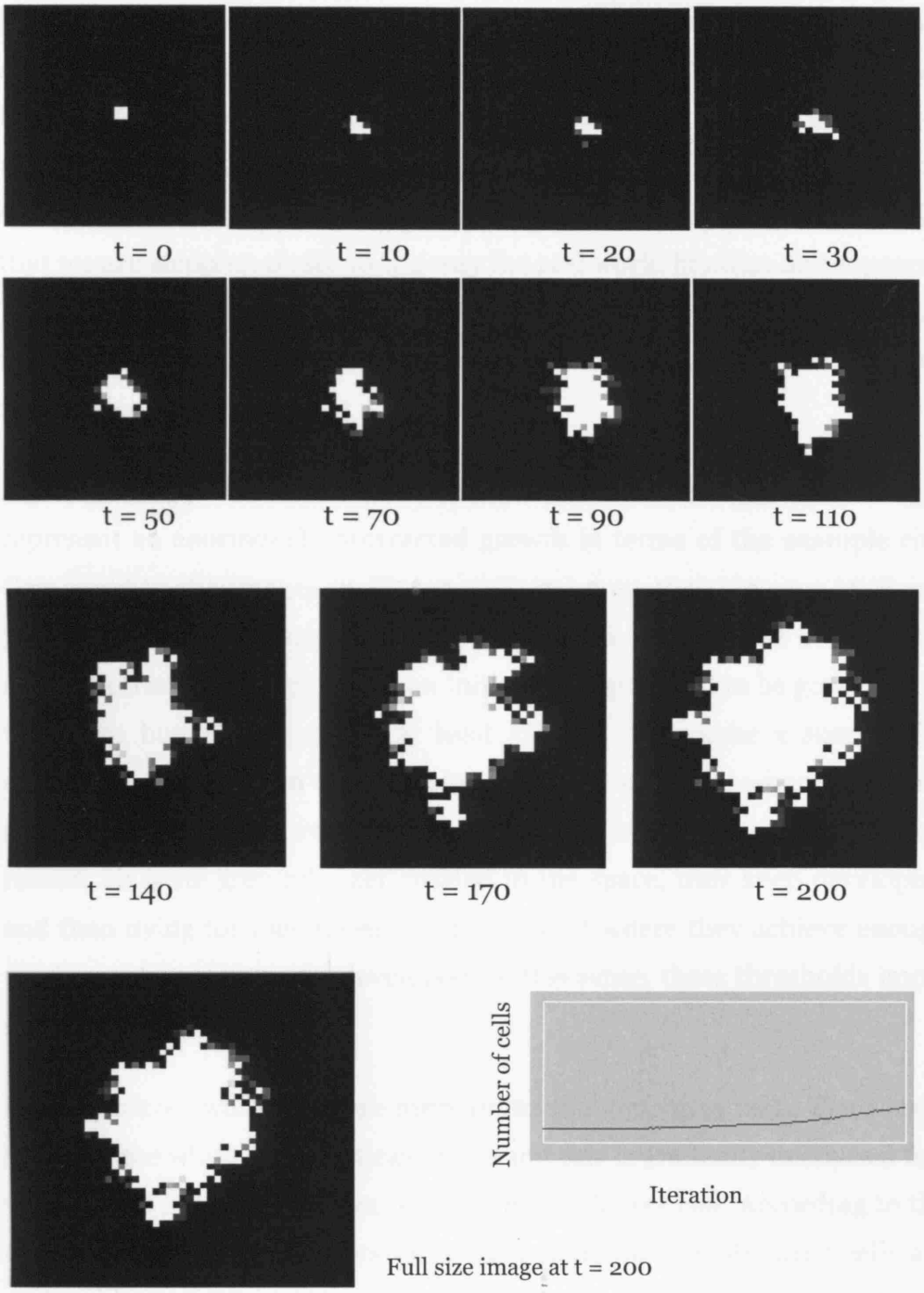


Figure 5.19 Snapshots of the Scenario 2

3. The Lower and Upper Limits on the Developing Period: 5 – 10

The first two scenarios represent how easy or hard it is for cells to be successfully transformed as urban cells or 'white cells'. In this trial, instead of a lower limit of zero, this is set as 5 and it has an important effect on the

process of transformation. It makes it much trickier to turn a grey cell into a white developed cell. Each cell needs to be transformed at least 5 times by the CA rule to get any development; some need up to 10 before they make the transformation. All we are doing here is introducing mechanisms that make it hard for cells to become urban but as we do so, we consider that we are stepping closer to the way the real world handles development. In essence all cells need some sequential period over which to make the transformation from non-urban to urban and the longer this is the harder it is to become urban.

Figure 5.20 depicts the results of the 5:10 scenario; the sequential images represent an enormously protracted growth in terms of the example city. Compared to the images in Figure 5.18 and 5.19, these images in Figure 5.20 show more existing grey cells during the process than the other two. A set of CA transition rules forces an initial developing cell to be generated in the space but the cell needs at least 5 states to become a successfully developed cell, unlike in the case of the first two scenarios where cells have a much shorter period over which random generation takes place. For this reason, as some grey cells get created in the space, they keep developing and then dying for they never reach the point where they achieve enough successive 'hits' to remain developed. In this sense, these thresholds imply some sort of fitness criteria.

In comparison, white cells are more influential than grey cells. The weight score for the white cell is defined as 10 and this is gradually decreased to 1 which is the lowest weight score for a very dark grey cell. According to the CA transition rule in this model, in each state, the neighbouring cells are counted by weighting according to these scores. Hence, when central cells have many neighbouring white cells (each with a high score), they have a high probability of developing as urban. On the other hand, as soon as grey cells become white cells, this means that neighbouring dead cells have a higher probability of becoming urban cells in the next time period.

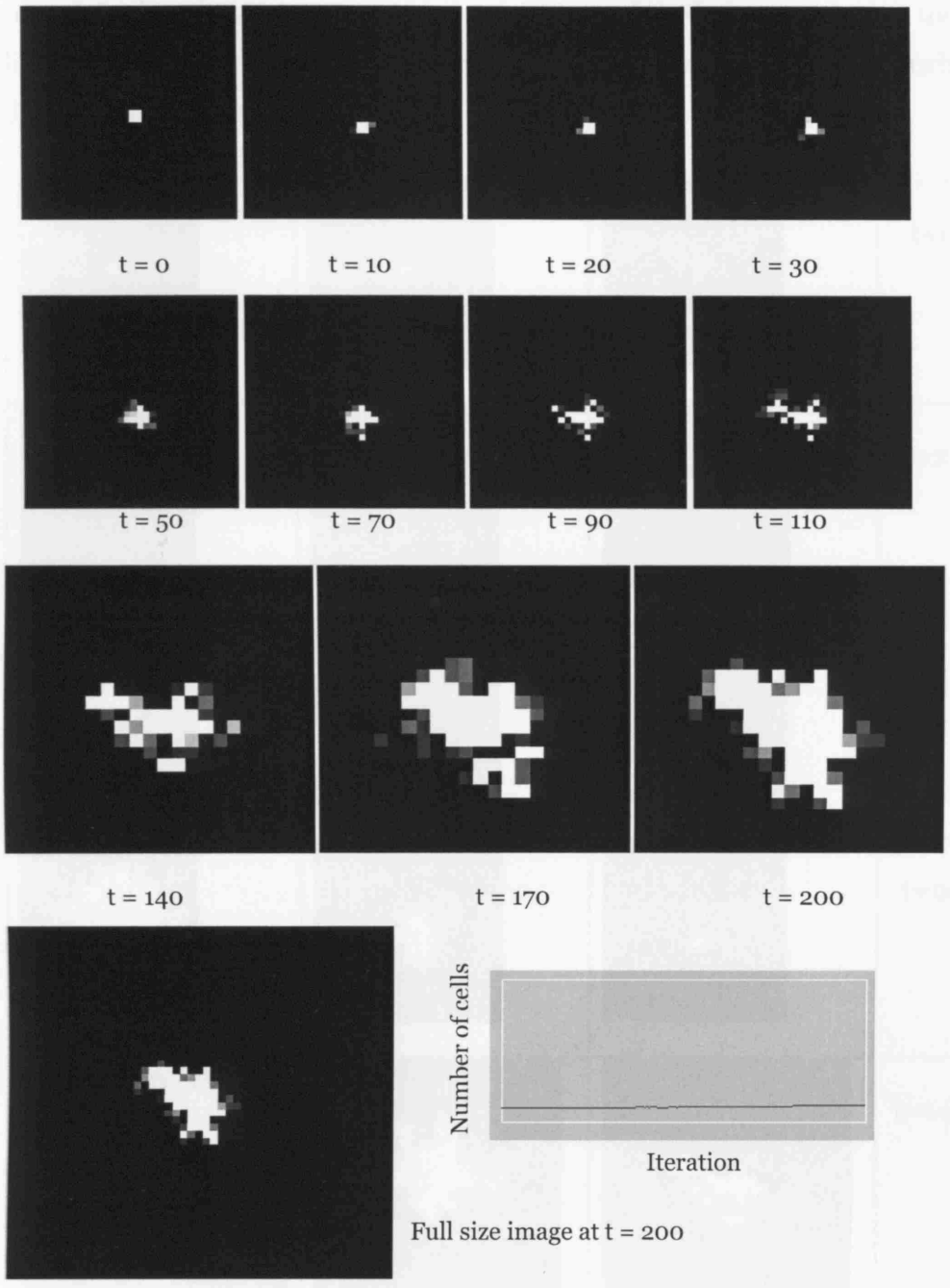


Figure 5.20 Snapshots of the Scenario 3

Figure 5.21 makes the comparison between various snapshots of the simulation results extracted from these three scenarios. These images obviously show the differences at each time state; the short developing period (0-5) causes faster growth faster with few grey cells remaining between the time instants. As we noted if a large number of white

(developed) cells appear in the space, this makes their neighbours more likely to become developing cells and make the transition to urban development in next time state.

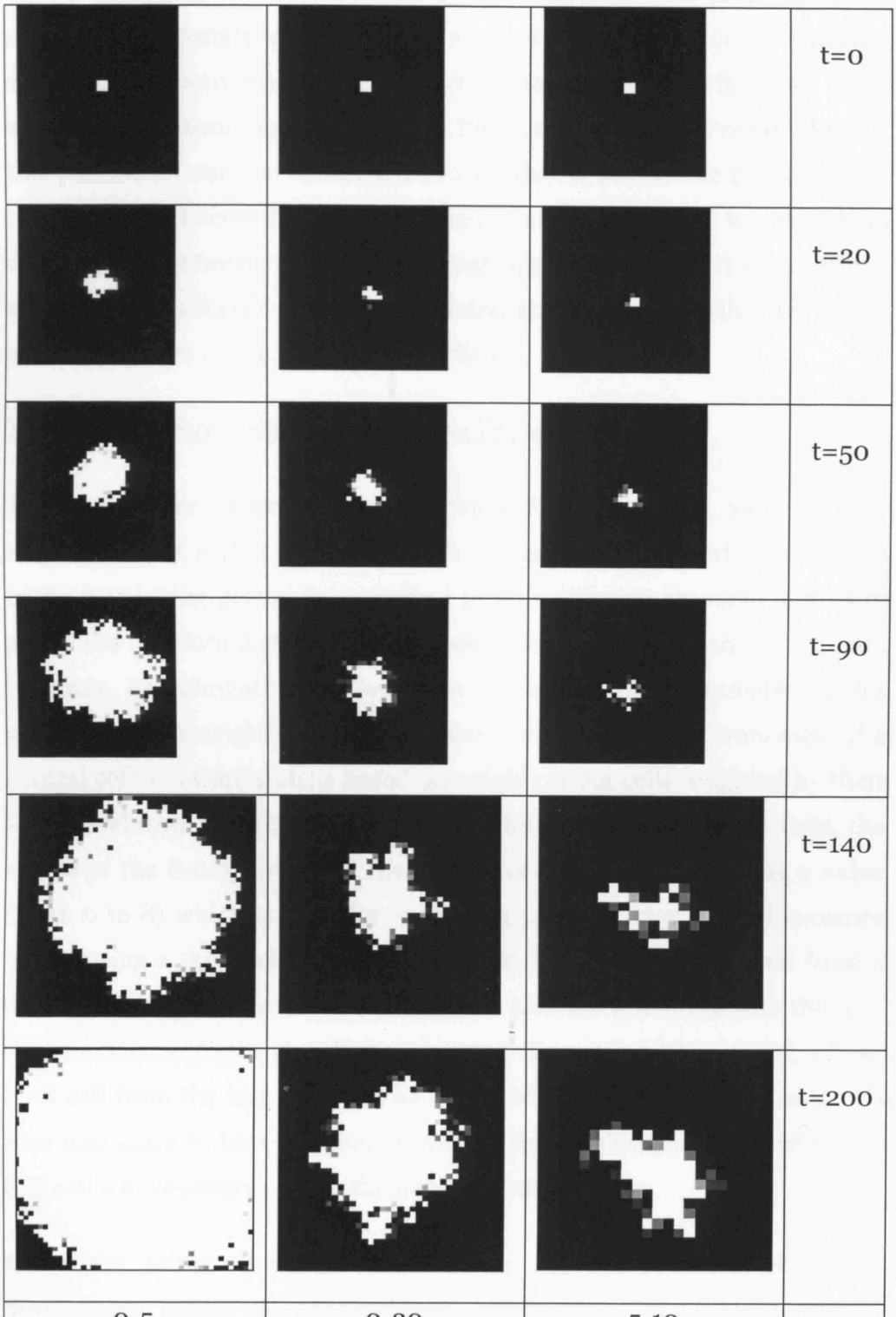


Figure 5.21 Comparisons of the Simulation Processes

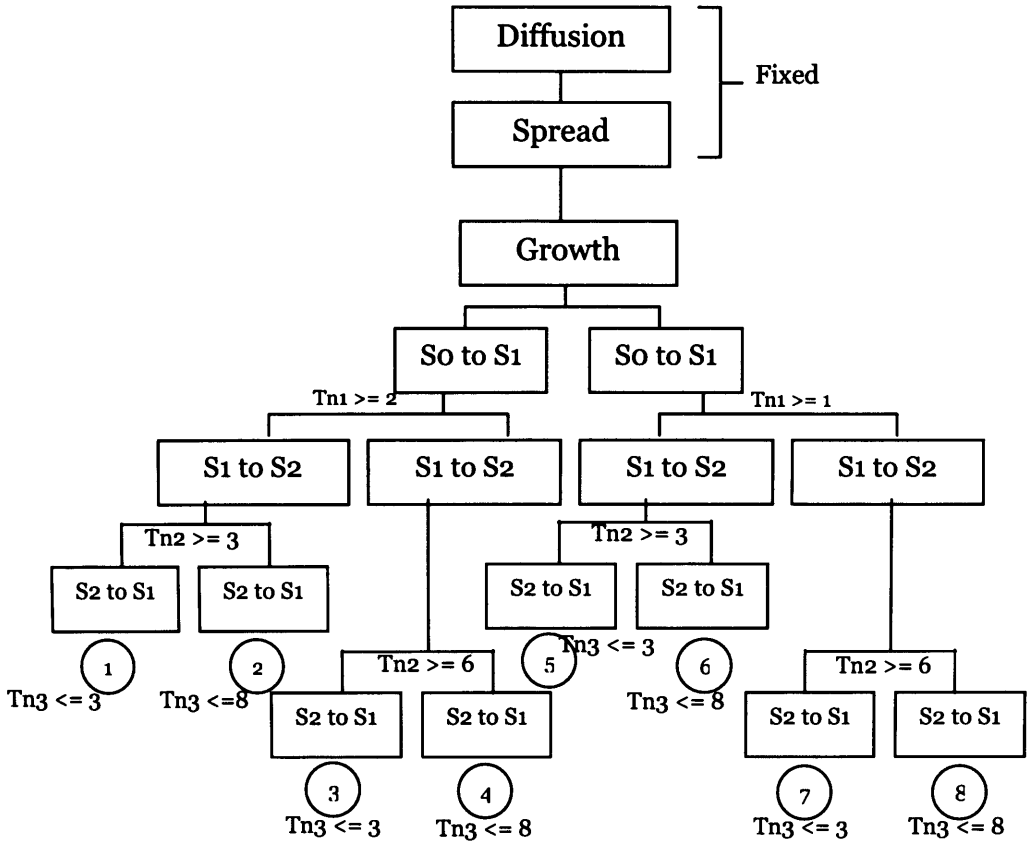
In conclusion, the developing period parameter is used to control the rate of urban growth, and it can be applied at two levels in the model: the global and the regional. In case of its being used as a global variable, this affects all cells in the simulation space. This is useful in practice for generating a variety of scenarios in an urban system or for making comparisons between two or more urban systems with the same developing scenario, for instance. On the other hand at the regional scale, this parameter can also be applied to a particular part of the urban system – in our terms here within a constrained boundary – so that we are able to discriminate between growth rates that apply to each part of a unique urban system; all cells are then simulated at different growth rates which as clearly shown above, yields quite different results.

The Neighbourhood Effect: Varying the Transition Rules

There are four important components for simulation with cellular automata: cells, states, transition rules and neighbours. Altering the length of the developing period is equivalent to how different amounts of growth affect the CA simulation. At this point, we need to change tack and illustrate experiments that focus on changes in the transition rules associated with neighbourhoods. As mentioned before, the transition of a central cell in a time state is based on neighbouring cells weighted by their score level from 1 to 10 (level 10^{th} is the white cell); at each time state, the scores of the 8-neighbours of the central cell are averaged giving a value (from 0 to 8) which is used for evaluating whether or not a cell becomes urban using a threshold rule. For instance, if the value calculated from a dead cell's neighbours is more than 5, the cell will transform into the first level of urban cell (the so-called grey state). However, in this model, a first-level cell from the last state will be activated as a developing cell and it is thus necessary to have another transition rule to evaluate whether or not this cell will develop or die in the next time state.

From the initial state of seed-cells in the space, there are three development stages that are used in the application of the transition rules. These stages are: 1) from dead (non-urban) cells to first-level (developing)

urban cells, 2) from developing first-level (grey) cells to developed white cells, and 3) from developed cells to a lower level – developing or dead. In experimentation, the developing period for every simulation was defined to be 0-5. Each development stage was set as two scenarios for visualising outcomes.



So = Dead cell or blank state
S1 = Developing state
S2 = Developed state
Tn1: So to S1
Tn2: S1 to S2
Tn3: S2 to S1

Figure 5.22 Different Scenarios for the Neighbouring Effect

The diffusion and spread processes were still discounted in this process of testing the CA mechanism. Generally, with 8 possible neighbours defining 3 levels of development in the simulation, there are 512 possible experiments can then be set up. However, it is impossible to trial all these possibilities and report all the differences between them. Accordingly we have sampled this space and chosen those combinations of values that we

consider give the most significant differences in transition. We show these in the constrained tree of possibilities in Figure 5.22 which show how different combinations of parameters generate 8 different model experiments which we will now discuss in turn.

1. 2 Ns for Birth, 3 Ns for Developing and 3 Ns for Decline

The transition rule in this stage was set shown in Block 5.1 below. As mentioned earlier, there are three processes to generate cells on the space at the same time state. Block 5.1 represents rules to transform from different stages: from dead cells to initial state of developing cells, progression of developing cells and becoming developed cells (Ns in the block represent neighbourhoods). Figure 5.23 represents snapshots of the simulation. As seen in the image at t = 20, cells surrounding the first four initial seed cells were constructed due to their need for at least 2 neighbouring cells according to the transition rule; some of them were still in the developing state, but some had already became developed white cells. Cells were born, survived and died based on the predefined transition rules as seen in other images in the figure.

From dead cells $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 3; \text{random} \\ 1- : u_t = 1; \omega_2 < 3; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 3; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 3; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.1 Functions for B: 2Ns, Dev: 3Ns, and Dec: 3Ns

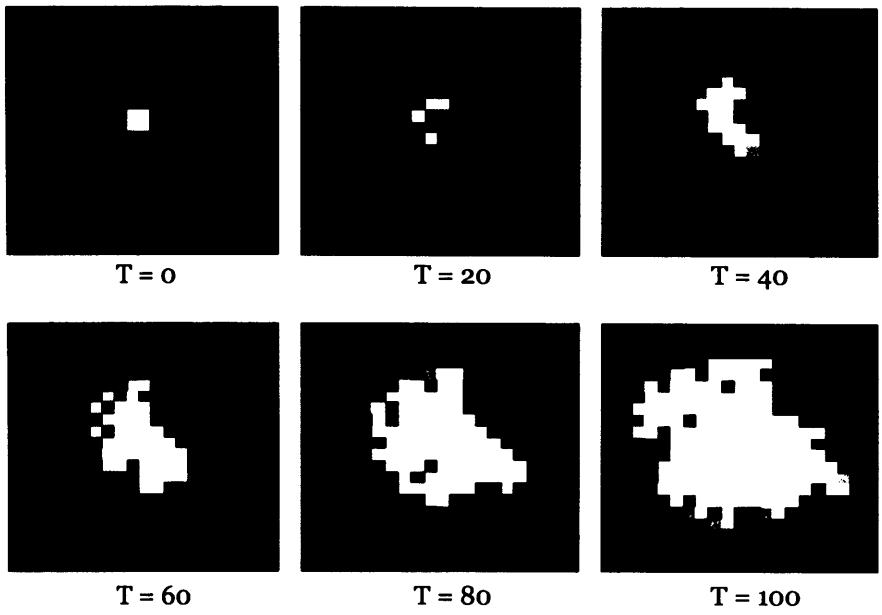


Figure 5.23 Snapshots of B: 2Ns, Dev: 3Ns, and Dec: 3Ns

2. 2 Ns for Birth, 3 Ns for Developing and 8 Ns for Decline

The parameter controlling the number of neighbours needed before a developed cell moved to a developing or dead state, was set at 8 neighbours. That means the probability of decline based on counting the neighbouring cells was changed from approximately 35 percent to 100 percent (from 3 Ns to 8 Ns) for the maximum numbers of neighbouring cells is equal to 8 in Moore neighbourhoods. Once a random number generated for a developed cell reaches the threshold based on how many iterations have passed since the cell was created, the cell will mutate from a white to grey cell (change to '1-' state), from developed to developing. Block 5.2 below represents the transition rules for this trial and Figure 5.24 depicts some sequential images that are generated by the simulation that are rather different from the previous trial.

During the simulation with this parameter set, we observe differences from the first trial in that more developed white cells change their status to the developing grey state. Some of them decline to the point where they become dark grey and then turn back into the white cells when all neighbourhood conditions for development are matched (see images at t = 100 of Figure 5.23 and 5.24).

From dead cells $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 3; \text{random} \\ 1- : u_t = 1; \omega_2 < 3; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 3; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 8; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.2 Functions for B: 2Ns, Dev: 3Ns, and Dec: 8Ns

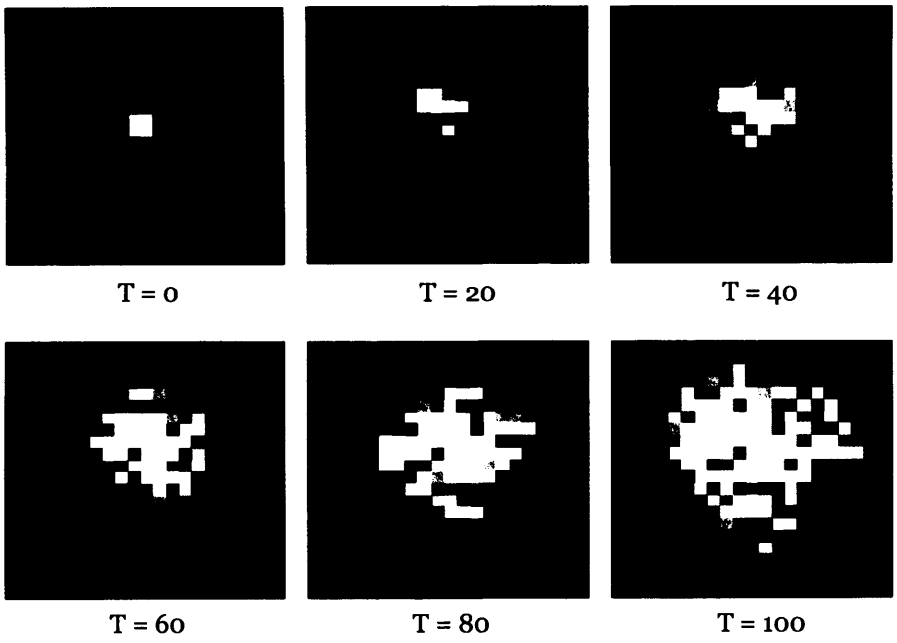


Figure 5.24 Snapshots of B: 2Ns, Dev: 3Ns, and Dec: 8Ns

3. 2 Ns for Birth, 6 Ns for Developing and 3 Ns for Decline

In this trial the neighbourhood threshold changes again increasing the development level from 3 to 6 for a transition to take place. While a cell is in a developing state (or a grey cell), if the neighbourhood function returns a count more than or equal to 6 with each cell reaching a random threshold, the cell will be raised one more level ('1+' state in Block 5.3). The associated probability is lower than in the first example and again, we observe similar results extracted from each time state of the simulation which we show below in Figure 5.25. Although the results look similar to the first example, the white cells in Figure 5.25 group around the large city centre. Due to the fact that at least 6 neighbours are required for grey cell development, only grey cells which have more than 6 neighbours will be evaluated but of course, this cannot be met on the edge of an urban simulation space.

From dead cells $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 6; \text{random} \\ 1- : u_t = 1; \omega_2 < 6; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 6; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 3; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.3 Functions for B: 2Ns, Dev: 6Ns, and Dec: 3Ns

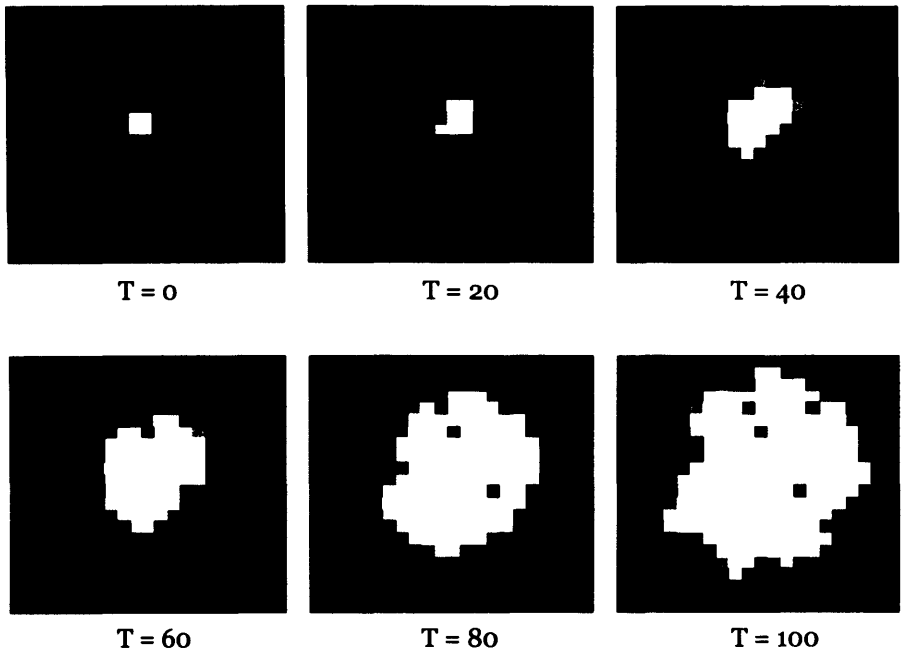


Figure 5.25 Snapshots of B: 2Ns, Dev: 6Ns, and Dec: 3Ns

4. *2 Ns for Birth, 6 Ns for Developing and 8 Ns for Decline*

From dead cells

$$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells

$$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 6; \text{random} \\ 1- : u_t = 1; \omega_2 < 6; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 6; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells

$$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 8; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.4 Functions for B: 2Ns, Dev: 6Ns, and Dec: 8Ns

This example mirrors the idea that it is 'difficult for a cell to develop but easy to decline'. The transition rules are shown in Block 5.4 below. As shown in the image results from the previous simulation in Figure 5.24 when assigning an 8-neighbour threshold to trigger decline, more white cells in the model space became grey cells. In the same manner as illustrated in the images in Figure 5.25, white cells should agglomerate, but due to the strict nature of the threshold governing decline, this means that more white cells disperse than in the previous examples as we show in Figure 5.26.

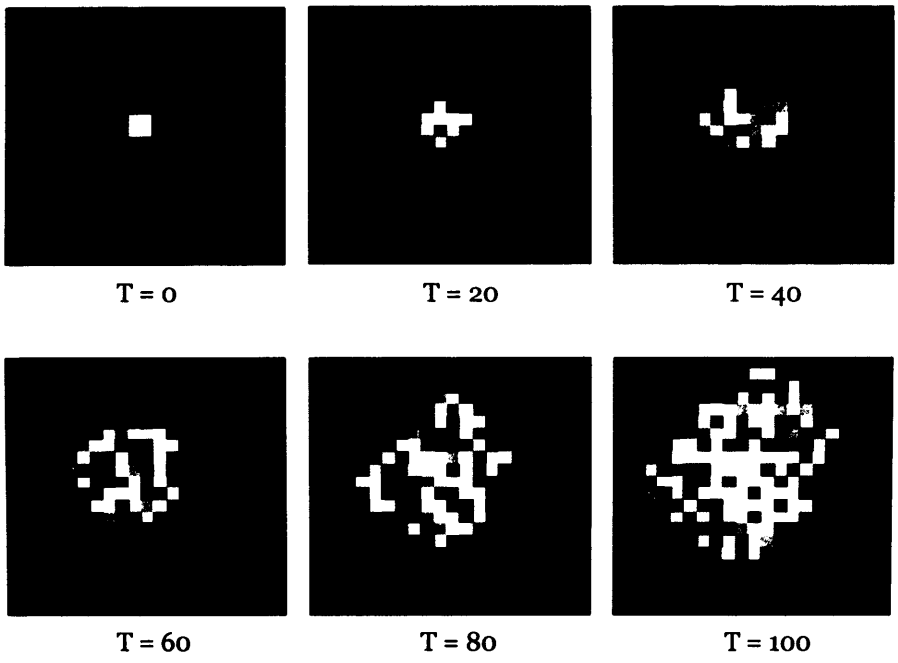


Figure 5.26 Snapshots of B: 2Ns, Dev: 6Ns, and Dec: 8Ns

The last four experiments (5 to 8) which we will now illustrate were implemented by simply changing the first parameter for the neighbourhood threshold associated with the birth of an urban cell from 2 to 1 neighbours. This increases the probability for growth in general particularly for all cells that have at least one neighbouring cell surrounding them. We used the same sets of parameters for the other processes as for the four cases just outlined, the only difference being that cells grow faster because of the lower kick start threshold. We will outline each of these four new cases below but to anticipate the results, Figures 5.31a to 5.31h illustrated later represent comparisons between the results

from this section. In the four cases which follow, we will not provide any commentary for the results speak for themselves. All we will do is set the parameters in each block as above and then illustrate results which pertain to the transitions between cells at each key time period.

5. 1 Ns for Birth, 3 Ns for Developing and 3 Ns for Decline

From dead $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 1; \text{random} \end{cases}$$

Developing $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 3; \text{random} \\ 1- : u_t = 1; \omega_2 < 3; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 3; t - t_u \geq p; \text{random} \end{cases}$$

Developed $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 3; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.5 Functions for B: 1Ns, Dev: 3Ns, and Dec: 3Ns

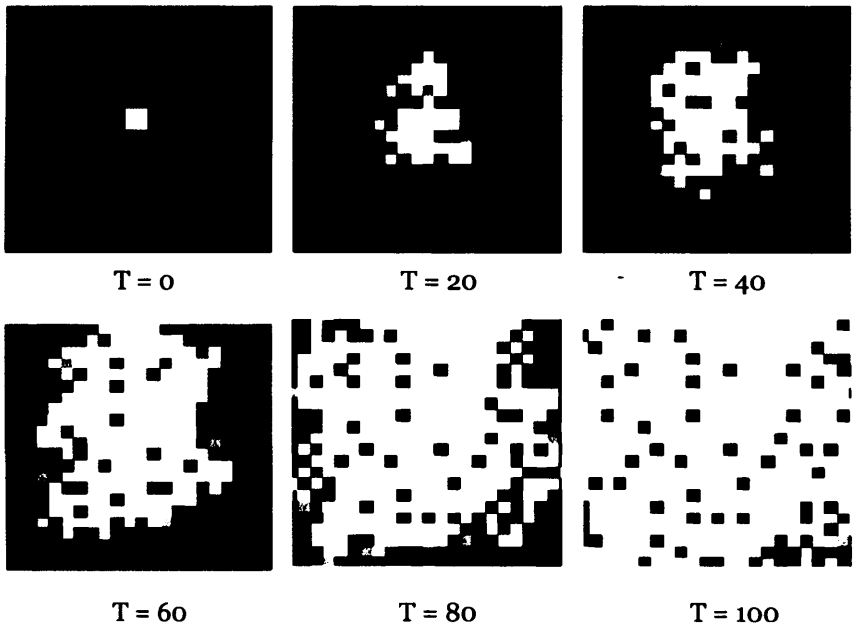


Figure 5.27 Snapshots of B: 1Ns, Dev: 3Ns, and Dec: 3Ns

6. *1 Ns for Birth, 3 Ns for Developing and 8 Ns for Decline*

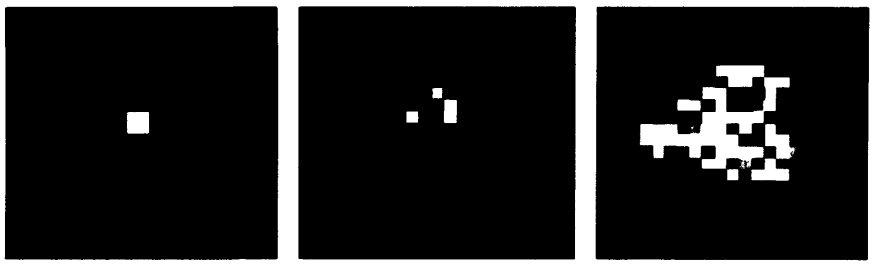
$$\text{From dead} \qquad u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 1; \text{random} \end{cases}$$

$$\text{Developing} \qquad u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 3; \text{random} \\ 1- : u_t = 1; \omega_2 < 3; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 3; t - t_u \geq p; \text{random} \end{cases}$$

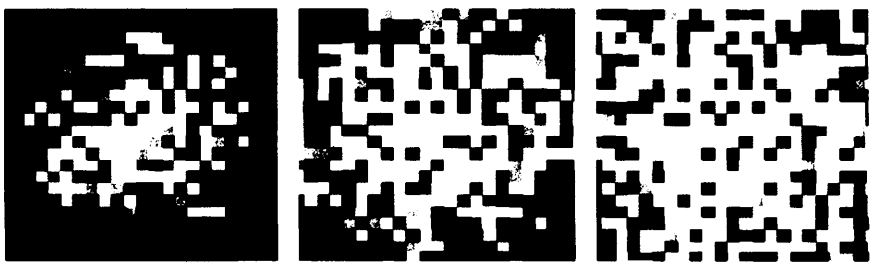
$$\text{Developed} \qquad u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 8; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.6 Functions for B: 1Ns, Dev: 3Ns, and Dec: 8Ns



T = 0 T = 20 T = 40



T = 60 T = 80 T = 100

Figure 5.28 Snapshots of B: 1Ns, Dev: 3Ns, and Dec: 8Ns

7. *1 Ns for Birth, 6 Ns for Developing and 3 Ns for Decline*

From dead
$$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 1; \text{random} \end{cases}$$

Developing
$$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 6; \text{random} \\ 1- : u_t = 1; \omega_2 < 6; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 6; t - t_u \geq p; \text{random} \end{cases}$$

Developed
$$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 3; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.7 Functions for B: 1Ns, Dev: 6Ns, and Dec: 3Ns

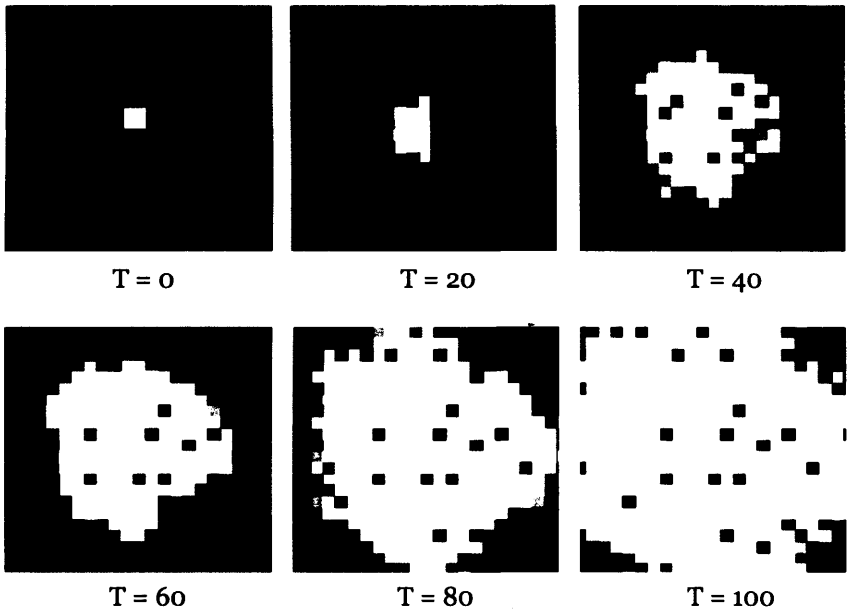


Figure 5.29 Snapshots of B: 1Ns, Dev: 6Ns, and Dec: 3Ns

8. *1 Ns for Birth, 6 Ns for Developing and 8 Ns for Decline*

$$\text{From dead} \qquad u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 1; \text{random} \end{cases}$$

$$\text{Developing} \qquad u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 6; \text{random} \\ 1- : u_t = 1; \omega_2 < 6; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 6; t - t_u \geq p; \text{random} \end{cases}$$

$$\text{Developed} \qquad u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 8; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.8 Functions for B: 1Ns, Dev: 6Ns, and Dec: 8Ns

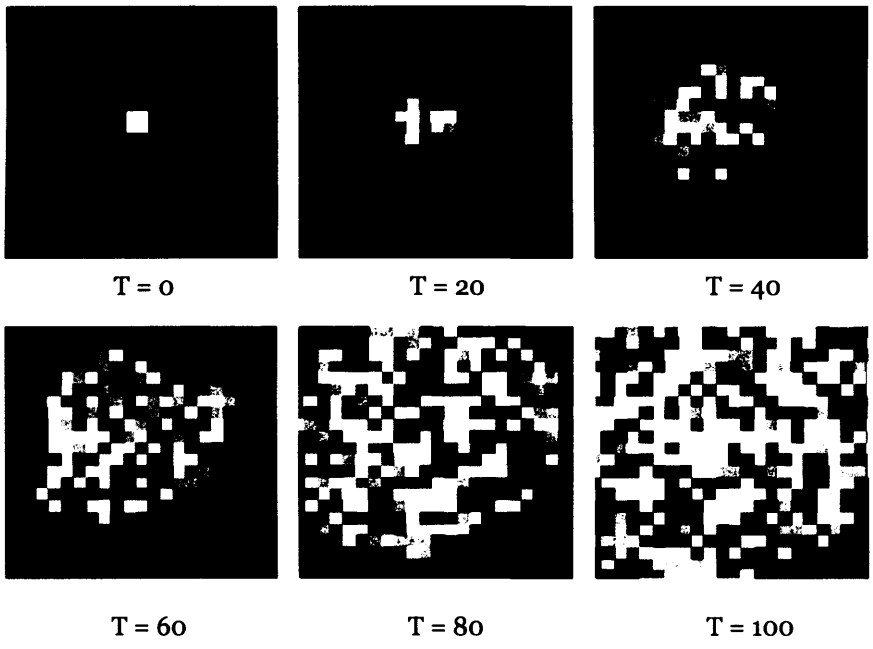


Figure 5.30 Snapshots of B: 1Ns, Dev: 6Ns, and Dec: 8Ns

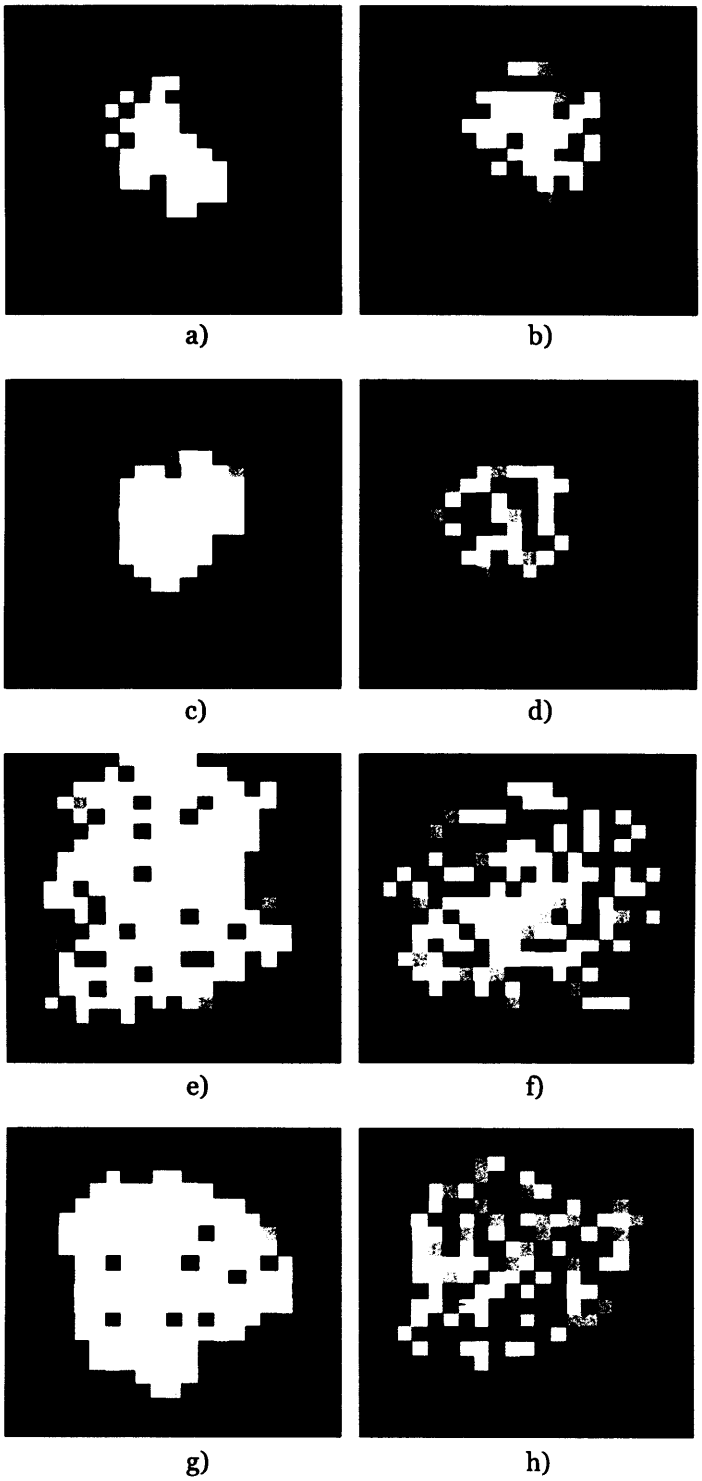


Figure 5.31 The Simulation Results at t = 60 for the Eight Neighbourhood Growth Experiments

In summary, tests of the cellular automata component in the model without any other changes in model parameters does give us a glimpse of how these processes control growth and change. What is particularly new about the way we parameterise the development process in three phases of developing, developed and the transition from developed back down the sequence to developing which coincide with birth, survivorship and death of urban cells, is that we show this controls the growth process of the model. The neighbourhood effect too changes levels of urban growth and this is a particularly innovative addition to CA modelling in that we intrinsically link the growth rates to what happens locally. Moreover we can also tune growth rates to be reflected in different regions of the urban system

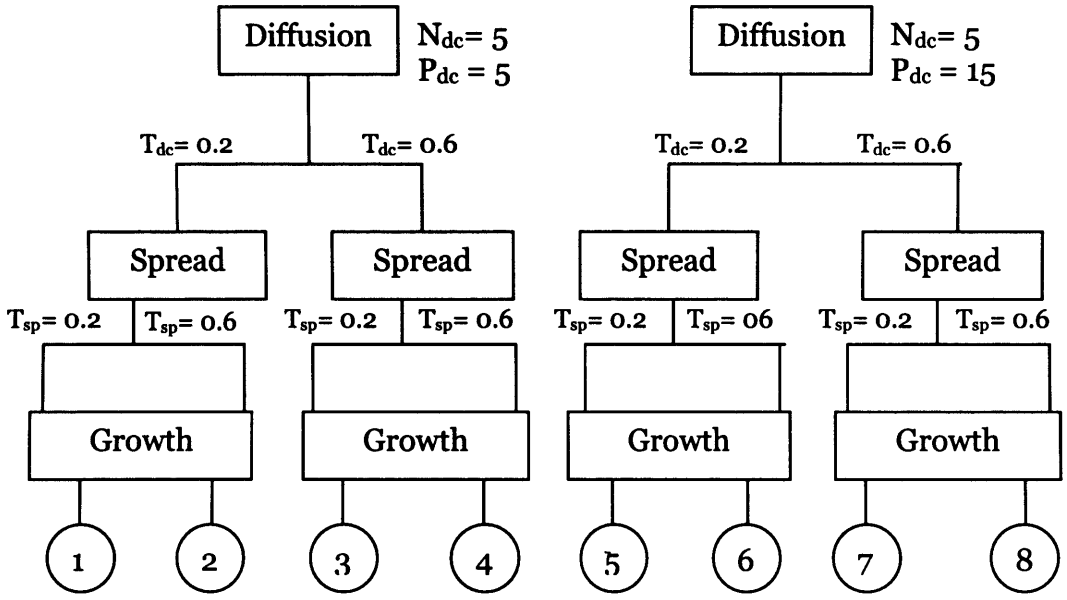
The various results which come from these experiments indicate in general that the morphology of the CA process results in urban forms quite close to the symmetric circle, notwithstanding the fact that the underlying grid is pixel based which is implicitly square. This is one reason why the square grid rather than any other tessellation in the plane is preferable because symmetric circular morphologies indicate an evenness for expansion in all directions with no one direction being preferred. If there are no constraints on growth, the urban shapes produced are close to circular but in fact such growth is never perfect: constrained effects in the space as well as the random seeding and random choice mechanisms governing cellular development create a random world in which historical accident determines the path dependence of the system. In the next section, this more complicated 'random world' will be integrated with other processes and constraints such as diffusion, additional map layer constraints and so on. As we will see, the models become less regular but nevertheless more realistic.

Cellular Automata Simulation with the Diffusion Process

Cells in the urban space are generated using three processes which include diffusion, spread, and growth as described in last two chapters. All growth in the cellular space is then controlled by sets of thresholds which are

defined for each process. In the experiments in this section, a small number of urban seeds were defined as the initial state, all constrained map layers were deactivated, and various parameters controlling diffusion were defined to generate different scenarios so that the urban growth processes generated by the model could be examined.

Earlier experiments represented results from running the model with the diffusion process discarded. In this case, the set of parameters for "growth" from the previous experiments was preset as represented by the values shown in Block 5.9 below. For exploring the effect of diffusion, a parameter based on the number of possible diffusing cells, was fixed as shown in Figure 5.32. This tree diagram represents 8 possible experiments with various parameters defined for each process. These will be discussed later in this section.



N_{dc} = Number of possible diffused cells generated in a time period
P_{dc} = Possible periods of diffusion
T_{dc}= Threshold level for generating or not generating diffused cells
T_{sp}= Threshold level for transforming diffused to urban seed cells

Figure 5.32 Diagram Representing Different Diffusion Parameters for Experiments

Firstly, the period of diffusion was examined defining two different time period values: 5 and 15. Each diffused seed cell that was placed in the space is associated with a random number which is then compared to the threshold which initiates the diffusion. As seen in Figure 5.32, a threshold for the diffusion process is defined as a probability, for example, as say, 0.2. This number covaries with the level of difficulty for generating a diffused cell in the diffusion-space. If the random number generated in this way when the model is run, is more than the threshold, the program will allow a new cell to be placed in the space. If the spread threshold is set at 0.2 for instance, this means there is about an 80 percent chance for diffused cells to transform to urban seed cells in the space. We now deal with eight possible combinations of these parameter values.

From dead cells $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 5; \text{random} \\ 1- : u_t = 1; \omega_2 < 5; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 5; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 4; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 5.9 Functions for B: 2Ns, Dev: 5Ns, and Dec: 4Ns

1. A Short Diffusing Period, a High Level of Diffusion and a High Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 5 possible iteration periods, an 80 percent chance of the cell diffusing due to the random threshold, and thus an 80 percent chance of the cell becoming developed.

Figure 5.33 represents the first sequential images in the diffusion process. A small number of seed cells are placed at the centre of the space as initial conditions. As seen in earlier experiments without diffusion, these cells grow into a pattern of developed cells which surround the initial seed cells. But with diffusion during the simulation, groups of cells over the space are generated randomly, with every cell in the space having the same probability as being part of the diffusion. The results primarily show rather different patterns compared to the CA simulation without diffusion. Cells do not agglomerate around the initial seeds, but spread all over the space.

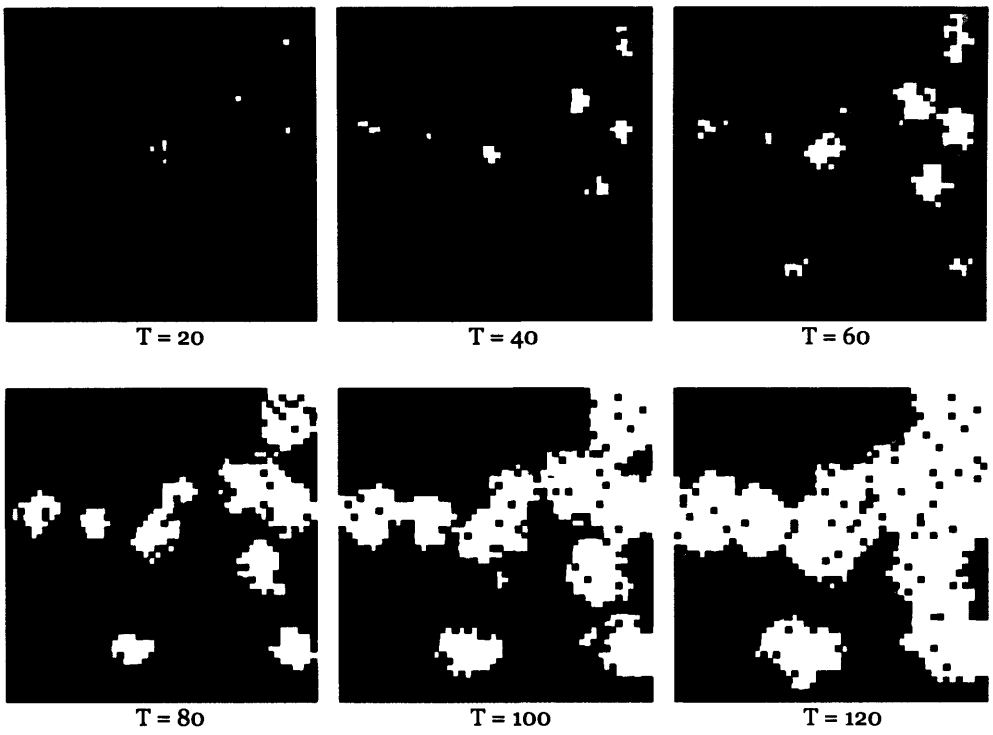


Figure 5.33 Snapshots of CA with Diffusion Scenario 1

2. A Short Diffusing Period, a High Level of Diffusion, and a Lower Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 5 possible iteration periods, an 80 percent chance of the cell diffusing due to the random threshold, and thus a 40 percent chance of the cell becoming developed.

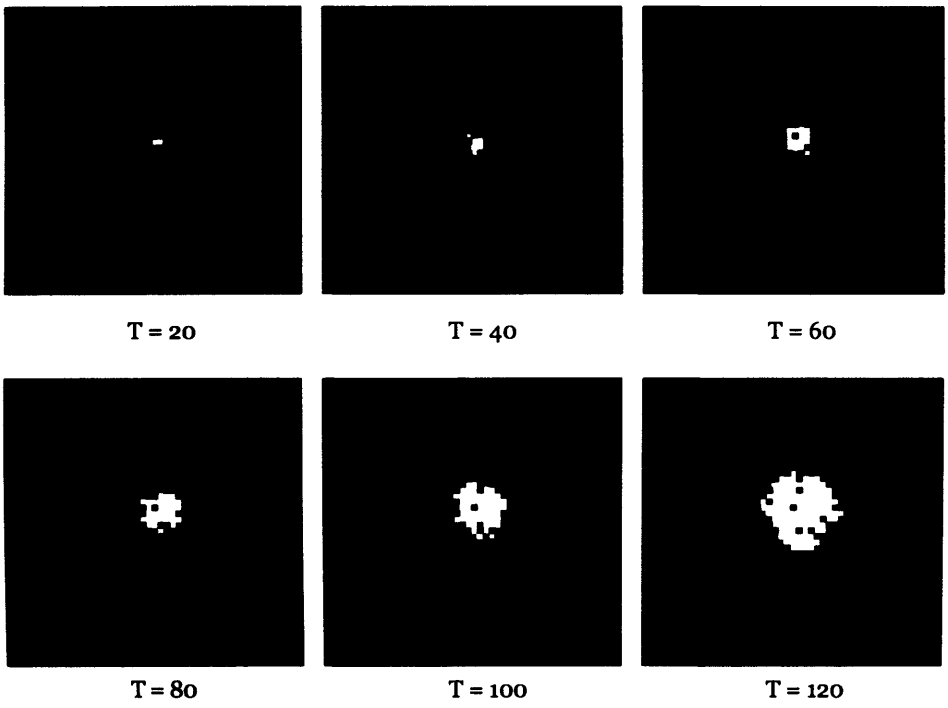


Figure 5.34 Snapshots of CA with Diffusion Scenario 2

In this case, Figure 5.34 illustrates the sequential images associated with this scenario. Again there is a high possibility that cells can diffuse from any location in the space. We think that in the background of the CA simulation, the diffusion module is generating many potential cells that can mutate to new urban cells at each time state. However, by defining a very low chance for diffused cells to become urban, this can cause the simulation to give quite different results from the other scenarios. The images show that there is little influence from the diffusion process; white cells grow in similar fashion to that of the CA without diffusion which we explored in the last experiments.

3. A Short Diffusing Period, a Lower Level of Diffusion, and a Higher Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 5 possible iteration periods, a 40 percent chance of the cell diffusing due to the random threshold, and thus an 80 percent chance of the cell becoming developed.

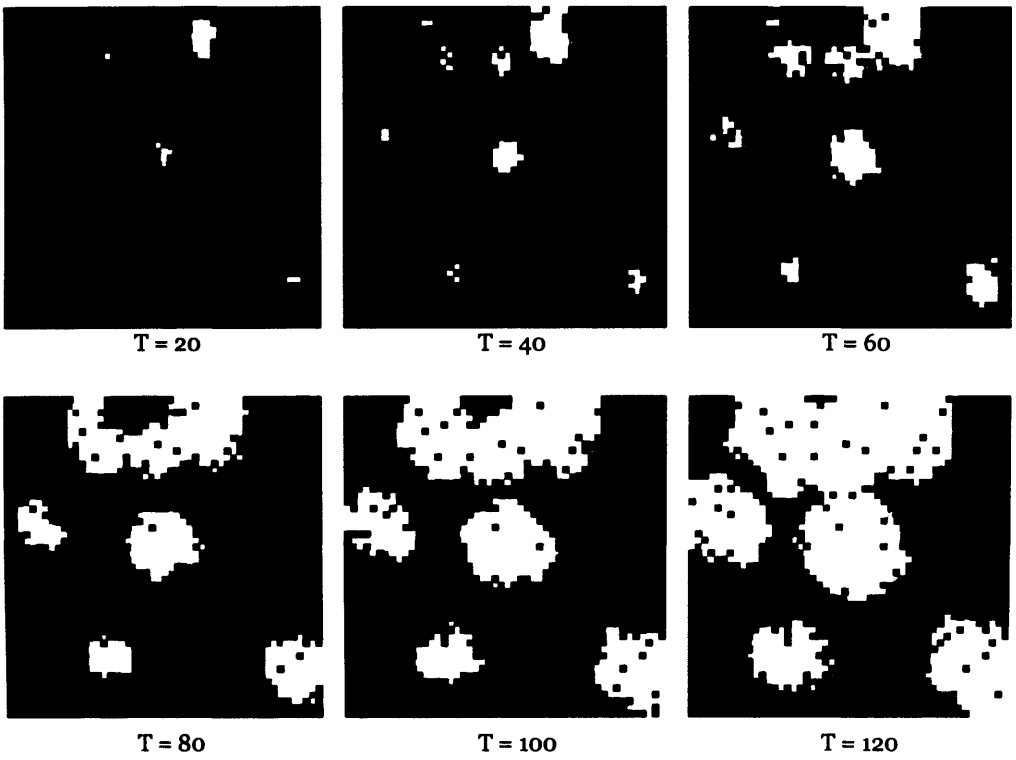


Figure 5.35 Snapshots of CA with Diffusion Scenario 3

In an opposite way from scenario 2, there is less possibility that cells can diffuse but there are more diffused cells that can change into urban cells. Hence, the CA space easily allows such cells to reach the threshold associated with the diffusion. Snapshots from Figure 5.35 represent similar images to those shown in scenario 1. However there are some differences. Due to the small chance that cells diffuse but the greater chance for spreading, fewer cells are therefore placed in the CA space. Those cells are developed under other CA transition rules as we discussed in the last section. In case of the scenario 1, both diffusion and spread probabilities are high, and hence some cell groups are created based on the

diffusion and spread. These groups thus tend to be larger than those that naturally grow under the simple CA rule.

4. A Short Diffusing Period, a Lower Level of Diffusion, and a Lower Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 5 possible iteration periods, a 40 percent chance of the cell diffusing due to the random threshold, and thus a 40 percent chance of the cell becoming developed.

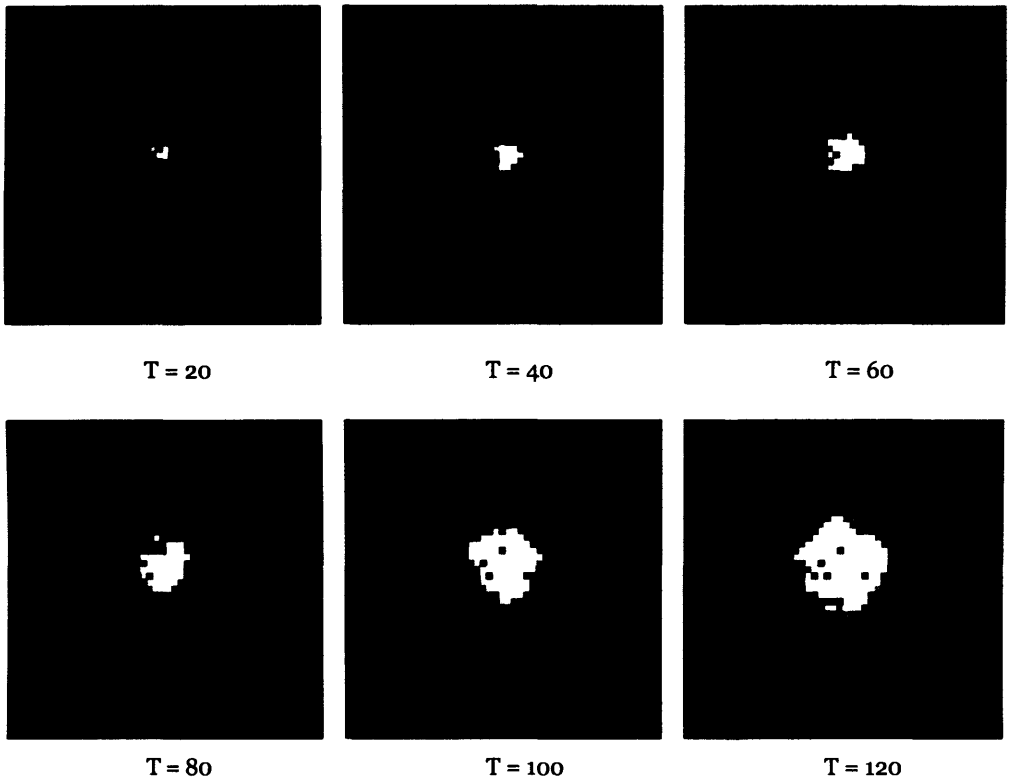


Figure 5.36 Snapshots of CA with Diffusion Scenario 4

Compared to scenarios 2 and 3 (see Figure 5.36), this experiment has less chance for all cells both to diffuse and to become urban. The results are similar to the images shown in Figure 5.34 in scenario 2, with the implication that the diffusion has no influence on this growth.

5. A Longer Diffusing Period, a Higher Level of Diffusion, and a Higher Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 15 possible iteration periods, an 80 percent chance of the cell diffusing due to the random threshold, and thus an 80 percent chance of the cell becoming developed.

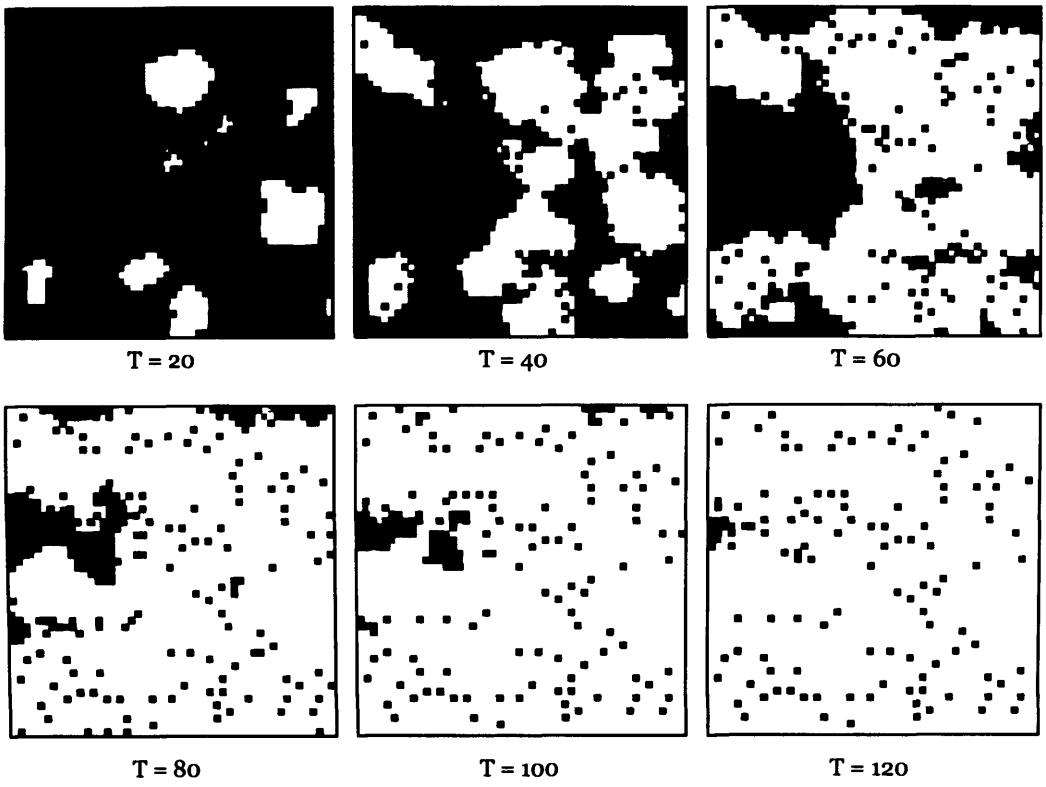


Figure 5.37 Snapshots of CA with Diffusion Scenario 5

This can be described in the same way as the first four scenarios. The longer diffusion period generates more cells in the space. As we see in Figure 5.37, the image at T = 20 clearly shows that some groups are growing much faster than the initial seed cells in the centre of the image. These appear to become bigger through the positive feedback of other rules in the later time states.

6. A Longer Diffusing Period, a Higher Level of Diffusion, and a Lesser Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 15 possible iteration periods, an 80 percent chance of the cell diffusing due to the random threshold, and thus a 40 percent chance of the cell becoming developed

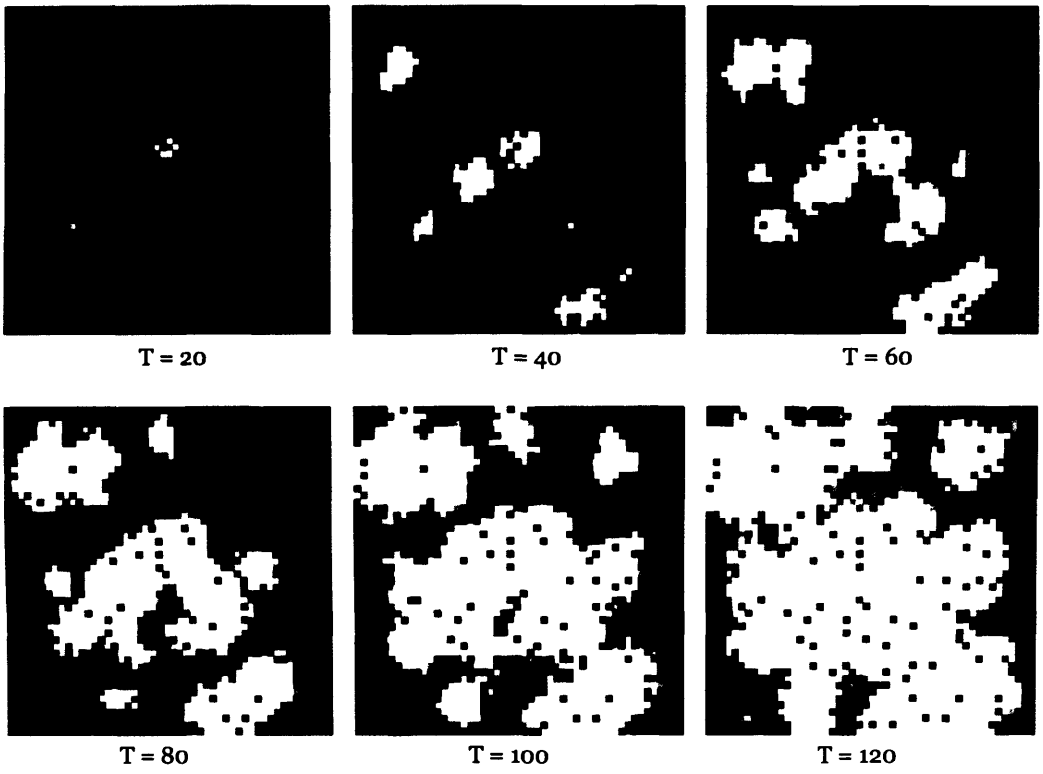


Figure 5.38 Snapshots of CA with Diffusion Scenario 6

Except for the developing period, all the rules in this trial were set as the same as those in the second scenario. However the results shown in Figure 5.38 are different. The longer diffusion period causes the simulation to generate more potential cells in the CA space, and those cells begin to develop according to other CA rules. Figure 5.34 showed development without the diffusion process for there were no cells to diffuse which reached the threshold for urban development, and thus all new cells were created from the initial seed cells only. It can be argued that with an

increasingly longer period for diffusion, it is difficult to generate growth through diffusion, as is clearly shown in Figure 5.38.

7. A Longer Diffusing Period, a Lower Level of Diffusion, and a Higher Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 15 possible iteration periods, a 40 percent chance of the cell diffusing due to the random threshold, and thus an 80 percent chance of the cell becoming developed.

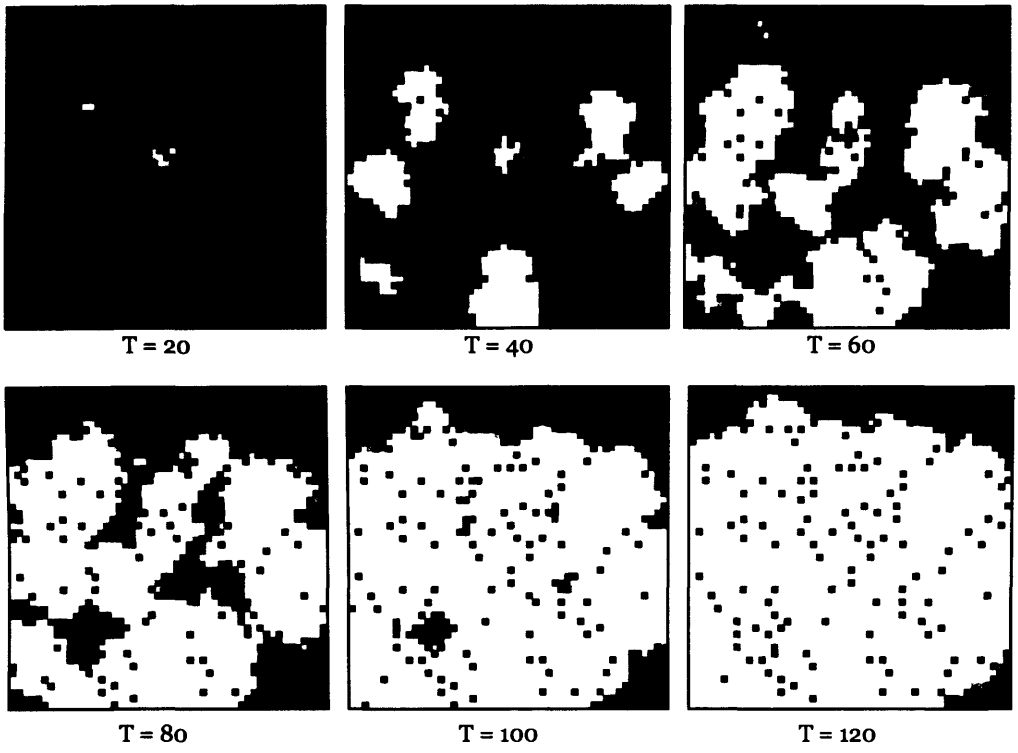


Figure 5.39 Snapshots of CA with Diffusion Scenario 7

In the case of this set of rules, cells are more difficult to diffuse, but there is a higher probability that they will transform into urban cells. Unlike Figure 5.38 in the previous scenario, only six observable groups of urban cells emerge in the space, from which they then begin to expand as depicted in Figure 5.39. Due to the lesser possibility of their diffusion, but the higher chance of their becoming urban, only a few groups of urban cells were generated which were similar to scenario 3. These cell groups do not

become too dispersed and as such, they are more concentrated around their centres than those generated in scenario 6.

8. A Longer Diffusing Period, a Higher Level of Diffusion, and a Lower Level of Spread

In this experiment, the chances for the diffusion of a cell requires 5 possible seed cells, 15 possible iteration periods, a 40 percent chance of the cell diffusing due to the random threshold, and thus a 40 percent chance of the cell becoming developed.

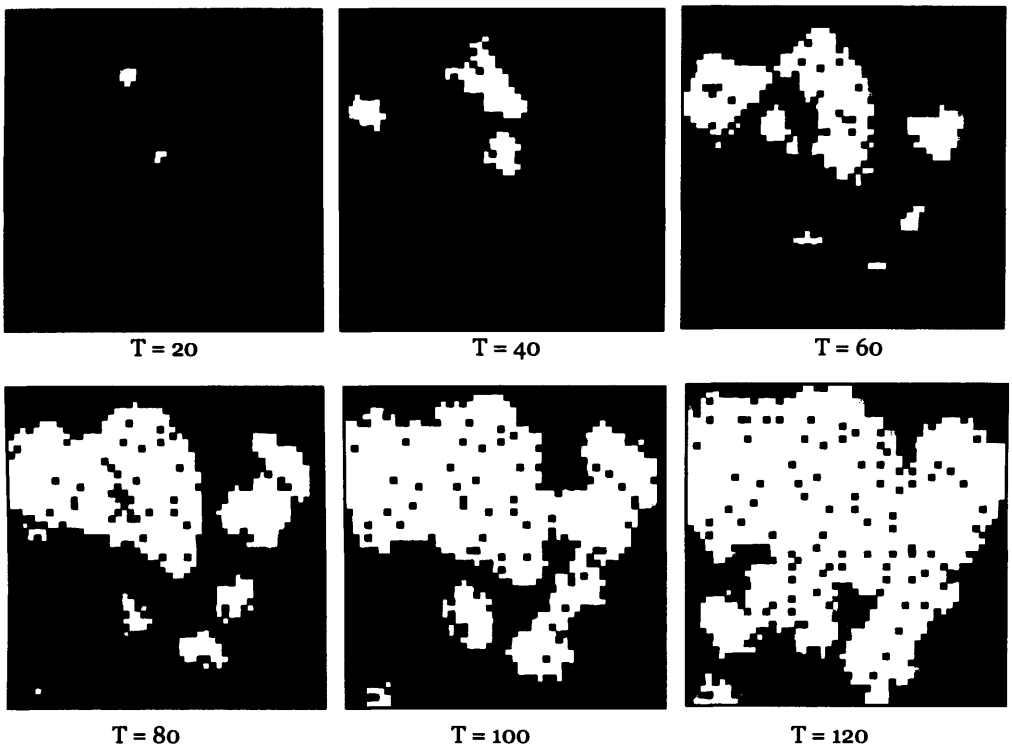


Figure 5.40 Snapshots of CA with Diffusion Scenario 8

This experiment can be compared to the fourth scenario where there is no influence from the diffusion process. In this case, even though the rules can affect cells that are more difficult to diffuse and transform into developing cells, there is the possibility that diffusion can be increased because of the length of the diffusing period as shown both in Figures 5.36 and 5.40 above.

Constrained Cellular Automata Simulation and Dynamic Maps

The CA simulation is normally based on local transition rules that apply uniformly to the entire cellular space. This assumes, of course, that there are no differences between the cells in the space as we have assumed in all the previous experiments. The only influences on cells have been changes in the rules but these have been done uniformly and no one cell is privileged or disadvantaged over any other. Nevertheless in reality, there are many factors that influence the urban growth processes, which can be reflected as conditions and constraints, for instance, topographic features, transportation, population and so on. By implementing the model to integrate these factors, the simulation becomes much more complicated and thus our experiments in this section will be more involved than those in the previous two sets of experiments.

In this section, all the algorithms and rules from previous experiments have been adapted to a standard set of data which are shown as dynamic map layers in the program. These map layers are used to calculate and return cell-by-cell constrained probabilities that are used to control growth in the CA simulation. Conversely, changes of cell states in the CA space will cause transformation of the features comprising the map layers and these are the results of the positive feedbacks from changes to location which is the essence of the nonlinearities and complexities simulated within the model.

As discussed in Chapters 3 and 4, at every definable time state, a constrained map layer can be created based on the spatial weighted overlay method. The map layers in the model include the standard set of layers that we have noted several times previously: the river, transportation, population, land use/cover, population density, topographic feature (slope), land demand, centrifugal and centripetal forces, all expressed as maps. According to the weighted-overlay method, we need to assign a set of weight scores to all map layers which as they vary, will return different constrained conditions. This section will test the simulation model using a fabricated dataset with arbitrary and

predetermined sets of weighted score parameters. All the dynamic maps which result from the simulation will be represented as snapshots at each time state, except the centrifugal and centripetal force layers that will be discussed in a later set of experiments.

Table 5.2 Scenarios for the Constrained CA and Dynamic Map Experiments

Scenario	Map layers: Weight(sum = 10)
1. Growth depending on accessibility, topography and land use, and land demand respectively	o Centripetal force: 0 o Centrifugal force: 0 o Accessibility: 5 o Topography: 2 o Land use: 2 o Land demand: 1
2. Growth depending on topography, land use, land demand and accessibility respectively	o Centripetal force: 0 o Centrifugal force: 0 o Accessibility: 1 o Topography: 4 o Land use: 3 o Land demand: 2
3. Growth depending on land demand, land use and topography, and accessibility respectively	o Centripetal force: 0 o Centrifugal force: 0 o Accessibility: 1 o Topography: 2 o Land use: 2 o Land demand: 5
4. Growth depending on land use and accessibility, topography and land demand respectively	o Centripetal force: 0 o Centrifugal force: 0 o Accessibility: 4 o Topography: 1 o Land use: 4 o Land demand: 1

Table 5.2 provides four experiments defining four possible sets of weights for each map layer. The centrifugal and centripetal forces are not included in the calculations of constrained conditions at this stage for we consider them to be sufficiently complex to warrant a separate set of experiments which we will deal with later. As seen in Table 5.2, in the first scenario for instance, the CA simulation is constrained mostly by the accessibility layer which is given a weighted score of 5, with a low score set as 1 for land demand (Note that the summation of all weights is always 10, so these are relative weights and enable direct comparisons to be made between the factors). Logically, we might expect outcomes from this scenario to show cells that will expand according to the existing transportation features in

the space more than any of the other factors. We will, as Table 5.2 implies, set the other 3 scenarios in different fashion and these will, we hope, be reflected in the sequential results to be presented below.

1. Growth Ordered from Accessibility, Topography, Land Use, to Land Demand Respectively

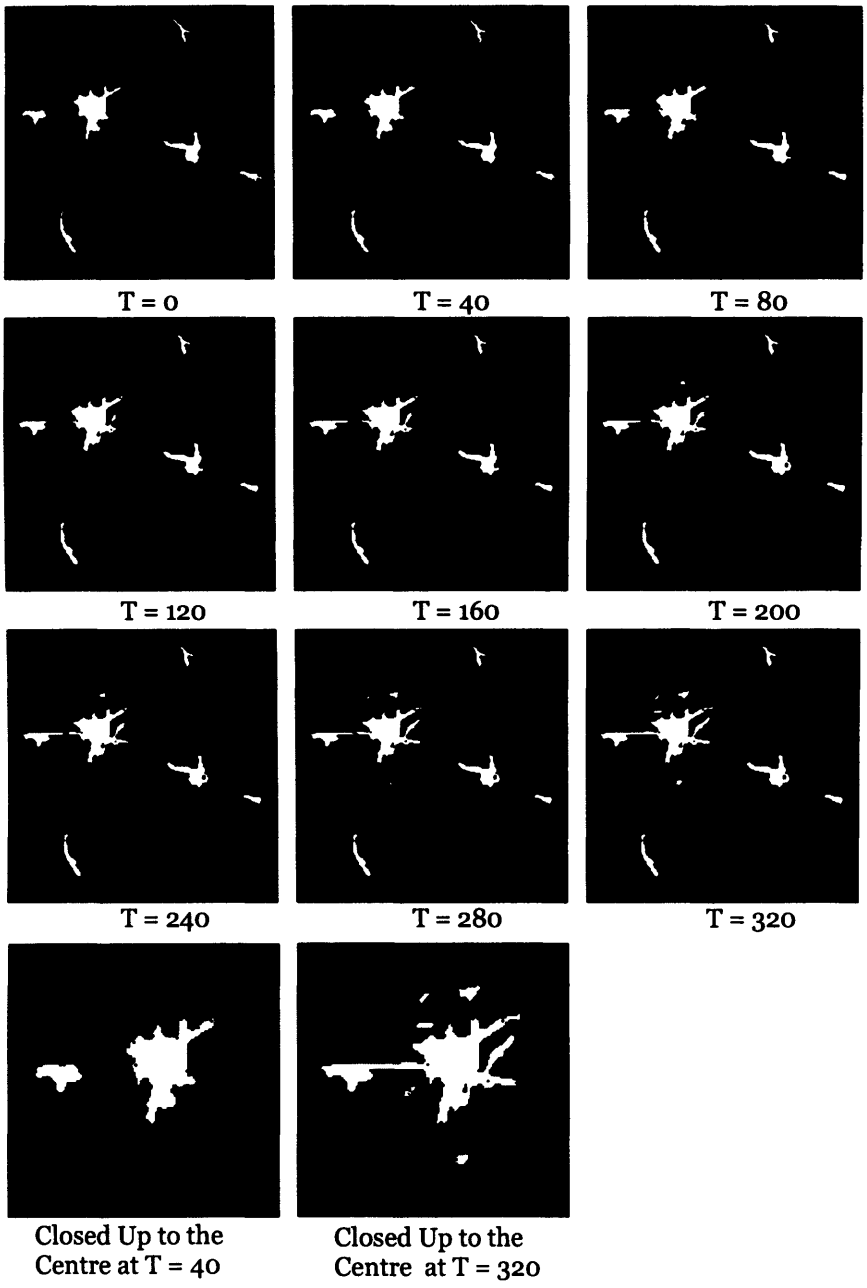


Figure 5.41 Snapshots of Urban Areas for Scenario 1

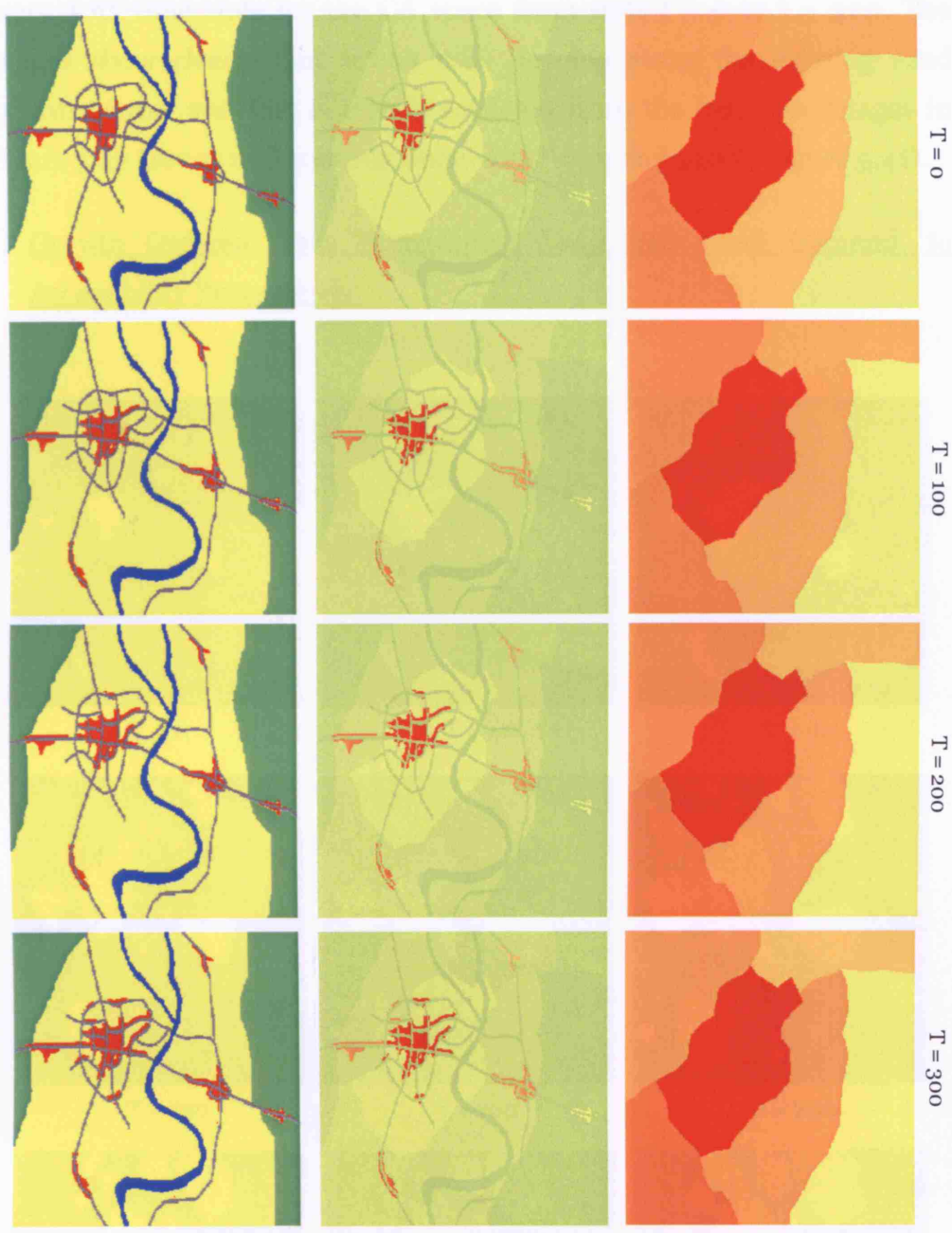


Figure 5.42 Snapshots of Land Use, Population Density and Land Demand for Scenario 1

The highest weighted score (5) was set for the road map layer and therefore any cells that locate close to the existing road cells have a greater probability of transforming to urban cells. In the case of urban land demand, the weighted score was set as 1 meaning that this has much less influence on urban growth. Figure 5.41 provides snapshots of change at

every 40th time state for the CA space from state t = 0 to t = 320. The images show clearly that urban cells develop along the existing road infrastructures and this is clearly seen too from the land use images in Figure 5.42 (see as well zoom-in images at T =40 and 320 in Figure 5.41).

2. Growth Ordered from Topography, Land Use, Land Demand, to Accessibility Respectively

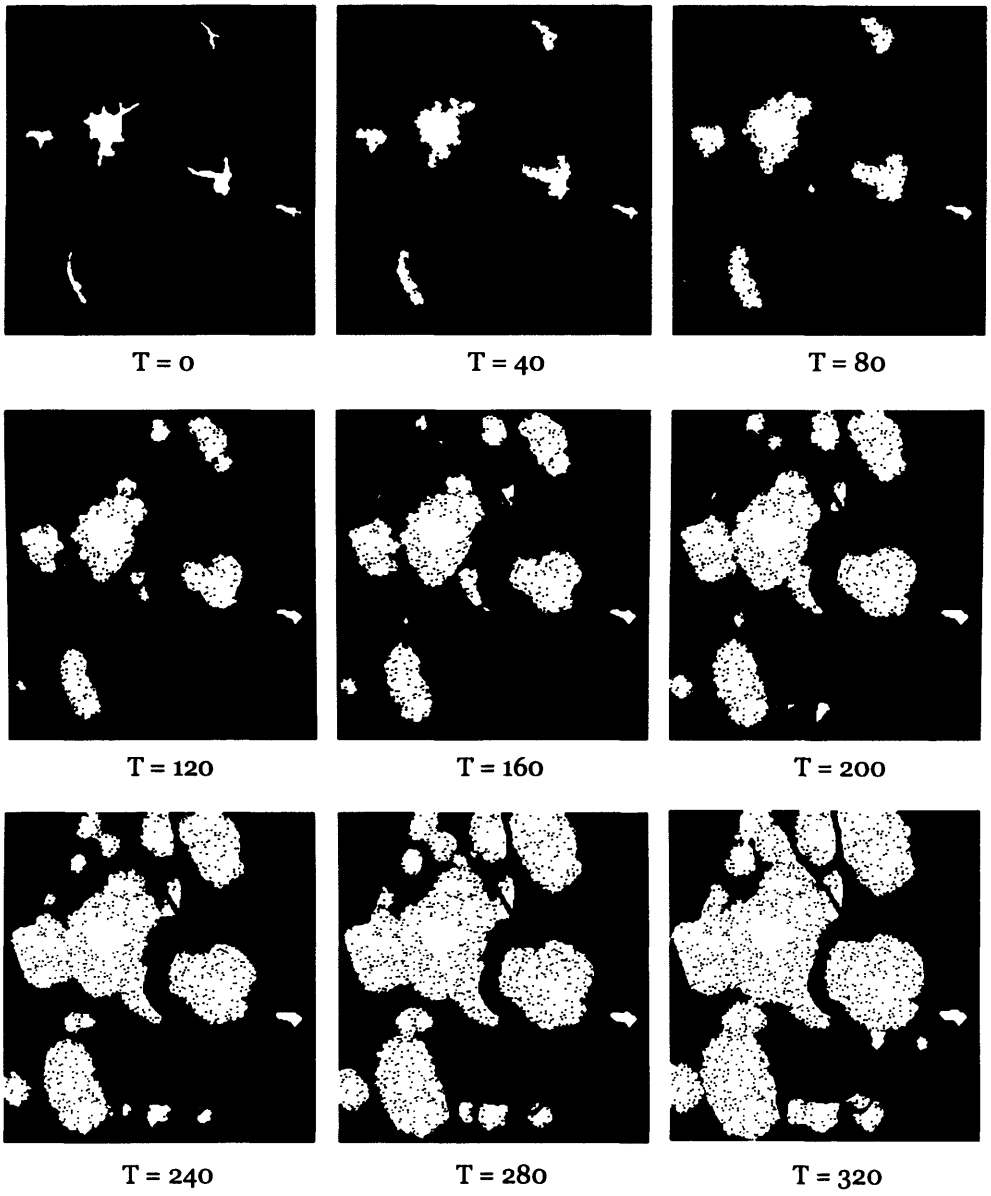


Figure 5.43 Snapshots of Urban Areas for Scenario 2

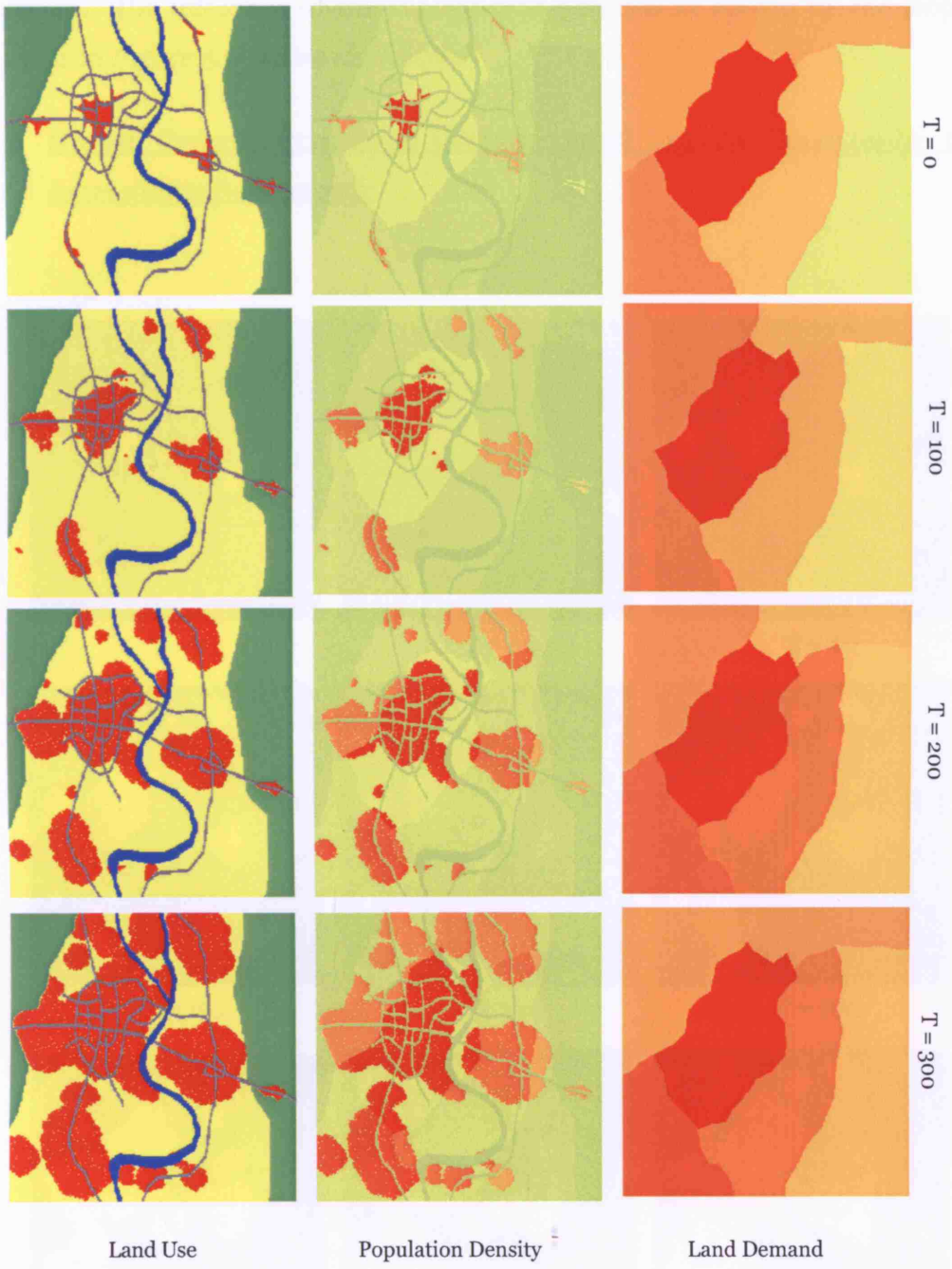


Figure 5.44 Snapshots of Land Use, Population Density and Land Demand for Scenario 2

This scenario examines how the urban area will develop with a much smaller score for the accessibility and much more weight for topographic features and land use. The results are presented in Figures 5.43 and 5.44 above which make it clear that the smallest weighted scores defined for

accessibility take urban growth away from the existing transportation system. The influence of the topographic features as shown by the slope maps in Figure 5.6 is also significant.

3. Growth Ordered from Land Demand, Land Use and Topography, to Accessibility Respectively

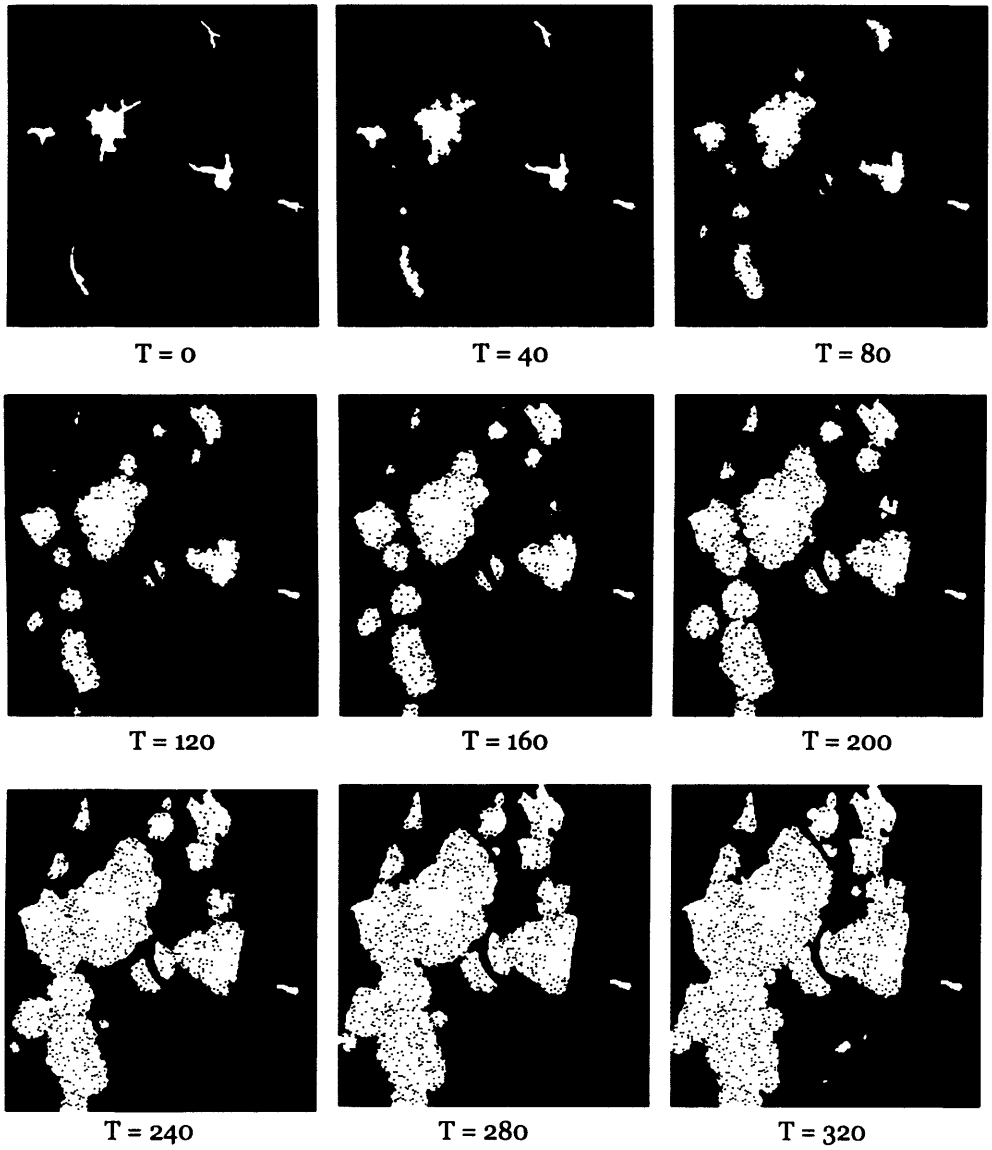


Figure 5.45 Snapshots of Urban Areas for Scenario 3

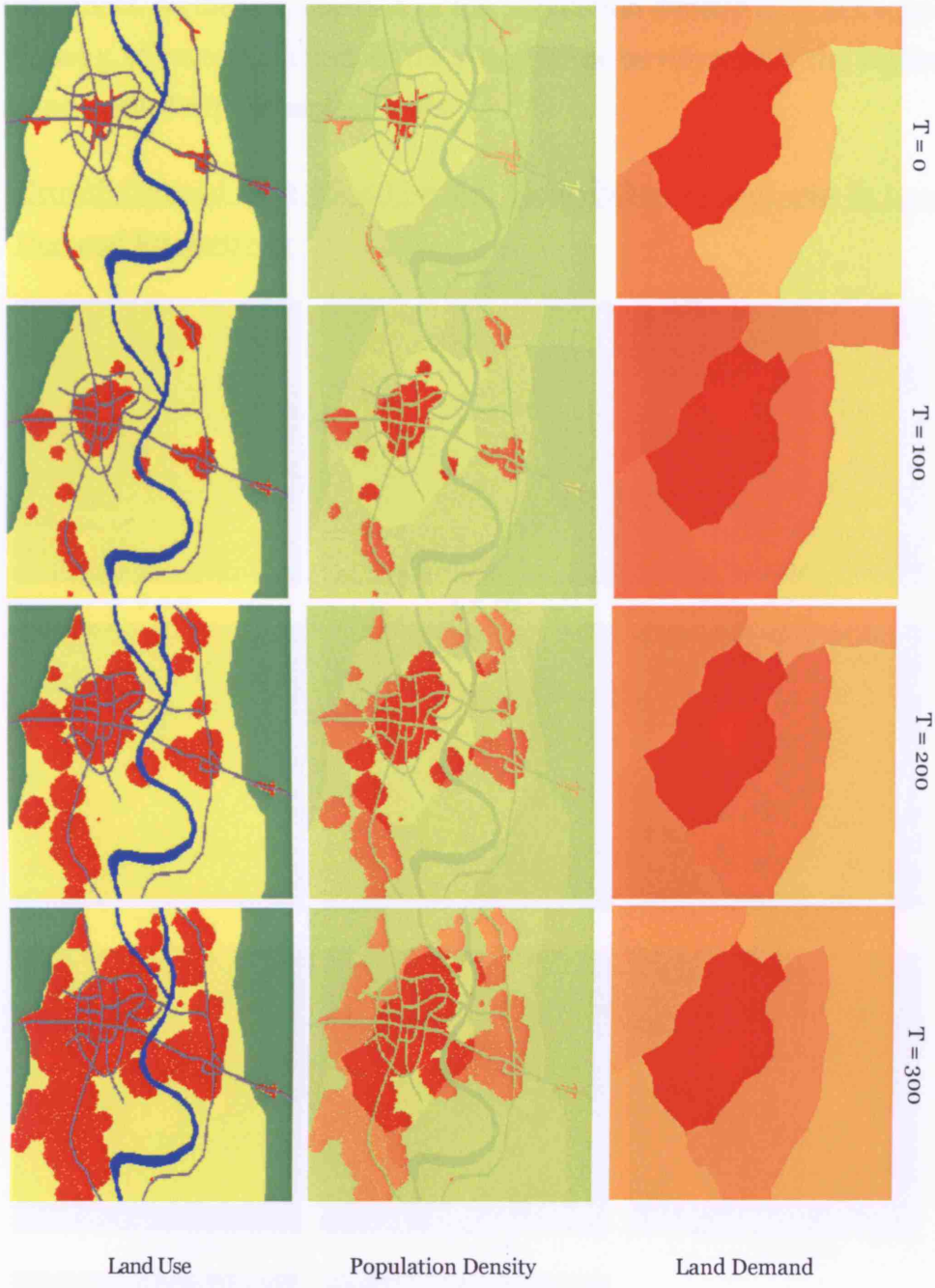


Figure 5.46 Snapshots of Land Use, Population Density and Land Demand for Scenario 3

During this growth process, the sequence of images in Figures 5.45 and 5.46 clearly illustrate the differential impact of different levels of land demand on each of the administrative territories. As represented by the land demand maps in Figure 5.46, areas associated with higher demand generate much higher probabilities of transformation to urban

development. As the visualisation in the population density map at t = 300 in Figure 5.46 reveals, almost all the space in the territory with the highest demand is filled with urbanised cells.

4. Growth Ordered from Land Use and Accessibility, Topography to Land Demand Respectively

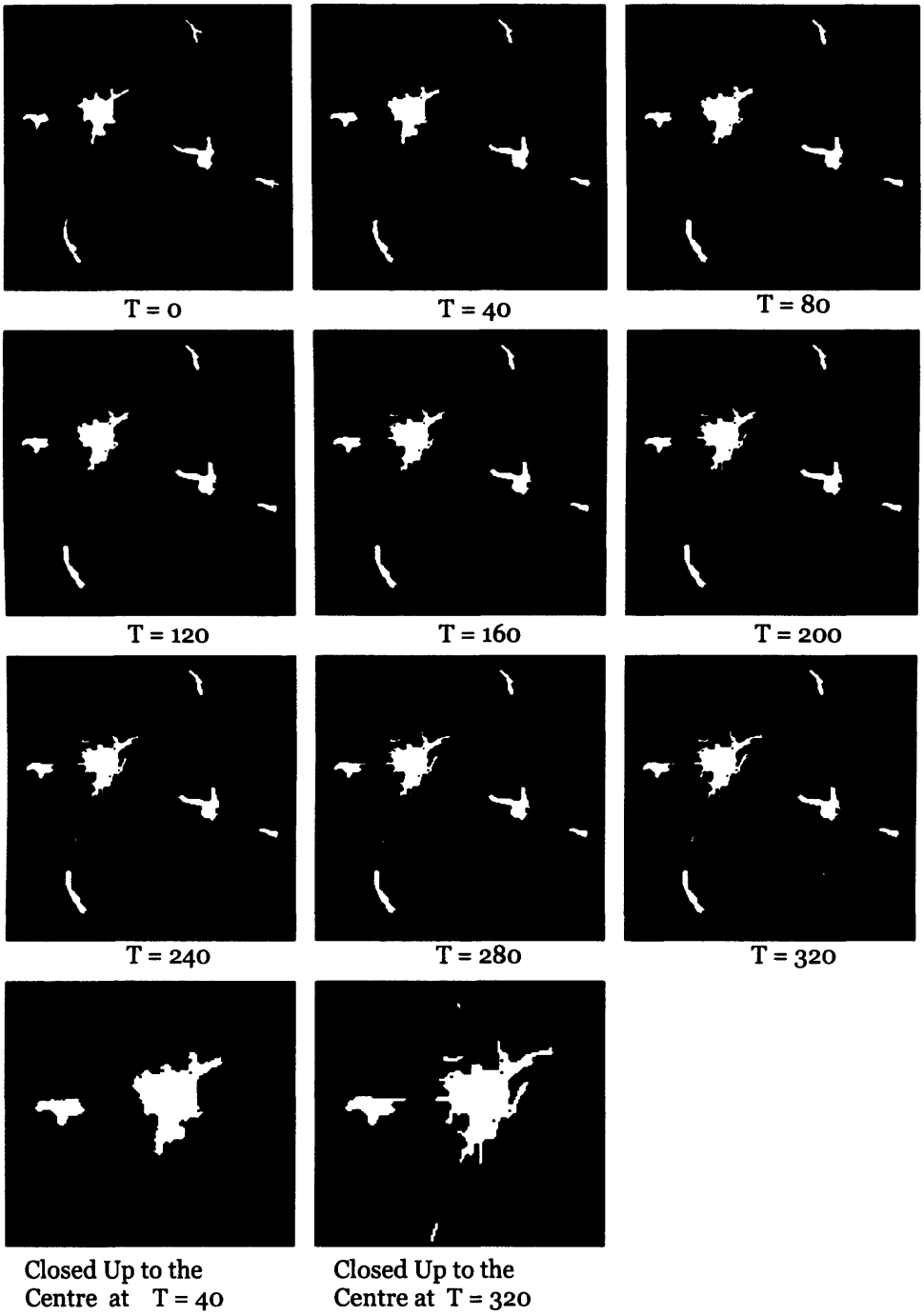


Figure 5.47 Snapshots of Urban Areas for Scenario 4

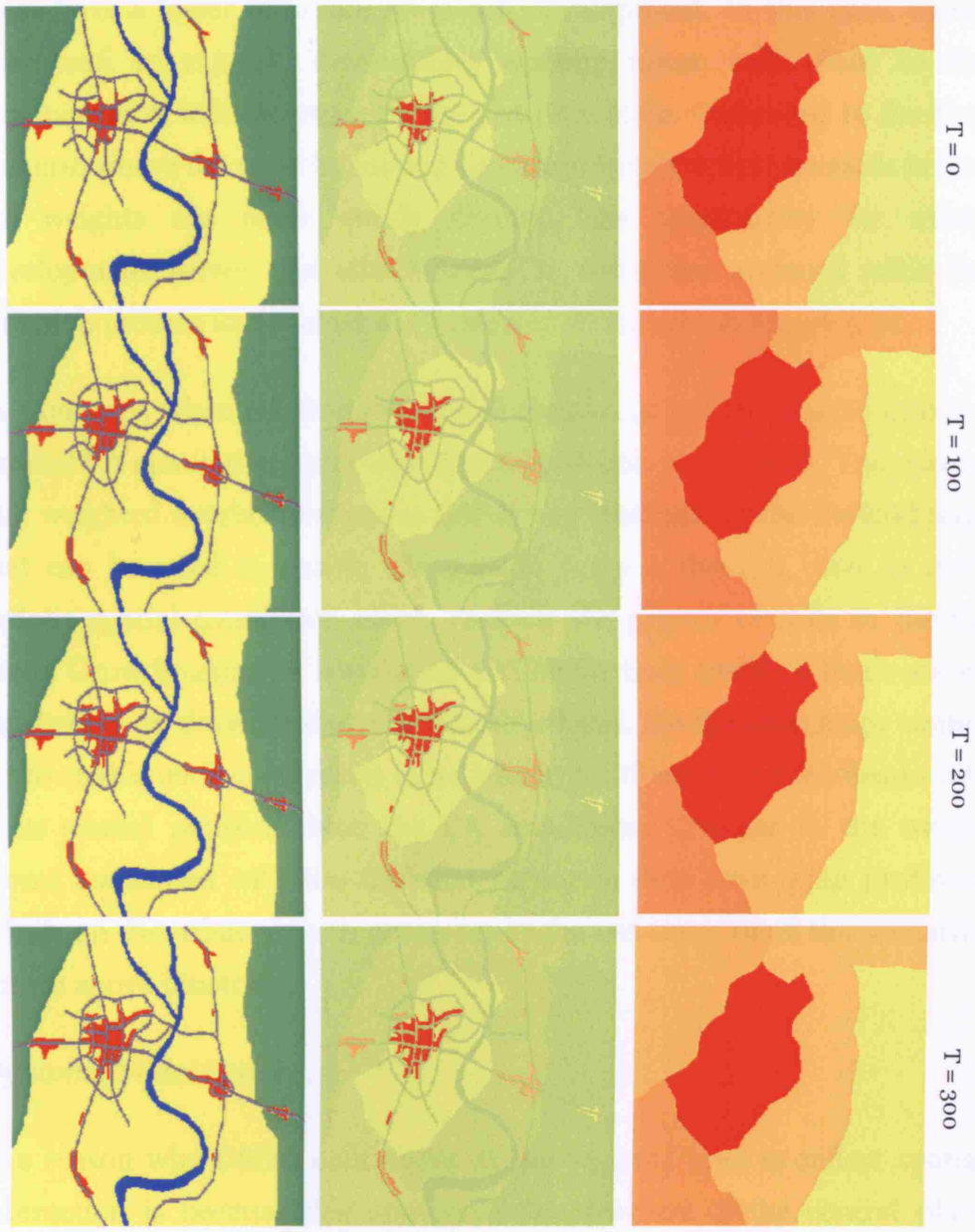


Figure 5.48 Snapshots of Land Use, Population Density and Land Demand for Scenario 4

This scenario defines an equivalent weighted score for both accessibility and land use and the simulations are illustrated in Figures 5.47 and 5.48. Cells which are closest to the transportation features and have a majority of neighbouring cells of urban land use, have a higher probability of becoming urban (see close-up images in Figure 5.47). On the other hand, as the importance of land demand and topographic features is reduced,

these have a lesser influence on urban development. In this case, urban cells tend to expand, surrounding existing urban cells close to the transportation infrastructure by the last time state. Compared to the first scenario where the most important condition from the first scenario is that the weights are more evenly divided, the opportunity for urban development across the whole space is somewhat reduced with the diffusion process less evident in Figure 5.47 compared to Figure 5.41.

In summary, when constrained spatial conditions are introduced into the general CA simulation, quite significant differences can result. This means that weighted overlay analysis in GIS is one vital part of the methodology that can be used to enable the growth process through time to meet various spatial conditions which restrain the growth of cells in the CA space. Growth using CA simulation will be far from reality if there are no constraints on development. On the other hand, the dynamic maps cannot act to enable such constraints to be effective if there are no feedbacks into these spatial patterns from the CA simulation. Changes in the weight scores associated with the different scenarios thus have quite profound effects on the urban growth processes as the outcomes of all the scenarios shown above illustrate.

Dynamic Nodal Spaces

The reason why DSSM uses nodes at the regional level to reflect spatial interaction is because this reinforces the structure of the central place hierarchy which is important to the more aggregate regional structure. As well as representing the urban areas in the CA space as we show in all previous experiments, we need to show the size and shape of each urban cluster which reflects their individual characteristics of 'centrality' and 'hierarchy'. Within the CA space, both change over time and essentially reveal useful socio-economic rather than physical information for users. Moreover, the distribution of nodes and their attributes created over time is also part of the source for constructing the dynamic surface maps that will be discussed in the next session.

Experiments in this part were implemented for testing the nodal generating algorithm discussed in detail in Chapters 3 and 4. To observe how nodes and centres are generated based on urban clusters in the CA space, the small cellular space (50x50 pixels) was used with a handful of urban seeds to kick start the experimental simulations. Without any constraints and disregarding the diffusion process, this first trial represents a pure simulation of how the nodal system emerges. Secondly with the same size of cellular space, all the constraint conditions and the diffusion process were added so that additional experiments could be handled to find out how nodes are created and distributed with respect to these spatial factors. Finally an experiment for the fully constrained CA simulation was performed on the larger cellular space (250x250 pixels) so that we might obtain results that are closer to reality.

The Nodal System with an Unconstrained CA Simulation in the 50x50 Space

Centres of each urban area in the model are calculated using the number of existing cells and their attributes at each time state. An important parameter in performing this algorithm involves defining the search-radius kernel which applies to all the cells in the space, with consequent changes in the various search radius kernels causing different patterns of nodal system. In this trial, two different values of the radius were defined: 5000 and 6000 meters. Because the radius is defined on a known measurement unit, it is necessary to assign a resolution to the testing space; in this case, the dimension of space was 50 x 50 pixels and we assume that the resolution of 1 pixel is equal to 500 meters in its height and width.

1. The 5000 Meters Search Radius

First, an initial state of the simulation was defined with a handful of urban seed cells being defined around the urban centre in the space. At every 10 time states, the node generator algorithm is operated and this will relocate existing nodes, generate new nodes based on the distribution of urban cells

within a 5000 meters search radius, and calculate the size of these new nodes within 1/3 of the 5000 meters search radius to capture the urban population density that defines these nodes. In this case, this value is assumed to be a central area which is proportional to the whole search radius. Figure 5.49 represents how urban growth interacts with node generation with the size of nodes defined from a ranking of the values of the urban population density within the proportional search radius. At t = 90 in Figure 5.49, various sizes of node are generated in the space. For instance, a small node at the right edge of 90th time state in Figure 5.49 was generated from a separated group of cells which were randomly expanded from the CA simulation with this node becoming bigger larger at later time periods.

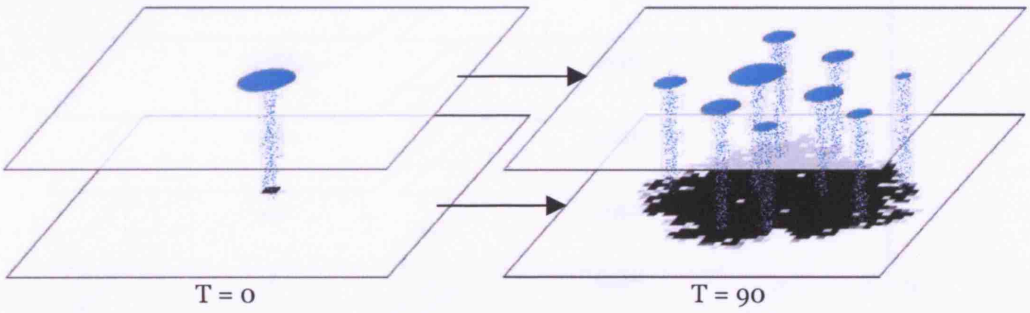


Figure 5.49 Nodes and Urban Areas at the Initial and End States

The time states between t =0 to 90 in Figure 5.49 are shown in Figure 5.50 below. This progression can be described from an initial node which is generated from the seed cells at the zero time state. During the simulation period t=0 to 30, the node continues to exist at the same status but there is a little change in its position. From the CA simulation based on random generation as discussed earlier in this chapter, after 40 time states of the simulation, the urban area does not approach a perfect circle-like shape. It becomes clear that the growth moves to the southeast direction and the node appears some distance from the centre. The sizes of these nodes become gradually bigger as the simulation progresses. However without any constraining factors, there is an obvious development trend with the urban area growing into a compact shape at t = 90 which is clear from Figures 5.49 and 5.50.

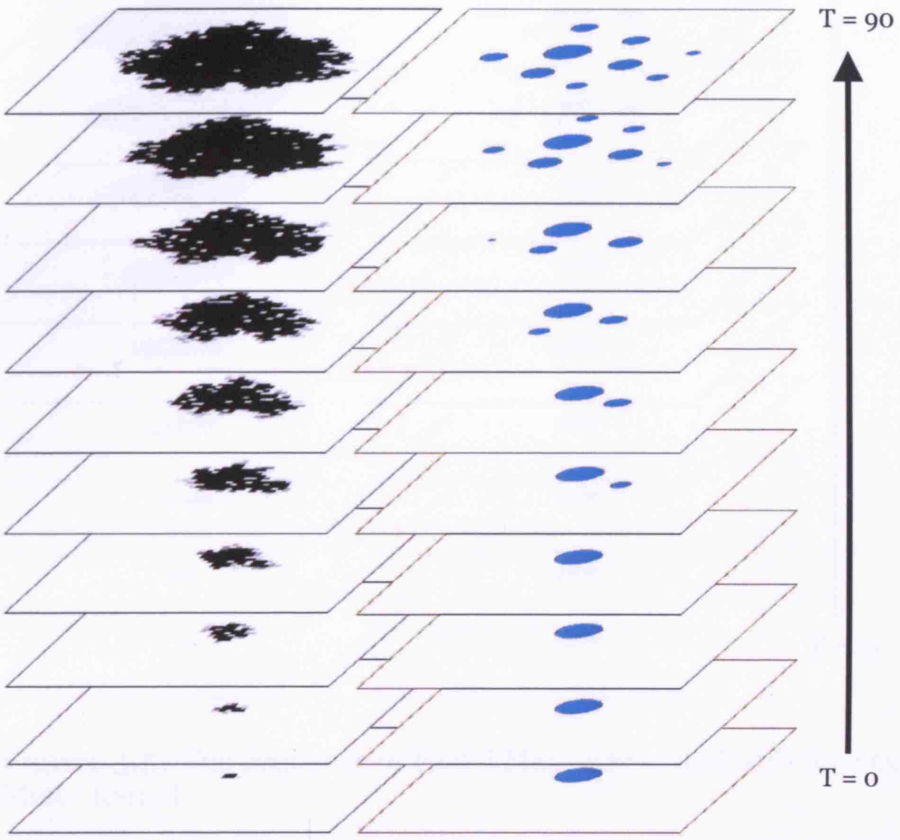


Figure 5.50 The Sequence of Nodal Maps Associated with the 5000-Meter Kernel

The 6000 Meters Search Radius

Increasing the search radius in the simulation leads to two clear changes: the central node spreads out and increases in its size. In the same manner as in the first simulation, Figure 5.51 shows the simulation using the 6000-meter search radius to create nodes. From these images, the initial node exists separately from new nodes somewhat longer than in the case of using the 5000 meter kernel. When the urban area gets big enough, two more nodes are created at some distance from the centre.

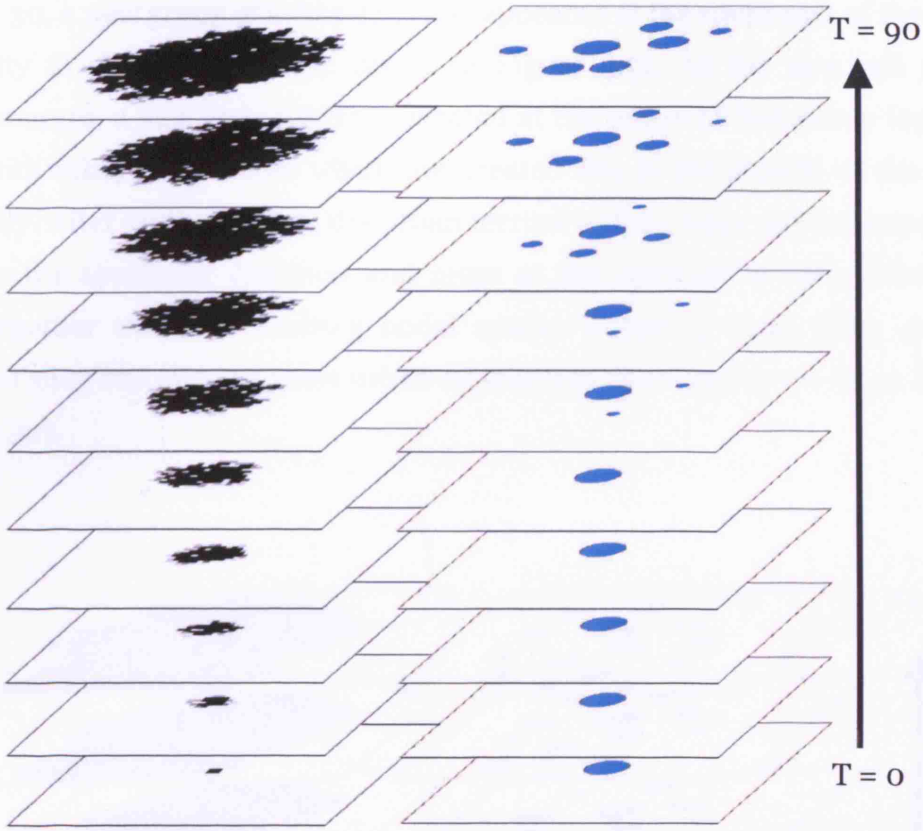


Figure 5.51 The Sequence of Nodal Maps Associated with the 6000-Meter Kernel

The Nodal System with a Constrained CA Simulation in the 50x50 Space

The previous experiments depicted how nodes are created solely based on the CA simulation but in this section, we will include the diffusion process and some constraining conditions leading to more complicated patterns of growth which are clearly reflected in the node generation process. We show two trials as follows which are implemented so that we can observe these changes in the nodal system. The search radius for computing and locating these nodes was set at 5000 meters, with the actual search radius for calculating hierarchies set at approximately 1700 meters (1/3 of 5000).

1. A Single Seed Group with Diffusion Process

This experiment begins by defining a small group of seeds at the centre of the space. Figure 5.52 below represents the simulation results that show

the way the urban areas grow. Compared to the previous experiments at t = 30, a new group of urban cells has appeared at the southwest of the main city as shown in the top image in Figure 5.52. As the new cell group emerges, a new node was then located at the centre of that group together with other three nodes which are created due to the growth of the main city. After 70 time states, the urban territory of the main city has expanded to the southwest direction and areas at the outskirts of both cities fuse together with the resulting nodal system reflecting these cities as now forming one and the same urban system (see the image at t = 90 in Figure 5.52).

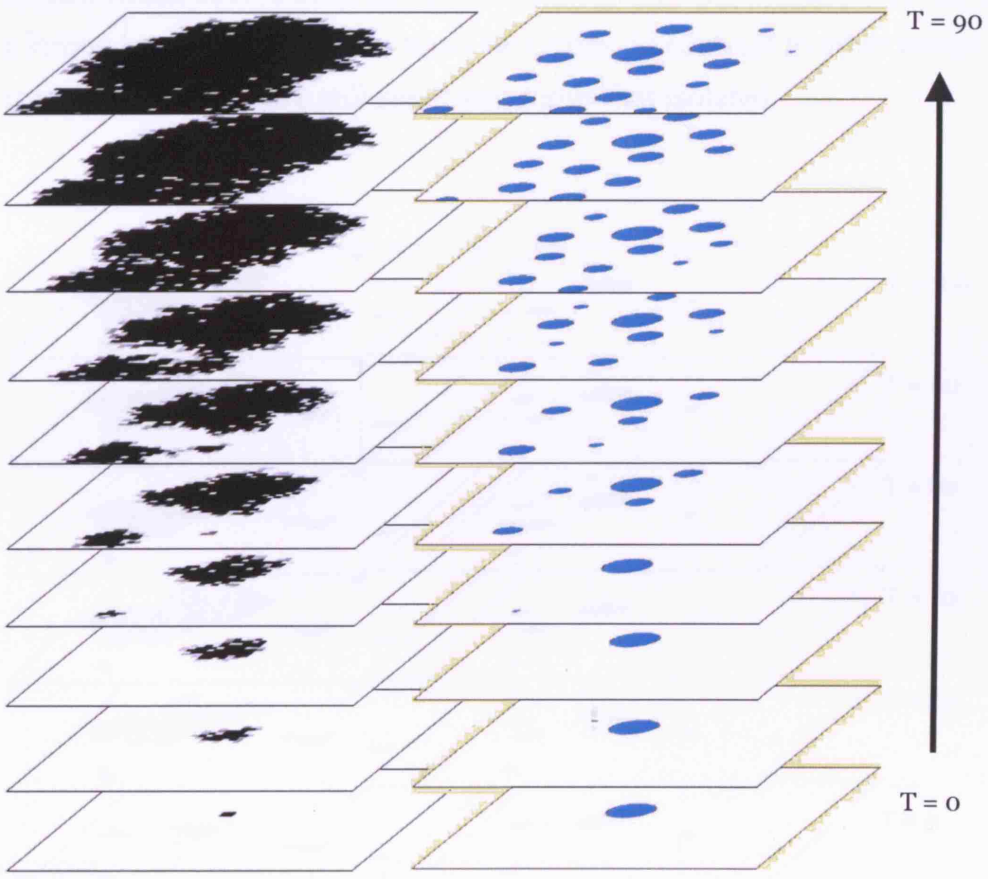


Figure 5.52 The Sequence of Node Maps Generated from the CA-Diffusion

2. Multiple Seed Groups with Constraining Factors

The images shown in Figure 5.53 below represent changes in the patterns of the urbanization associated with the cities and towns which result from multiple seeds under constraints. In the constrained CA simulation above, a set of weight scores was predefined for each map layer, and we used these here for this simulation. As seen in Figure 5.53, it is clear that there are some differences in growth over the space. From the centre city, the urban area tends to expand through the west side of the city as seen in the image at t = 20 in Figure 5.53. After the 40th time state, the main city covers all the area in between the original city and the town on the west side. At this point in time, the distribution of nodes reflects a unique nodal system which covers both cities, with the obvious implication that there is a strong interaction between these two cities, in contrast to other towns on the east side which are still small and somewhat isolated.

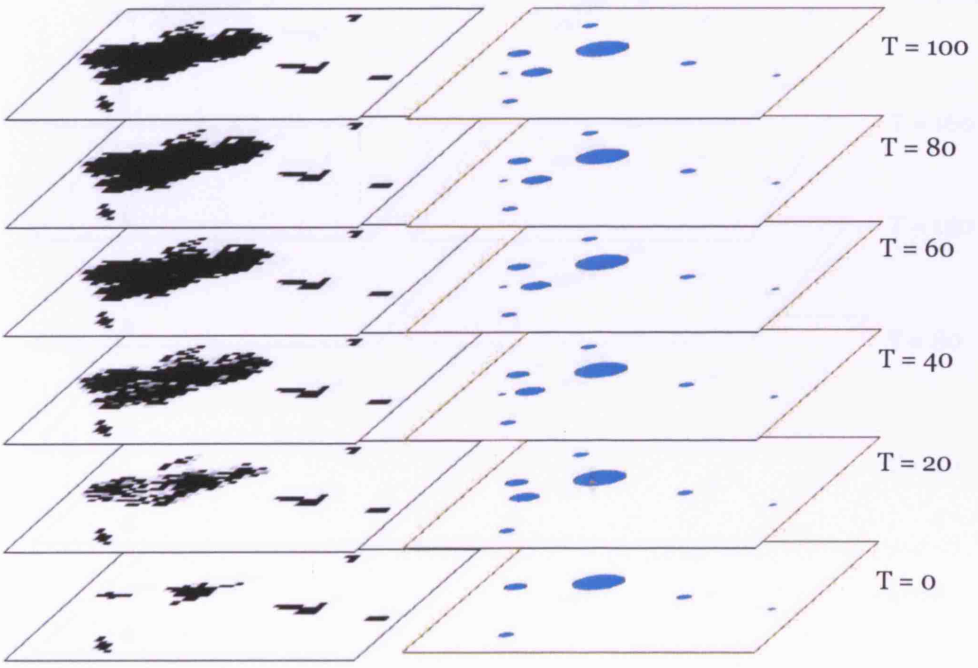


Figure 5.53 The Sequence of Nodal Maps Generated from the CA with Constraining Factors

The Nodal System with a Constrained CA Simulation in the 250x250 Space

All previous experiments discussed above have been performed on a low-resolution space but in moving our experiments closer to reality, we have run the simulation on the higher-resolution 250x250 space and we present the results briefly in this section. As shown in Figure 5.54, the various sequence of images depicts a growth pattern that looks a lot closer to real cities. Because of the constraining factors, it is clear too that the urban area is concentrated around the original centre and tends to grow along the transportation routes (see Figure 5.15).

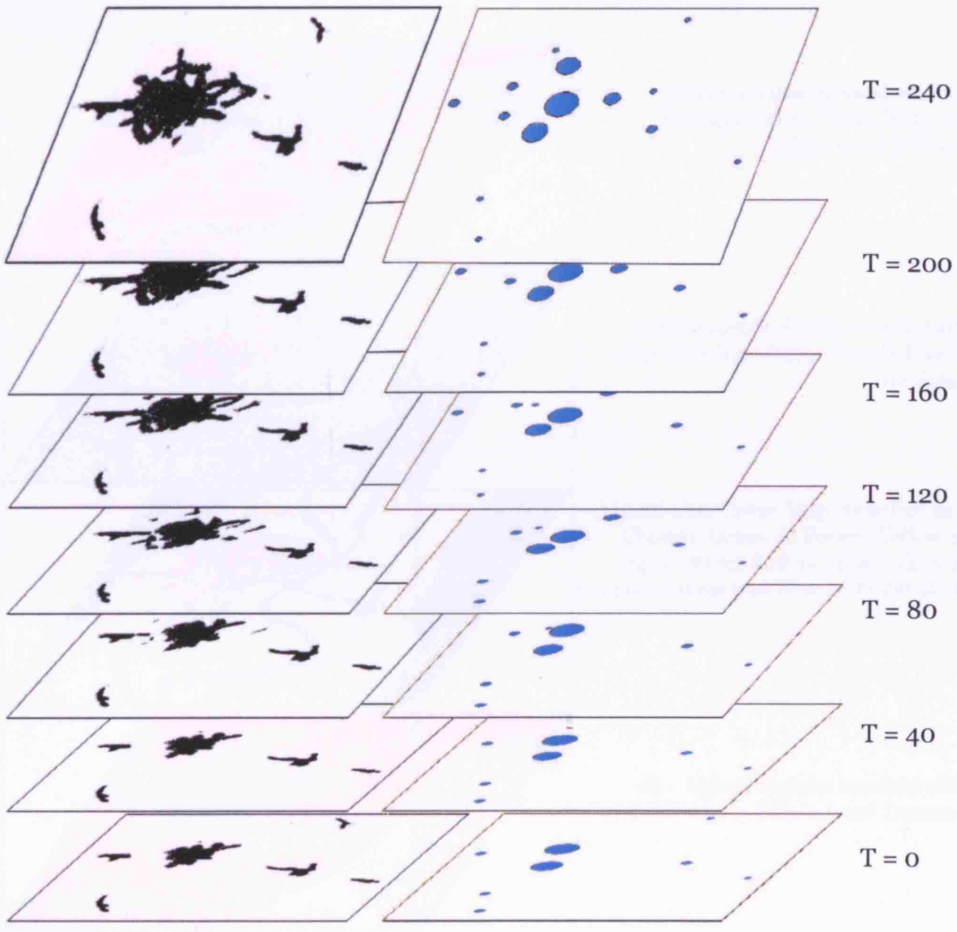
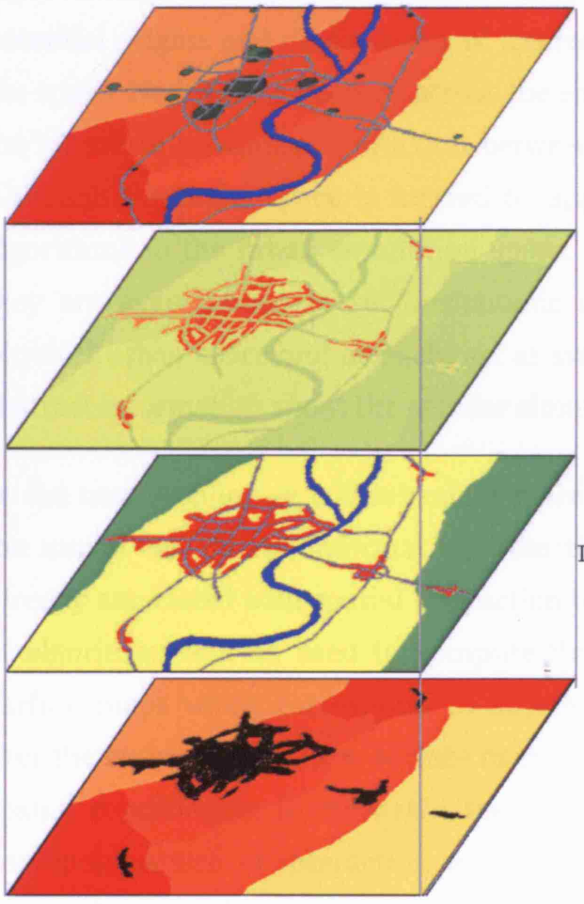


Figure 5.54 The Sequence of Nodal Maps Generated from the CA with Constraining Factors on 250-Pixel Space

At state t = 0, the moving kernel passes over the urban area which sits astride the two big nodes in the centre of the space as well as the other smaller nodes. At t = 240, the nodal system of the main city has become more complex as it composed of many more nodes generated from the growth of urban cells. In the meantime, the main city has been developing and fusing with another town on the west side and this forces the nodal system into a new pattern. The node of the small town then becomes part of the nodal system of the main city which again is evidence of an increase in spatial interaction between these nodes, a feature that we will deal with more explicitly in the next section.



a) Nodes Superimposed with the Transportation, Water Body and Urban Land Demand

b) Dasymetric Population Density Map showing where People Live in the Space

c) Land Use/Cover Map classified as 5 Classes: Green as Forest, Yellow as Agriculture, Red as Urban, Grey as Transportation and Blue as Water Body

d) Urban Pattern overlaid with Urban Land Demand

Figure 5.55 Extended Maps Layers Associated with the Simulation at t = 300

The simulation also produces other extended dynamic maps which were introduced earlier in the dynamic map experiments. Figure 5.55 illustrates these maps which provide another glimpse of the information with respect to how the urban growth from the simulation relates to other spatial variables and how its nodes can reveal patterns of spatial interaction. Figure 5.55a shows the nodal systems overlaid on the urban land demand and transportation maps. In Figure 5.55b and 5.55c, the maps show the population density and land use/cover respectively while in Figure 5.55d the interactions between the simulated urban area and the urban land demand are shown at time t = 300.

In summary, the nodal space is an imaginary space which is created for representing the expected centres, hierarchies of these centres, and potential origins and destinations of interaction between these centres in the space. Nodes are dispersed across the entire cellular space and provide the structure on which interaction between the urban areas takes place. Although the nodal space is formed by applying a set of comprehensive algorithms to the urban simulation space, these are still abstract in that they are aggregations of socio-economic attributes associated with the physical urban space and as such, act as symbolic centres to convey more abstract information about the cellular simulation.

In the next section, we will introduce more experiments which show how the model can extract information from the nodal space which is more directly associated with spatial interaction between the nodes. Another set of algorithms will be used to compute this interaction and to generate surface maps which correspond to different levels of spatial interaction over the entire space. These surface maps, of course, will be finally used as spatial conditioning to constrain the CA simulation, and to reflect the emerging pattern of interaction which of course is a two-way feedback process between the physical and socio-economic simulations.

Spatial Interaction and Potential Forces

The nodal space which we have shown above as providing a sense of the interaction between all nodes in the system, can be used to understand the relative importance of centres and how large centres interact with other smaller centres using size and distance to represent their importance and strength of relationship. However interpreting the actual meaning of size and distance is complicated as the model applies quite complex algorithms to extract and use such data. To examine such new information, what we will actually do is transform the relationships within the nodal space into new maps using various mathematical techniques based on taking the interaction maps and computing spheres of influence using the Voronoi diagrams and the gravity model.

As presented earlier in Chapters 3 and 4, the potential surface technique which extends the gravity model into the spatial interaction approach, is central to the simulation. The potential surface model is a mathematical function which measures the relationship between mass and distance are as a function of the nodal space created in the last section. The spatial arrays of mass and distance generate various patterns of potential, which are applied to measure two sets of spatial interactions which we call the centrifugal and centripetal forces. In this section, we will show how we can model these two somewhat invisible or abstract forces which when converted into raster map layers, are used to influence allocation at the cellular level.

In this experiment, two different scenarios were set up based on different alternative parameters of the distance function. In Chapter 3, a constant exponent value is defined for calculating the potential of all nodes in case of the centripetal forces. On the other hand for the centrifugal forces, the model employs a non-linear function which computes the exponent values using definable ranges of distance from the urban centres. These two forces are not testable from real measurements but the model converts them into probabilities of development for each cell in the space. We will

now examine two experimental scenarios with the influence from the urban centres set at 5000 meters and 10000 meters respectively.

The 5000-Meter Centrifugal Forces

This experiment was implemented on the 250-pixel panel with a 50 meter resolution associated with each pixel with the initial parameters given above as in the constrained CA simulation. Various images are illustrated in Figure 5.56 and then represented in 3 dimensions in Figure 5.57 below. As mentioned before, the values in both surface maps are the probabilities of conversion to urban for each cell and these values are mapped as a set of graduated colours as seen in the figures. In the image at t = 0 in Figure 5.56, both surface maps are quite similar. Considering the centrifugal surface map after 40 iterations, the potential intensities begin to spread out into areas surrounding the centre in a northwest direction. In contrast, the centripetal surface map shows a concentration of high probability cells in the central region.

Figure 5.57 illustrates 3-dimensional views of both surface maps at different time states. The 3D surface models demonstrate how potential makes us aware of different concentrations, flows, and accumulations. Deeper areas with a red colour characterise high potential with lighter colours indicating lower potential. If we imagine the potential surfaces as plains or fields on a 3D surface, the maps in Figure 5.57 clearly show how the plain area associated with the centrifugal surface swells up while on the other hand, the flatter plain areas on the centripetal surface shrink over time.

Other interpretations to be made from the surfaces in Figure 5.57 involve flows and accumulations. From the concentrated central region when discussing flows, the trend in urban development is guided by growth to other easier or higher potential directions. For instance, at time t = 300, the potential terrain in the northeast direction of the central region is starting to transform into a lower-gradient terrain. Consequently, it is possible that the urban area tends to spread out from the centre while the

3D surface map also conveys other information when combined with other map features as can be seen in Figure 5.58.

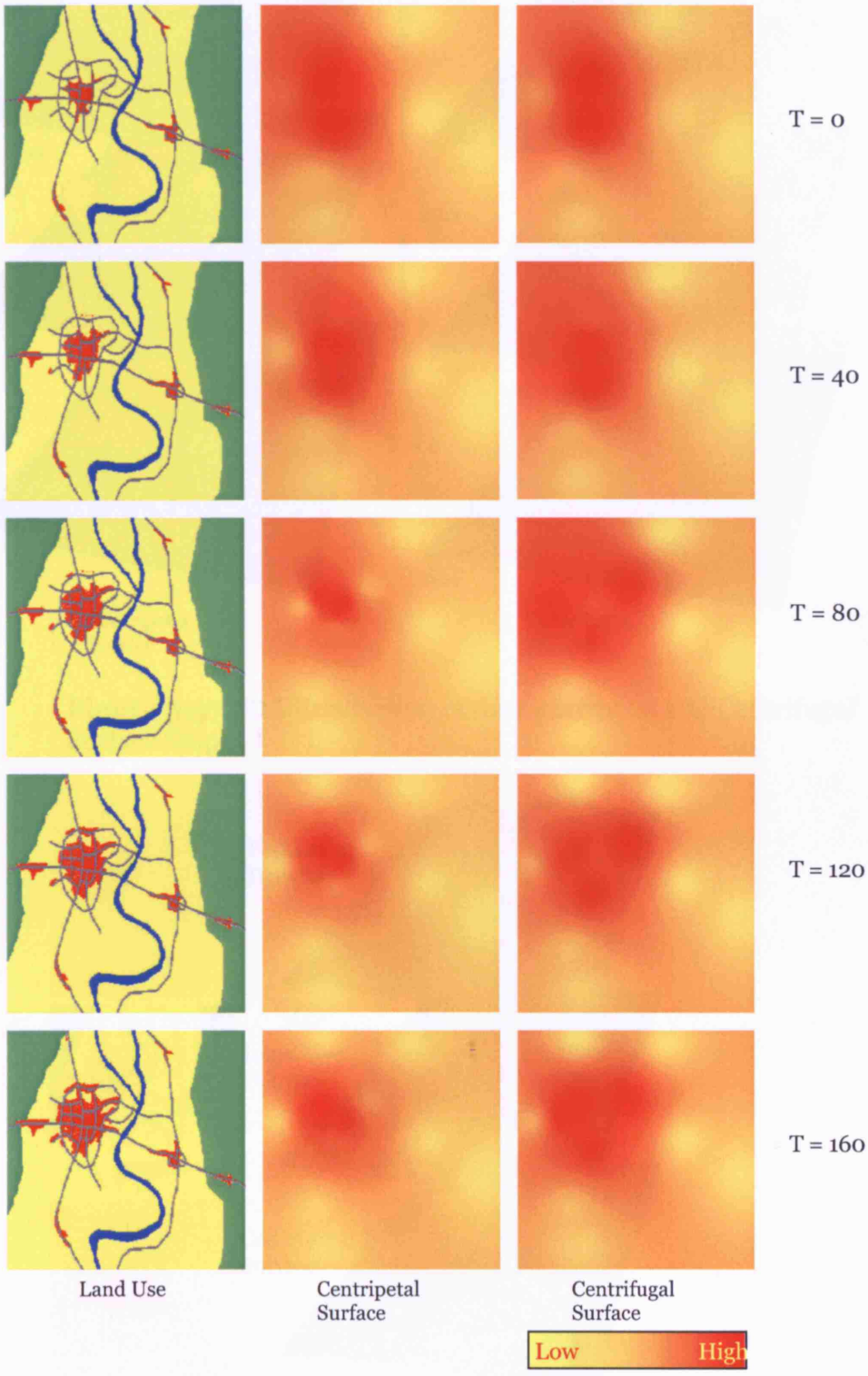


Figure 5.56 Urban Growth and the Centripetal and Centrifugal Surface Maps for 5000-Meter Sphere of Influence of the Scenario 1

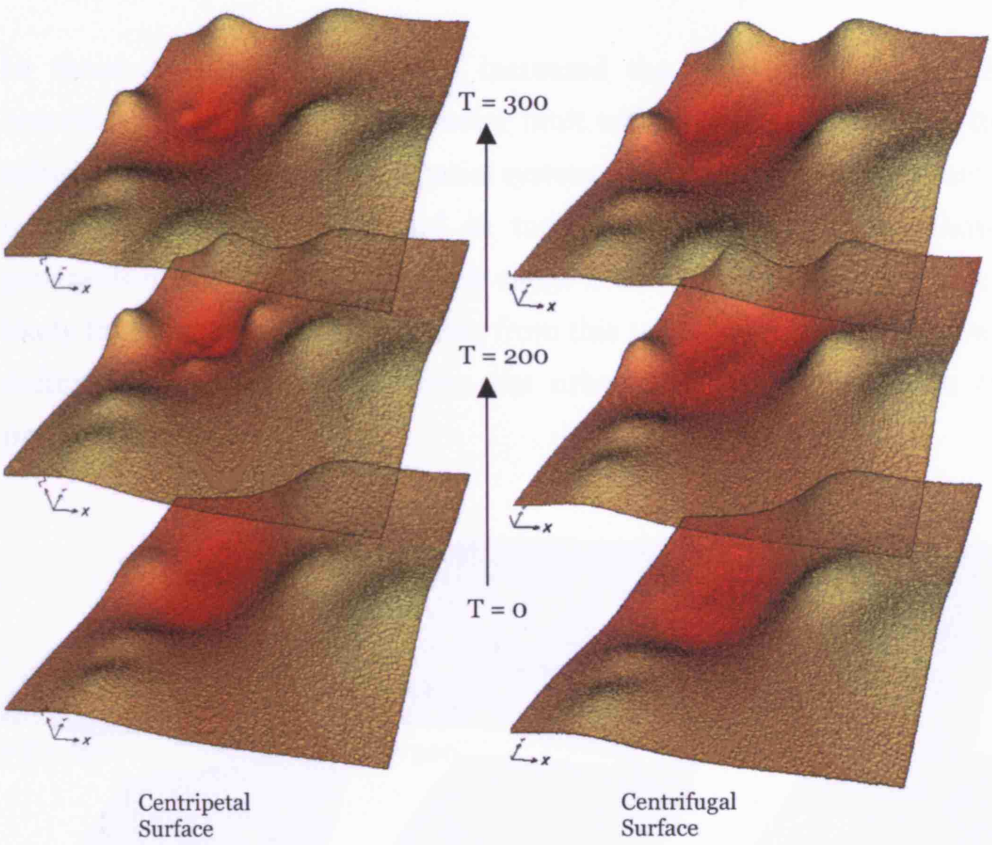


Figure 5.57 3D Visualisation of the Centripetal and Centrifugal Surface Maps

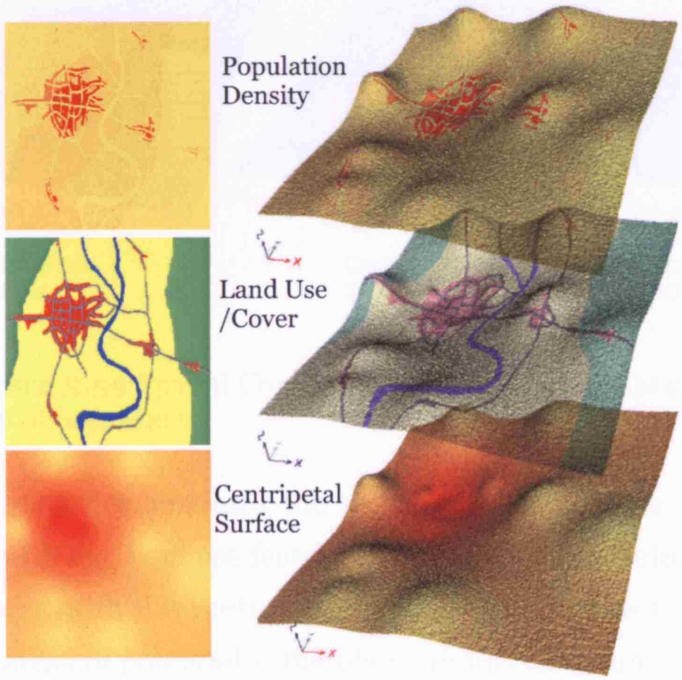


Figure 5.58 Spatial Comparisons of 3 Maps at the t = 300 of Scenario 1

The 10000-Meter Centrifugal Forces

In these experiments, we have increased the sphere of infuence for centrifigal forces to the 10-kilometer limit with the consequence that this sphere affects all parts of the spatial system. These two surface maps act as constraints on the model and in turn change as the CA simulation generates new urban cells. As their range is across the entire system, it is likely that the urban form resulting from this 10000-meter simulation will definitely show differences from the urban simulation created in the previous experiment.

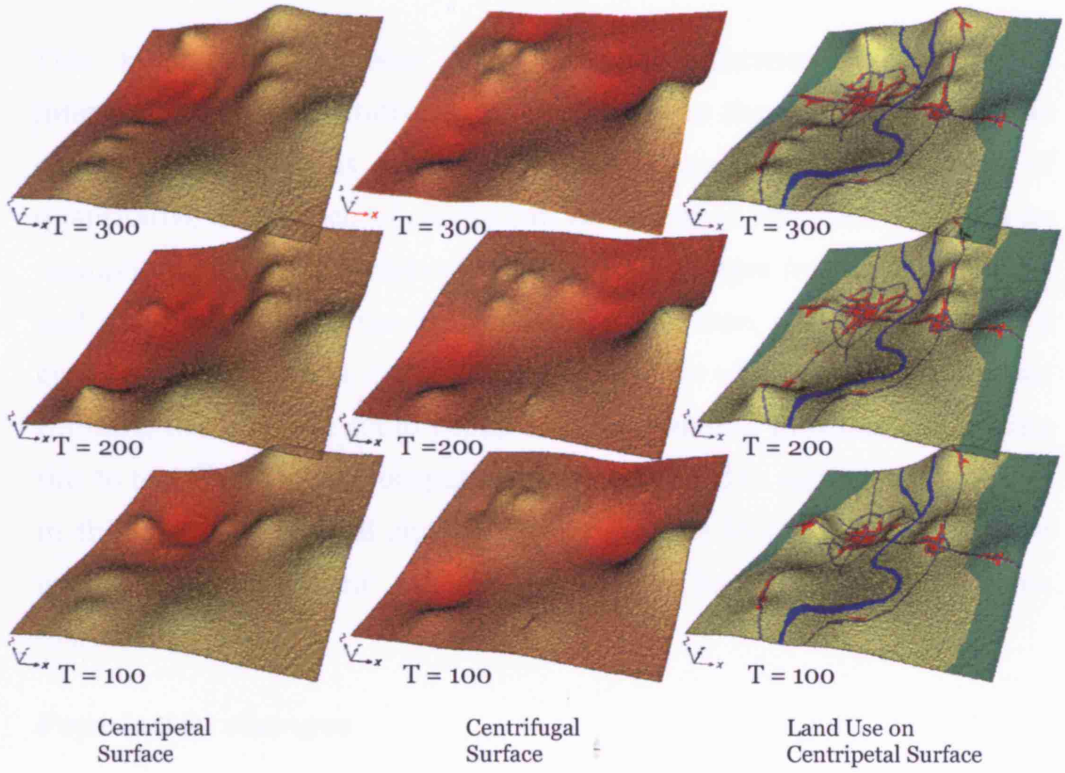


Figure 5.59 Spatial Comparisons of Two Surface Maps and Land Use Map of Scenario 2

Figure 5.59 demonstrates the sequence of images for the two opposing forces with the land use features overlaid on the centripetal surface maps. Compared with the centrifugal surfaces from Figure 5.57, it is clear that flatter areas of potential – the plains in this landscape - begin to broaden around all areas, creating isolated areas at the centre of the city. The

surface then shows that there are flows and accumulations from the central city to all surrounding areas by concentrating on the lower-hierarchy nodes shown as swamp-like terrain on the images.

We have developed many experiments here but before we finish and engage in a little discussion, we will look at some additional visualisations from the simulation so that these can give us some additional insights into the way the model is working.

4. Quantitative Outcomes

This section demonstrates other possible outcomes which can be interpreted in a quantitative way. In addition to the many visualisations shown in the previous sections, the model can provide a sequence of quantitative data which of course can be output as text. This can thus be manipulated using spreadsheet and similar packages from which tabular and graphical outputs can be derived. In this case, a simulation with a given set of parameters was selected to show how when the simulation was running, the program could capture all the required information, writing this to text files. The text output was then subjected to various analysis and in this example, we will simply look at changes in population and land use/cover although any other variables can be handled in the same manner.

Population changes

With the initial state of population of each region as shown in Figure 5.4 above, we have defined population growth rates randomly in each region from a range based on 0 – 3 percent change for each temporal iteration. Based on the dasymetric mapping technique described in Chapter 3 taken from the input population for each administrative area, the model spatially redistributes these population numbers into populations for each urban area cell, in the agricultural sector, and in forest area cells. The graph in Figure 5.60 demonstrates the trends in total populations for each of these

land uses over the entire region. This graph clearly shows different population growth between three sectors, but it is important to note that here we are converting urban cells to populations by using the dasymetric surface as a way of making this transformation.

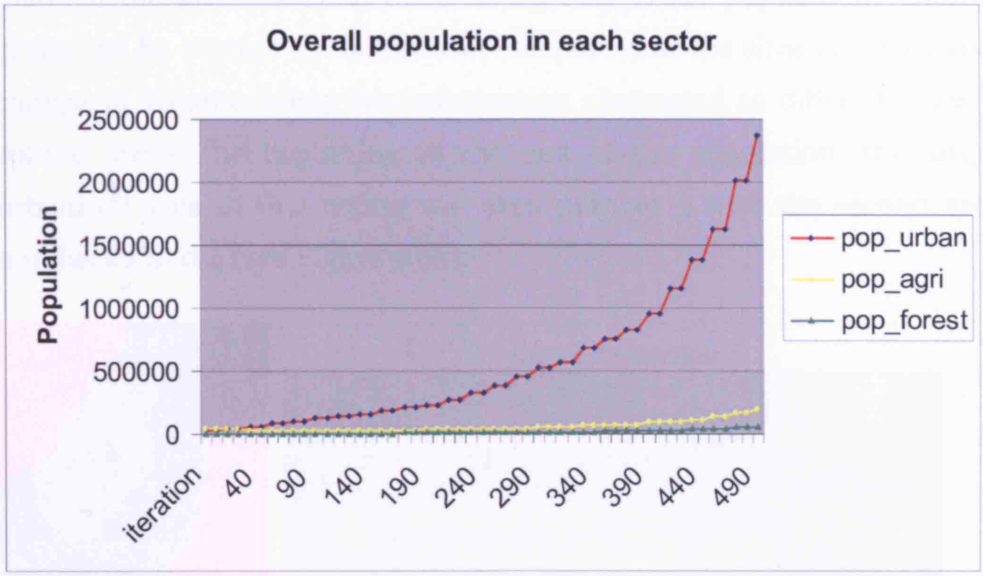


Figure 5.60 Overall Population in Urban, Agriculture and Forest

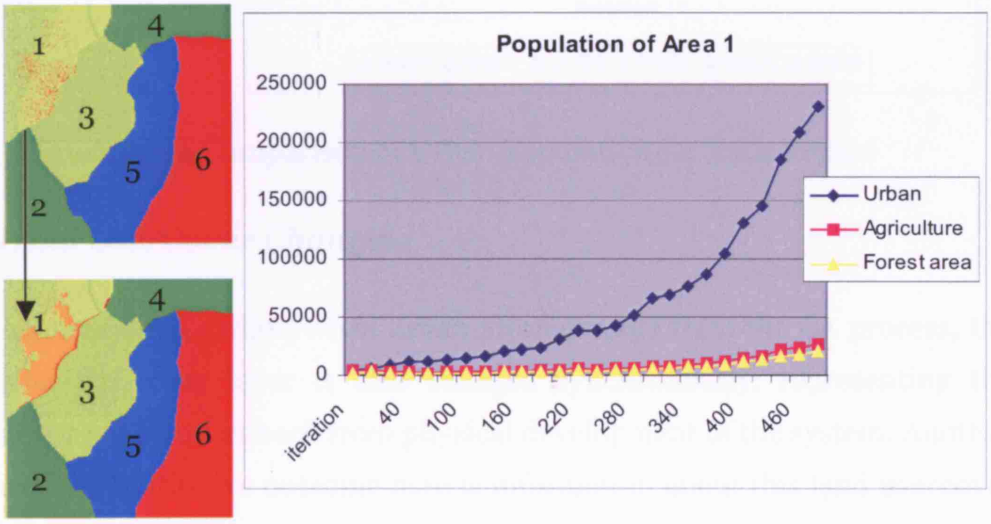


Figure 5.61 Population Changes in Area 1

Furthermore, the numerical file also provides information about the population for each sector in each region. This information can be used for creating graphs as shown in Figure 5.61 where the images to the left of Figure 5.61 depict the population density maps of area 1 at t = 0 and t =

500, and the graph to the right represents the overall trends in population in this area.

The population information of all other regions can be used to create trend maps in the same fashion. Considering only urban populations, another graph can be created so that we can compare how the sizes of urban areas change in volume using this information abstracted as different sizes of sphere. From the beginning to the end of the simulation, the largest urbanised area in this region was area number 3 with the second areas numbers 2 and 4 (see Figure 5.62).

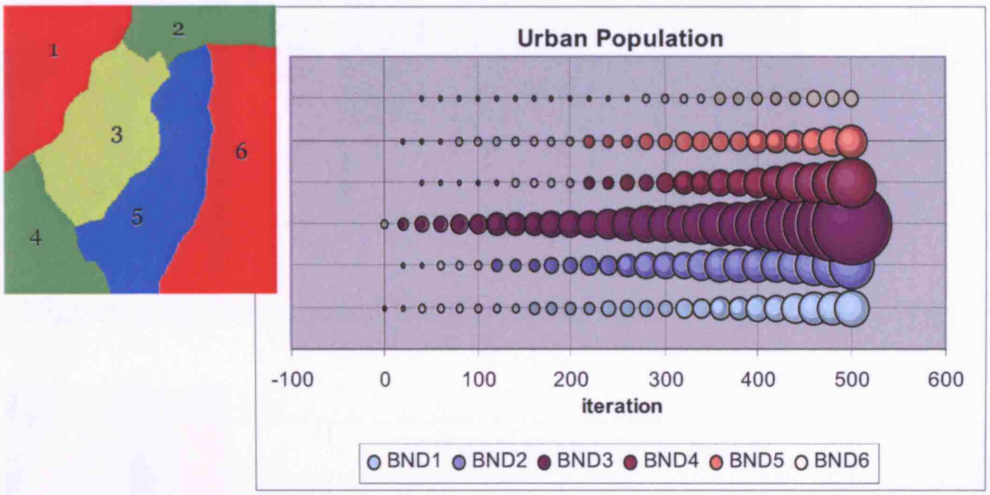


Figure 5.62 Comparison of Urban Population for Each Region

Land Use/Cover Changes

As mentioned earlier, when urban areas emerge from the CA process, the land use/cover layer is also changed synchronously, representing the direct positive feedback from physical development of the system. Another useful quantitative outcome here is information about this land use/cover change. The model tracks the number of cells in each land use sector in each administrative region in the same way as it tracks the population data.

Figure 5.63 is a graph created in order to view these trends in changes in land use/cover in the model space. For the entire area, the graph shows that agricultural cells decline rapidly whereas the line graph for urban cells clearly rises until about the 350th time iteration when the number of urban cells approaches that of agriculture. The numbers of forest cells slightly decrease over the period.

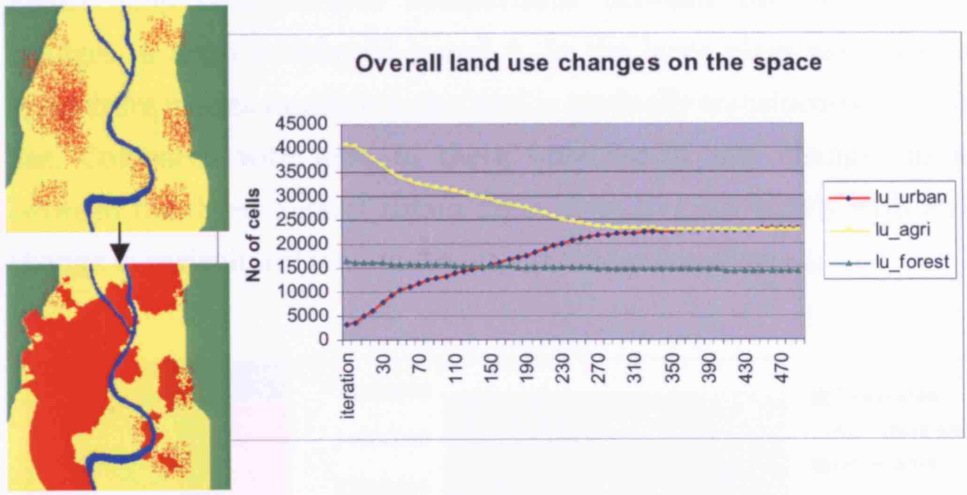


Figure 5.63 Overall Land Use/Cover Changes

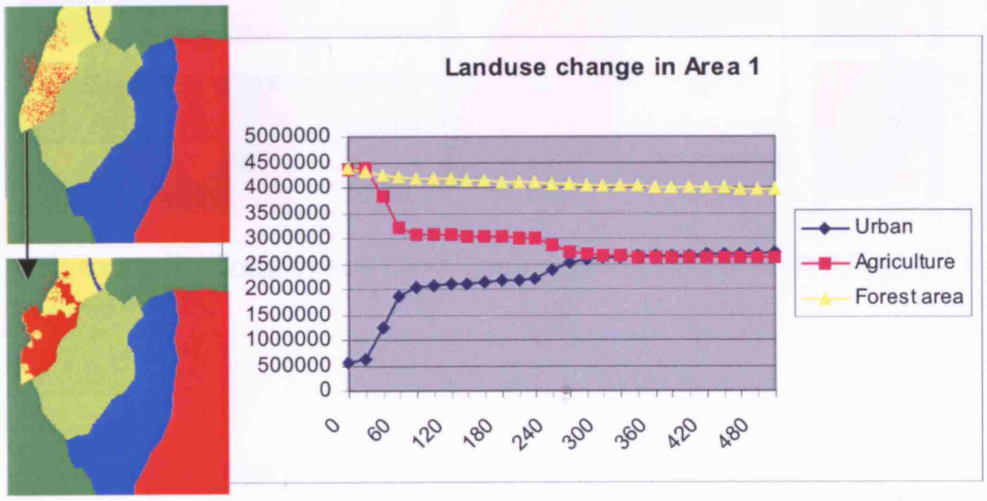


Figure 5.64 Land Use/Cover Changes in Area 1

We can of course translate these quantities into physical units rather than counting the numbers of pixels at the relevant level of resolution and in this way, we can measure the area in units of square metres. We show land use / cover in square meters in Figure 5.64. For region 1 at the initial state, two largest areas are agriculture and forest, but the graph shows that the

forest areas slightly decline as does the agricultural areas whereas urban land increases rapidly. Of course as the model does not deal with transitions from agriculture or from forest to uses other than urban, then this is perhaps to be expected due to the fact that agricultural land is closest to the urban land cells at the beginning of the simulation.

Figure 5.65 demonstrates comparisons between the land use/cover changes in areas numbered 3 and 6. In the large plain associated with agriculture in area number 3, the land is gradually transformed into urban use. Compared with area 6, there were hardly any changes in areas between the three classes; urban areas seem to grow slowly with a slight change in agricultural land and a little more for forest areas.

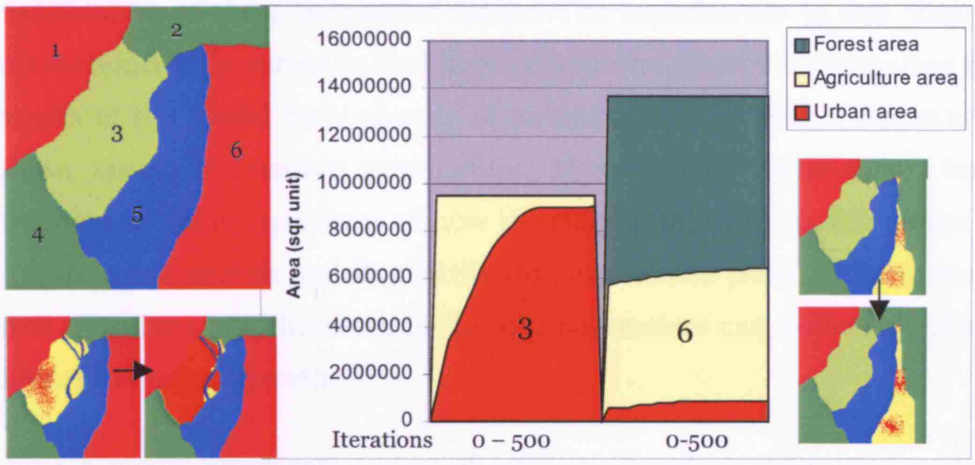


Figure 5.65 Comparisons and Trends Land Use/Cover between Area 3 and 6

5. Conclusions

In this chapter, all the concepts and techniques which were discussed in Chapters 3 and 4 have been embodied in the simulation model and then used as a basis for experimentation, the results of which we have largely illustrated in graphical fashion. Artificial but plausible sets of data at different scales were described and used for in-depth experiments with the simulation model. The cruder data sets in terms of levels of spatial

resolution was used for examining the impact of transition rules from the CA simulation as well as for exploring the operation of the diffusion process. We also used the cruder resolution model to explore the evolution of the nodal space in visual terms so that we might observe how nodes are created from cells in the space. On the other hand, the higher resolution dataset was useful in testing how spatial constraints influence the CA simulation, and how changes in urban form affect these same constraints.

The model contains several algorithms which are driven by a very large number of definable parameters assigned in many possible ways. There are no strict rules for combining these parameters to represent a real situation and thus we are forced to choose parameter values which are plausible. On the other hand, it is not possible to test the model under every combination of these parameter values. The experiments in this chapter are therefore only partial in that they provide snapshots for comparing the results of the model for a sample of parameter values within the overall phase space of possible parameters. Most of our experiments have provided useful comparisons of how the simulation leads to differences in overall urban growth and its distribution and at this point we have some idea as to how modifications of the key parameters can noticeably affect patterns of urban growth.

Our CA model represents a kind of artificial life which grows and evolves in an urban space. Based on the set of transition rules that apply to cells and their neighbours over time together with the set of random functions which drive the allocation process, this kind of unpredictability is enough to produce natural patterns from which order emerges from the bottom up. Except for the randomness that is intrinsic to the way allocation takes place in these types of simulation, all other CA components are controllable to some degree by assigning parameter values and thresholds to various parts of the model. These, of course, pertain to cells, their neighbours, the rules for their transition, and the time over which such transitions take place. At the cell level in addition to the state transitions that are the consequence of the CA rules, changes in these cell states can come from locally constraining conditions such as topographic features,

accessibility and so on. Neighbourhoods are generally manipulated by a transition rule such as the central cell changing its state to another if there are at least, say, 3 live (urban) neighbouring cells. In a spatial sense, neighbourhoods can be defined as 4 or 8 adjacent cells (the von Neumann or the Moore respectively) or they can embrace an 'action-at-a-distance' using the search radius instead. The neighbourhood effect can also embody spatial functions that are used to generate new maps fed back to the simulation model such as the focal function, accessibility, and so on. Lastly, time is another variable that can control the speed of growth in the simulation. In this prototype, these variables were integrated into the model and represented in each section of the experiments.

The first two experiments covered a number of tests dealing with the neighbourhood and the time effect involving the time taken for transitions to occur. Without any constraining conditions, cells developed from local transition rules applied to the neighbourhoods over every cell in the space. Changes in different number of cells in the neighbourhoods generated a progression of urban cells with patterns that we illustrated in Figure 5.31 above. In the case of the temporal effect, this model shows how the length of the time period for development of change could be used as a variable in the CA simulation. For example, cells in the developing state have a lower potential to grow into urban cells as those which are already in a developed state. Therefore, when manipulating standardised scores to the levels of development, this will clearly affect the values associated with the neighbourhood effect. From these results, this clearly shows that increasing the development period in the model affects the rapidity of growth. These ideas are clearly seen in Figure 5.21 which we discussed in detail earlier.

When looking at an urban area in terms of classified satellite images, almost all such images reveal some curiosities in the pattern of development. Urban areas are rarely agglomerated to one cluster; instead, there is a hierarchy of clusters appearing as large and small areas distributed over the space. We might assume that where remote settlements exist within a well-developed urban region, there is another

process that causes those tiny cells to be generated over the space at a distance from the main cities. In this model, diffusion is the process that is developed to handle this issue. The outcomes shown in Figures 5.33 to 5.40 represent effects of adjusting various parameters in the diffusion algorithm which affects cell development. From these examples, we learn that the diffusion process can lead to quite unpredictable spatial patterns but even though these can be constrained, diffusion can cause many changes in the characteristic morphology of the urban area.

Closer to reality, how an urban area actually evolves and grows is limited by many other factors that we can only hint at here. Not all areas that are classed as urban cells can develop; for instance, wetland areas, mountainous areas, or areas restricted by political mandate. However we are able to build in various constraints that we can observe from data into our more general CA simulation. As seen in the experiments above, the multi-criteria analysis method which we incorporate as a weight-overlay analysis is the main approach used to embody constraint factors as raster maps into the simulation. In the constrained CA experiments in this chapter, the model also results in modifying the spatial data used to create the constrained maps because of changes in the state of the cells in the CA space (as we illustrated earlier in Figures 5.41 to 5.48).

The last two sections dealing with the experiments involve two important parts of the model: the nodal space and spatial interaction. To create a sense of size and distance, a set of algorithms was developed to transform patterns of urban cells in the CA space to be defined as nodes on a new space which we illustrated in Figure 5.49. Additionally, we observe gradual changes in the size and distribution of nodes during the growth process which were illustrated in Figures 5.50 to 5.55. In practice, attributes in the nodal space are then transformed to two other spaces based on the concepts from the spatial interaction model theory. These new maps are created dynamically and used as two additional constrained conditions which in turn affect cellular transformation through the dynamic map components. The first of these, the centripetal surface map, provides information regarding the concentration of population that increases the

capability of growth under the spheres of influence of the main centres in the system. On the other hand, the centrifugal surface map is created by readjusting the distance function in the spatial interaction model with this map clearly revealing rather different characteristics. Compared to the other force surface, it seems that there is some potential flowing across the space from the main centres to surrounding nodes. Both surface maps are not only used in the simulation process, but they can also be employed to visually represent the trends and directions of urban growth.

The model can also output another kind of data that can be used to create related numerical information as spatial aggregates. This chapter demonstrated a couple of examples involving making this kind of output intelligible in terms of aggregate graph analysis of trends in land population and in land use/cover. Thus far, the most important model components have been examined and visualised using sets of fabricated data. This chapter has examined a range of experiments from the simplest to the most complex, from simple exploration of the cellular process to situations in which all the constraint information has been incorporated. It is clear as these experiments have progressed, that urban form and the map layers shown become progressively closer to what we might expect to be characteristic of an evolving and growing urban reality. However, even as we have come closer to the reality, we are not able to say how well the model represents real world phenomena until we actually confront the model with real world data. In the next two chapters, we will begin to do this, first examining real data. In the next chapter, we will describe the context to the applications to towns in Thailand, describing the dynamics of urban growth from multi-temporal satellite images which we will classify and reformat for use in the model.

CHAPTER SIX
City Case Studies: Background and Data

1. Introduction

This chapter begins to confront the model with real data and to this end, we initially focus on general descriptions of two case study areas of Thai cities: Chiang Mai and Phitsanulok. We will simulate growth and change in recent decades in these two cities using the model whose results we will present in the next chapter.

First, the city of Chiang Mai has a long history going back at least 700 years and as an historical city, it is a well-known place for tourists while playing a key role as in the commercial life and service provisions for the population in the Upper Northern Thailand region. Chiang Mai is not only the largest city in Northern Thailand but it is also the second-ranked largest city in the whole country. It is currently an attractive city for business and commercial investment and this, more than anything else, has lead to rapid growth in urban construction and infrastructure. The Chiang Mai area is particularly interesting for not only is its urbanising rapidly, its place in northern Thailand means that it is an excellent example to show how local urban change interacts with the wider pattern of urban-regional development. This is one of the key features of the model we have proposed.

Phitsanulok, on the other hand, is a medium-sized city in the Lower Northern part of Thailand. The city although smaller and growing more slowly than Chiang Mai, is the dominant city in its region because its advantageous location surrounded by many other cities has given it a high profile in national policies. This has defined it as one of the major Thai cities with potential for investment in the commercial and industrial sectors. Moreover, it is part of a regional mega-project called the 'Indo-China intersection' that is directing investment into this area which is

likely to increase its growth rate in the near future. These factors also suggest that Phitsanulok is a significant example for studying trends in city growth in this type of region which is typical of growth strategies in developing countries.

The first part of this chapter will introduce various details about both areas in terms of their backgrounds and short history. This will give the reader some sense of the importance of these cities while at the same time providing information about their general physical and geo-demographic characteristics. Finally, recent changes in urban form in terms of the growth of both cities will be discussed. All this information will be important in analysing and interpreting the results from the various simulations which we will build in the next stage.

We will also detail various map data and related information which we will transform into formats that complement the data used in the model. Most of this data has been prepared using GIS and image analysis techniques which convert all the map layers into digital form, extract all the thematic maps required from information in the GIS database, and lastly, transform these outputs to raster maps. Some specific remote sensing techniques for analysing satellite images will be briefly described as well. All the maps and data which have been prepared and created at this stage will be visualised and explained in the last part of this chapter.

2. The Background to the Case Study Cities

In this section, we will provide general information about the two case study areas of Chiang Mai and Phitsanulok in some detail. This will be separate from detailed data concerning these two cities which we will present in the next sections in terms of the data layers used in the model. Firstly, we will introduce broad pictures of the two cities at their regional scale in terms of their relative locations, their administrative territories, their topography and so on. Subsequently, once we get some sense of the scale of these study areas, we will focus next on the city level where we will

recount a brief history of each city, the political background, socio-economic culture and structures, as well as illustrating the urban form of both cities. Lastly, we will discuss urban development processes in each city so we have some sense of the way these have changed which is essential in evaluating the model in the next chapter.

Prior to getting to know details of the case studies, we will introduce the shape and geographical characteristics of Thailand in order to clearly perceive its whole image. Figure 6.1 represents the locations of the principal provinces superimposed on its topographic features. Thailand is located in the Southeast Asia region and has a size of approximately 514,000 square kilometres. It shares its border with Myanmar in the west and north, Laos in the northeast, Cambodia in the east, and Malaysia in the south. It is administratively divided into 76 provinces with Bangkok Metropolitan being the nation's capital city located in the central region.

The central region is an immense abundant floodplain known as 'Thailand's Rice Bowl' which has been formed over many centuries from deposits from river floods. This plain encompasses the principal provinces belonging to this region such as Ayuthaya, Nakorn Sawan, Sukhothai and Phitsanulok. Areas in the northern part of Thailand are mostly mountainous. There are a few plains that lie between these mountainous areas which are suitable for urban settlement while agriculture too is difficult. Along the west side of this region lies the border between Thailand and Myanmar and the north to the territory of Chiang Rai province and the boundary of Laos, is the area known as 'the Golden Triangle'. Most of the major provinces are centred on the plains between hilly areas such as Chiang Mai, Chiang Rai, and Lam Phang.

The topography of the northeast region is a huge plateau separated from the north and central part by the periphery of the tableland as well as being bordered from Laos by the Mekong River in the eastern part of the region. This region is also known as 'Isan: the cradle of civilisation' where the world's oldest Bronze Age civilization prospered some 5,000 years ago.

Important provinces in this area are Nakorn Ratchasima, Khon Kaen and Udon Thani.

The eastern region lies between the sea and the range of the northeast tableland. The land in this eastern part connects to Cambodia which is the starting point of the longest coast in Thailand, originating in the Gulf of Thailand. The main provinces in this area are Chon Buri, Trat and Chantha Buri. Lastly, the southern region which is the peninsula, sprawls from the central region around the capital into the south. This area shares its western border with Myanmar while the lower region is the land in between the Andaman Sea and the Gulf of Thailand. The margin of the peninsula to the south is the border of Thailand with Malaysia and the major provinces are Prachub Kirikan, Phuket, Songkhla and Surat Thani.

Chiang Mai

Location and Territory

The province of Chiang Mai in the Upper Northern part of Thailand is situated between latitude 17-21 degree north and longitude 98-99 degree east, with the Chiang Mai basin approximately 310 metres (1,027 feet) above mean sea level. From the Bangkok metropolitan area, it is approximately 760 kilometres by rail and about 700 kilometres by road.

Chiang Mai province shares its border with the Shan State of Myanmar which lies astride the watershed of a mountain range called Dan Lao in the north. In the south, the neighbouring areas are the Sam Ngad district of Tak province with the Mae Tuen water channel, with the Doi Riam and Doi Luang watersheds acting as the border. Chiang Rai, Lam Phun and Lam Pang provinces lie on the east side and are defined by the deep channel of Kok River, and the watersheds of mountain ranges on the border close to Chiang Rai province. The watersheds of Doi Hunn Huay La, Doi Chang Nung and Mae Ping River channels act as the border for the Lam Phun province. The west side connects to the districts of the Mae Hong Son province which are bordered by the watersheds of Doi Kiew Daeng and Doi

Prae Mueang, and the river channels of Mae Rit and Mae Oin (see Figure 6.2).

Figure 6.1 Thailand, Showing the Main Cities

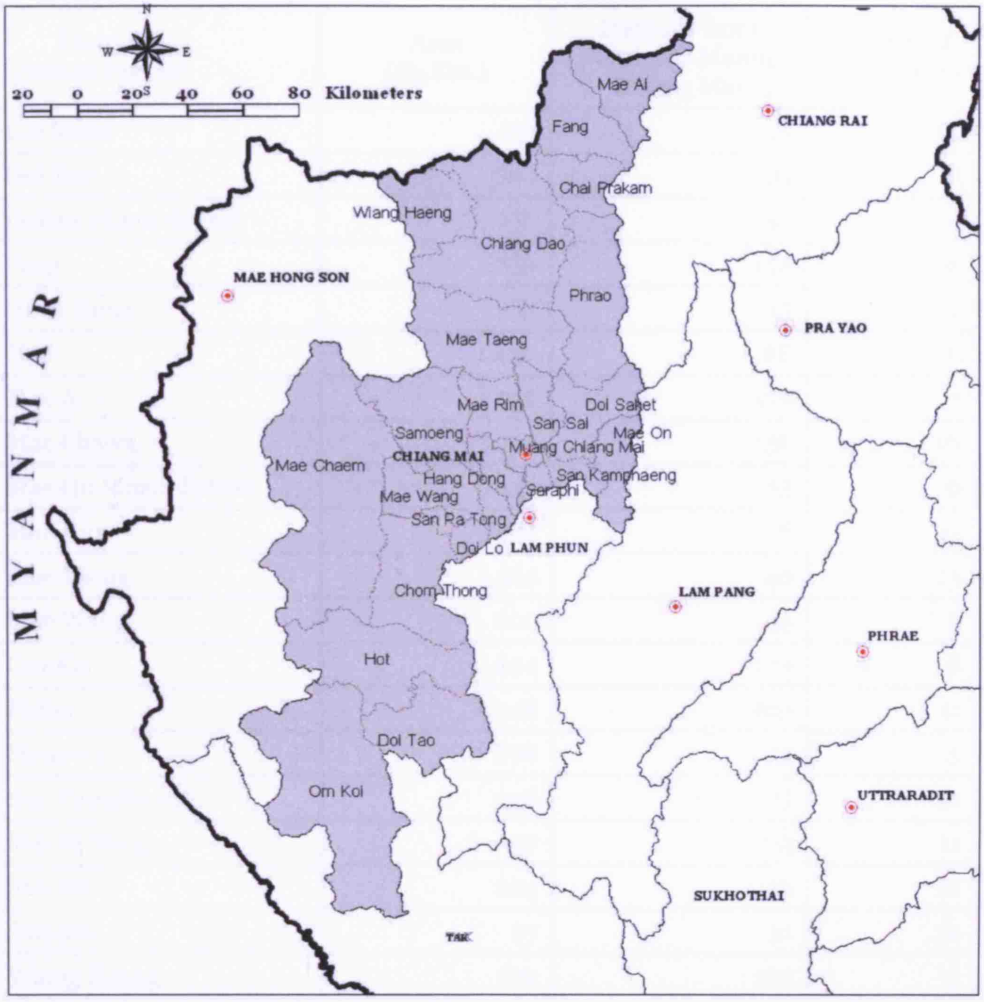


Figure 6.2 Political Boundary of Chiang Mai Province and its Districts

Chiang Mai is administratively divided into 22 'Amphoes' or districts as shown in Figure 6.2. Each is divided into smaller administrative units called 'Tambons' or sub-districts and municipalities. The areas, distances from districts to the province, and the number of sub-districts are shown in Table 6.1 below.

Table 6.1 Administrative Divisions of Chiang Mai Province

District /Minor District	Area (Sq.Km.)	Distance from District to Muang Chiang Mai	Number of Sub-districts
Muang Chiangmai	166	-	16
Chai Prakan	511	131	4
Chiang Dao	1,882	68	7
ChomThong	754	58	6

District /Minor District	Area (Sq.Km.)	Distance from District to Muang Chiang Mai	Number of Sub-districts
Doi Saket	671	18	14
Doi Tao	804	121	6
Doilaw Minor district	218	45	4
Fang	888	154	8
Hang Dong	263	15	11
Hod	1,430	88	6
Mae Ai	737	174	7
Mae Chaem	3,361	156	10
Mae On Minor district	443	33	6
Mae Rim	444	8	11
Mae Taeng	1,363	40	13
Mae Wang	602	35	5
Om Koi	2,094	179	6
Phrao	1,148	103	11
Samoeng	898	54	5
San Kamphaeng	218	13	10
San Pa Tong	177	22	11
San Sai	285	12	12
Saraphi	97	10	12
Wiang Haeng	672	150	3

(Source: http://www.investmentthailand.com/en/LocaDet.asp?p=p04)

Geographic Features

The total area of the Chiang Mai province is 20,107 square kilometres which is linear in shape from north to south. The topography of the province may be roughly classified into three simple types: 1) mountainous areas, 2) plains between the valleys and 3) river plains. First, approximately 80 percent of the total area of the province are hill areas with plains at the base of the hill slopes. The mountainous regions to the west of the province act as the natural border between Chiang Mai and Mae Hong Son provinces. The peak is about 2,565 metres above mean sea level at Doi Inthanon. Most of the hill lands are covered with dense vegetation, with many headwater sources and outlets of the main rivers of the north of Thailand including Ping River, one of the main rivers passing

through Chiang Mai city, and finally originating in the Chao Phraya River. Second, the smaller plains between the valleys are generally occupied by small villages supporting agricultural activities. On the hill slopes and in the valleys, paddy fields on terrace areas close to small communities dominate the landscape. Lastly, the plain areas in the river basin are land on which the two main cities Chiang Mai and Lum Phun and many minor towns are located. Most land above the plains are agricultural areas with several types of cultivation (see Figure 6.3). The main river, the Ping River that originates from headwater sources above Doi Luang-Chiang-Dao, runs from north to south and is the main water source for the agricultural sector and indeed for water intake for households in the cities.

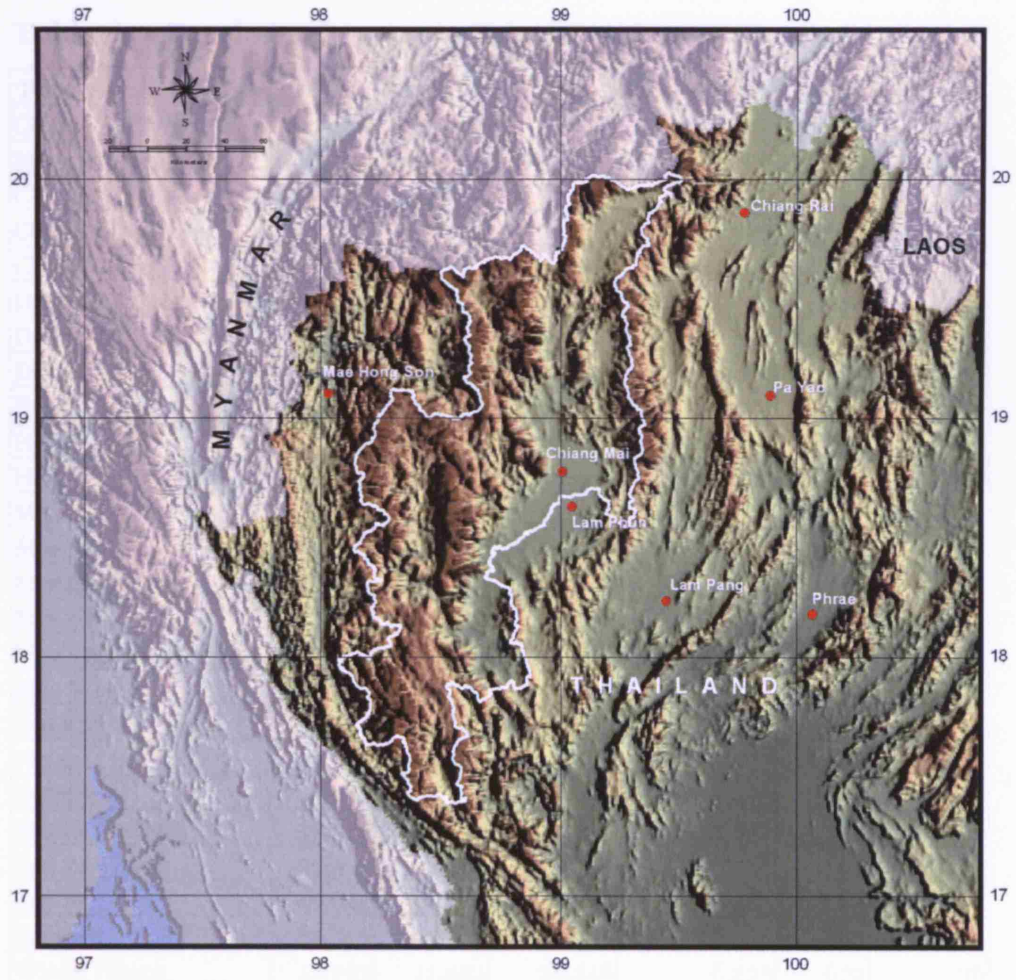


Figure 6.3 Topography of the Chiang Mai Province

Population

In 2003, Chiang Mai had a total population of 1,603,220 (790,107 males and 813,113 females) which gives an overall population density of about 80 persons per square kilometre. The most populated area was Muang Chiang Mai district with 248,340 persons, and given its small area, it had the highest density of 1,496 persons/sq. km. The second most populated was Fang district with 106,956 persons but with a lower population density of 120 persons/sq. km. The least populated area was the Mae On Minor district with a population of 21,859. However, the population density in the Mae Chaem district had only 20 persons per sq. km. Table 6.2 and Figure 6.4 present the detailed population characteristics of the Chiang Mai province.

Table 6.2 Population in 2003 in Chiang Mai Province

Province/Districts	Male	Female	Total	Household	Area	Density
Chiang Mai Province	790,107	813,113	1,603,220	551,696	20,126	80
Muang Chiang Mai	117,595	130,745	248,340	112,660	166	1,496
Chai Prakarn	22,498	21,415	43,913	13,643	511	86
Chiang Dao	39,076	37,348	76,424	20,922	1,882	41
Chom Thong	32,678	33,170	65,848	18,508	754	87
Doi Law Minor	13,835	14,176	28,011	9,599	218	128
Doi Saket	31,372	32,595	63,967	23,801	671	95
Doi Tao	14,521	14,212	28,733	9,161	804	36
Fang	54,047	52,909	106,956	34,555	888	120
Hang Dong	36,910	39,043	75,953	29,281	263	289
Hod	18,809	19,109	37,918	10,705	1,430	27
Mae Ai	37,115	36,227	73,342	22,714	737	100
Mae Chaem	33,905	32,452	66,357	13,506	3,361	20
Mae On Minor	11,095	10,764	21,859	7,288	443	49
Mae Rim	40,072	40,083	80,155	27,258	444	181
Mae Taeng	37,902	37,865	75,767	25,434	1,363	56
Mae Wang	15,443	15,602	31,045	8,991	602	52
Om Koi	24,934	24,889	49,823	12,494	2,094	24
Phrao	27,138	27,356	54,494	17,419	1,148	47
Samoeng	11,788	11,156	22,944	6,472	898	26
San Kampaeng	35,544	38,255	73,799	27,652	218	339
San Pa Thong	37,572	40,422	77,994	27,165	177	441
San Sai	47,625	52,670	100,295	40,181	285	352
Sarapee	35,840	39,197	75,037	26,064	97	774
Wiang Haeng	12,793	11,453	24,246	6,223	672	36

(Source: Department of Provincial Administration 2003)

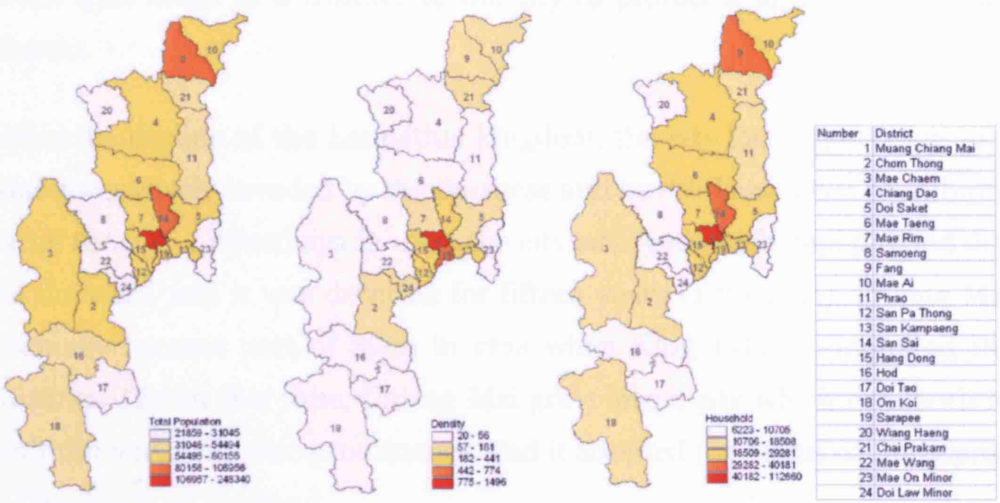


Figure 6.4 The Distribution of Population, Density and Households in the Chiang Mai Province

The City

Brief History and Background

Chiang Mai (New city) is an historic city which is at the centre of Lannathai Kingdom, the land of a million rice fields, which has been the colloquial name for the area of Chiang Mai and the Northern region for last century. Chiang Mai city was founded when King Mengrai, the first king of Lannathai, began to create his empire in the Kok River Valley in the mid-13th century, where Chiang Rai city, the first city of Lanna, had been founded. After occupying Haripunchai city (Lum Phun today) and securing joint leadership of Phayao, he sought a more suitable area for his new central headquarters and found the ideal location at Chiang Mai in the Ping River Basin.

King Mengrai, chose the area for the new city between the Ping River which lies to the east and Doi Suthep on the west side. He built the royal city of Chiang Mai on the high ground to the west. He designed the outline of the city in a geometric-rectangular shape, excavating an 18-meter wide defensive moat surrounding a brick and earth wall approximately 2,000 meters from the west to east and 1800 meters from north to south. The

walls thus acted as a frontier to the city to protect it against raids from Burma.

After the decline of the Lannathai kingdom, the city lost importance, and was successively invaded by the Burmese and by the Thais from Ayutthaya. After the fall of Ayutthaya in 1767, the city emerged much depopulated due to the wars, and it was deserted for fifteen years (1776-1791). Chiang Mai formally became part of Siam in 1774 when King Taksin conquered the Burmese. From this time, Chiang Mai grew into a city which rose both in cultural and economic prominence, and it adopted the status of the capital of the north of Thailand.

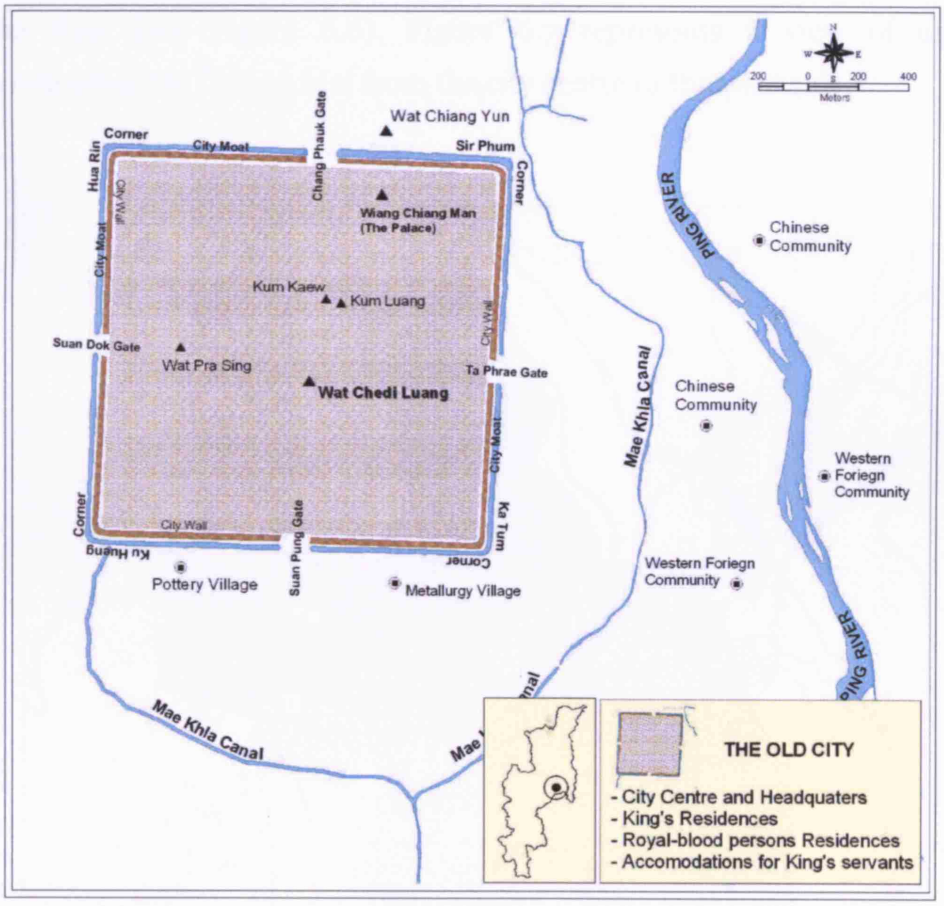


Figure 6.5 Chiang Mai: the Old City

The city has effectively been the capital of the north for 700 years staying in the same location. Its growth began within the 1800 x 2000-square metre blocks behind the moat and wall of the old city, but currently it

stretches out in every direction with urban growth rapidly increasing during the last 20 years.

Urban Morphology and Growth

From its original 2.7 square kilometres behind the walls and its moat, Chiang Mai now covers 14 sub-districts or 'Tambon'. To examine this growth and distribution, it is useful to first summarise information about Chiang Mai city in its administrative municipality boundary, although the city has already grown far beyond this territory. Chiang Mai municipality is approximately 35.85 square kilometres, covering the large big plain on both sides of the Ping River although the city is skewed to the west side of the river (see Figure 6.6). Figure 6.7 represents a view of urban development in Chiang Mai from the city centre in the west side.

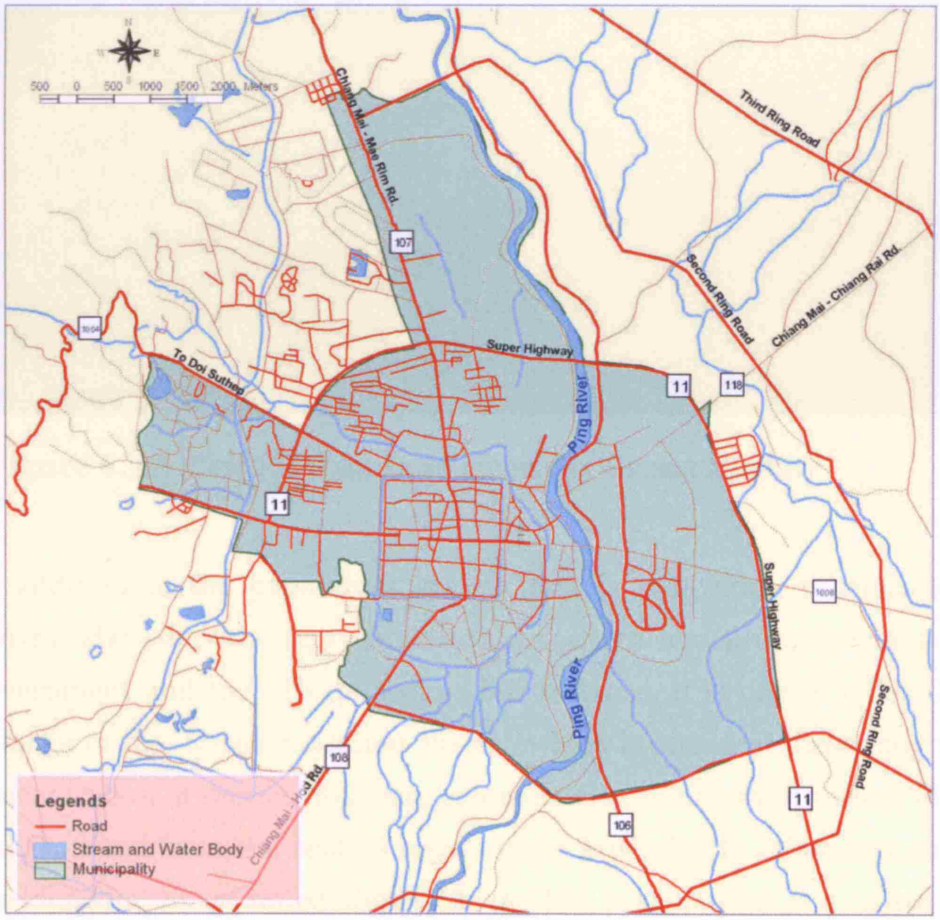


Figure 6.6 The Municipality of Chiang Mai

In 2003, the overall population in Muang Chiang Mai district was 248,340 persons (approximately 16 percent of all the provincial population). From this we estimate that 158,720 peoples lived in Chiang Mai municipality, with a density of roughly 4427 persons per square kilometre. The density patterns reveal a picture of an emerging huge demand by urban residents for housing in the future and this is reflected in the increase in dwellings which has proceeded rapidly from: 58,708 in 1993 to 65,112 in 1998 and thence to 69,075 dwellings in 2003.

Figure 6.7 A View of Urban Development of Chiang Mai City

In addition to the population growth that is motivating urbanisation, Chiang Mai supports many functions associated and promoted by both government and the private sector. For instance, it is one of cities that belongs to the 'Decentralisation Policy', which is aimed at dispersing the concentration of service, tourism, commercial and industrial sectors from Bangkok and from the central region more generally, to other regions of the kingdom. As we noted above, Chiang Mai is supported as a centre for tourism in the Upper Northern region while it is also a transportation hub

connecting other countries in the 'Great Mekong Subregion' (GMS). These potential factors are all crucial in stimulating the growth of investments in infrastructure, the commercial sector, tourism, and tertiary and quaternary services in Chiang Mai city. Indeed in the model we will apply in the next chapter, these factors ultimately become principal issues in explaining and simulating its urban development for Chiang Mai city.

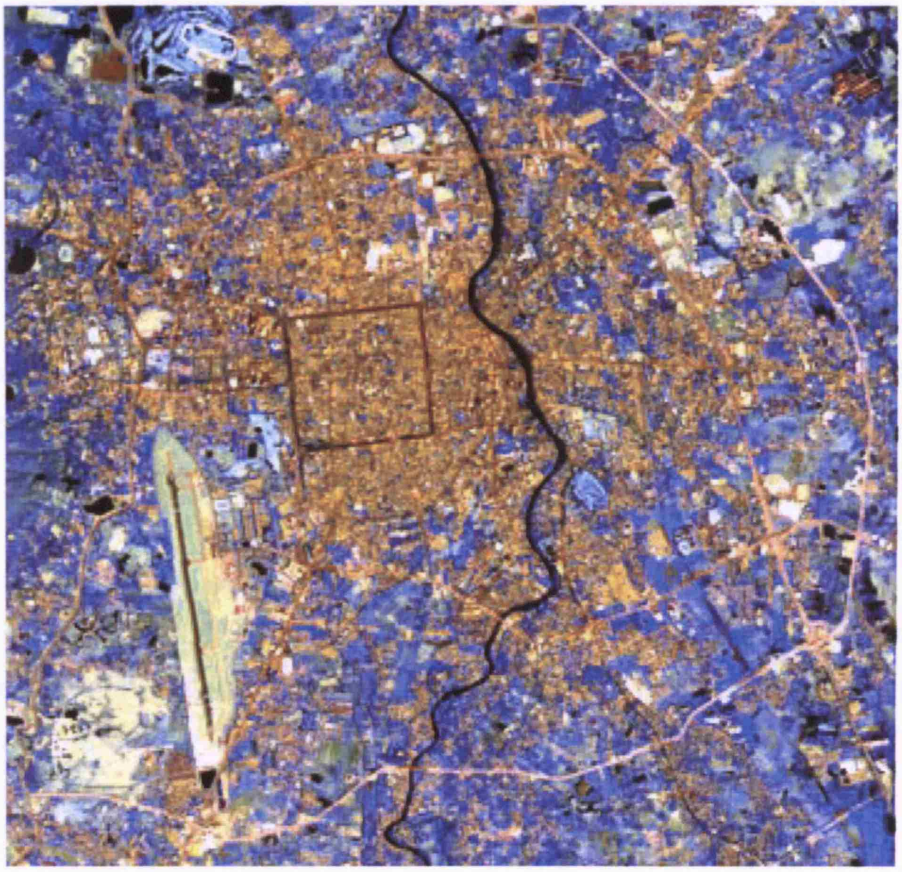


Figure 6.8 A Satellite Image of Chiang Mai City in 2003

The shape of Chiang Mai city is clearly represented in the multi-spectral Landsat satellite image of the city and its surrounding areas in 2003 which we show in Figure 6.8. We will present the details of this image analysis in the next section. The image clearly separates different apparent features which are shown in various colours: for example, built-up areas as light orange; water bodies as black; bare land and transportation features as white and yellowish colour; and vegetation areas as blue. The symmetric rectangular pattern at the centre of the image is the moat in the centre of

Chiang Mai city which is clearly revealed as a feature of the past. The urban patterns appearing in the image show concentrations of densely built-up area from within the old city outwards in every direction, especially to the east bridging both sides of the Ping River. As well as identifying these dense areas, the built-up areas occur in a ribbon development pattern expanding along existing transportation routes.

To the west of Chiang Mai city, the mountain range Doi Suthep which lies from north through to the south connecting to a large mountain range in the east of Mae Hong Son Province, shows how difficult it is to expand in that direction. Because of these topographic barriers in the west, the circular ring roads do not form a complete circle. Currently, three circular rings have been constructed (although the image covers only the inner two). The inner ring road, 'the Chiang Mai Super Highway', was built as a bypass for though traffic but some segments are already urbanised, particularly to the north of the city. The other two ring roads cover the areas more distant from the city centre but even through these rings have been constructed for avoiding traffic problems in the city centre, they have become one of the potential causes of urban expansion and sprawl. In addition to ribbon development along the main radials into the city, the image also shows another pattern of growth around the intersections of the ring road with the main radial routes focussing on the city centre (see Figure 6.8).

Summarising all these changes in growth, it is clear that Chiang Mai has grown from a compact urban form, based on a clear agglomeration of the built-up area within the moat and the areas on the west side of the Ping River, to a form that is more fractal-like, sprawling out into its more distant suburbs. Chiang Mai is continuing to grow in this fashion. This is fast becoming a crucial question for urban planners who need to prepare and manage the sufficiency and suitability of the infrastructure for a city of this size as greater and greater demands will be put upon it in future years.

Phitsanulok

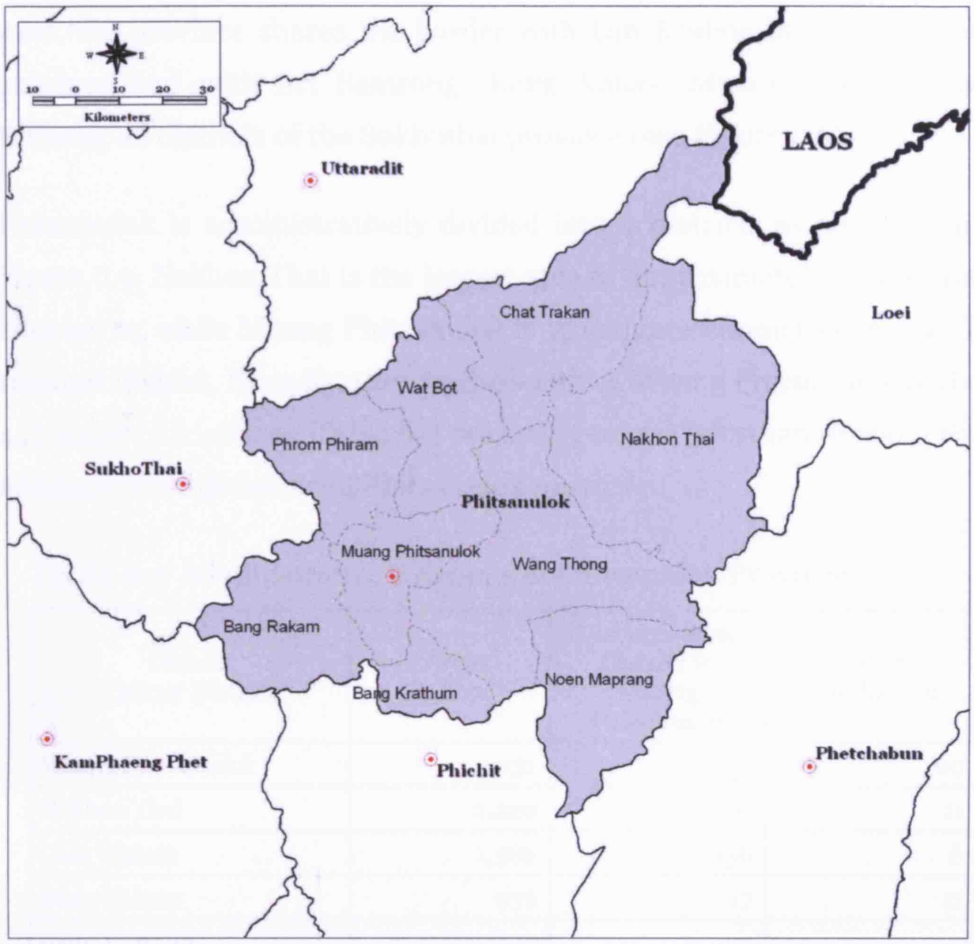


Figure 6.9 Political Boundary of Phitsanulok Province and its Districts

Location and Territory

Phitsanulok Province lies at the boundary between Northern and Central Thailand situated at about 16-17 degrees north and 99-100 degrees east. The city lies on a plain with an average altitude of approximately 40 metres above mean sea level and from the Bangkok metropolitan area it is about 377 kilometres by road. The province shares a short international border with Laos in the northeast formed on the watersheds of the range of Phu (Mount) Soi Dao. To the north, the neighbouring areas are Thong San Kun and Nam Pad districts of Uttaradit province. The eastern areas adjoin the Na Haew and Dan Sai districts of Loei province and Khao Kho and Wang Pong districts of Petchabul province. To the south, the neighbouring

areas are the Sak Lek minor district, the Sam Ngam district of Phichit province, and the Lan Krabue district of Kampaengphet province. To the west, the province shares the border with Lan Krabue in Kampaengphet province and with Sri Samrong, Kong Kailas, Muang Sukhothai and Kirimas, all districts of the Sukhothai province (see Figure 6.9).

Phitsanulok is administratively divided into 9 districts as also shown in Figure 6.9. Nakhon Thai is the largest area of approximately 2,220 square kilometres, while Muang Phitsanulok is 751 square kilometres in size. The smallest district, Bang Kratum, to the south of Muang Phitsanulok is about 447 square kilometres. Table 6.3 presents general information about these administrative divisions of Phitsanulok province.

Table 6.3 Administrative Divisions of Phitsanulok Province

District /Minor District	Area (Sq.Km.)	Distance from District to Muang Phitsanulok	Number of Sub-districts
Muang Phitsanulok	751	-	20
Nakhon Thai	2,220	97	11
Chat Trakan	1,586	136	6
Bang Rakum	936	17	11
Bang Krathum	447	35	9
Phrom Phiram	833	40	12
Wat Bot	1,326	30	6
Wang Thong	1,687	17	11
Noen Maprang	1,030	75	7

(Source: http://www.investmentthailand.com/en/LocaDet.asp?p=p22)

Geographic Features

The total area of Phitsanulok province is about 10,815 square kilometres and has a much more compact shape than the Chiang Mai province. Figure 6.10 represents its topographic features which are composed mainly of plains with average height approximately 40 metres above the sea level. Most of these plains are used for agricultural activities such as paddy fields, cornfields, and orchards, and in 2000, the land occupied for rice fields was approximately 2,455 square kilometres. The plains of the

southern and western areas of the province gradually increase in height to much more mountainous areas in the east and the northeast parts of the province, with the highest peak at 2,102 metres above the sea level. Most hill lands are forest areas located in three important National Parks: Phu Hin Long Khla, Phu Soi Dao and Thung Salaeng Luang. In fact, the forest area is about 50 percent of the total area of the province. The areas which lie between the mountain ranges – the valleys and hill slopes – are mainly agricultural land populated with small rural villages located close to their

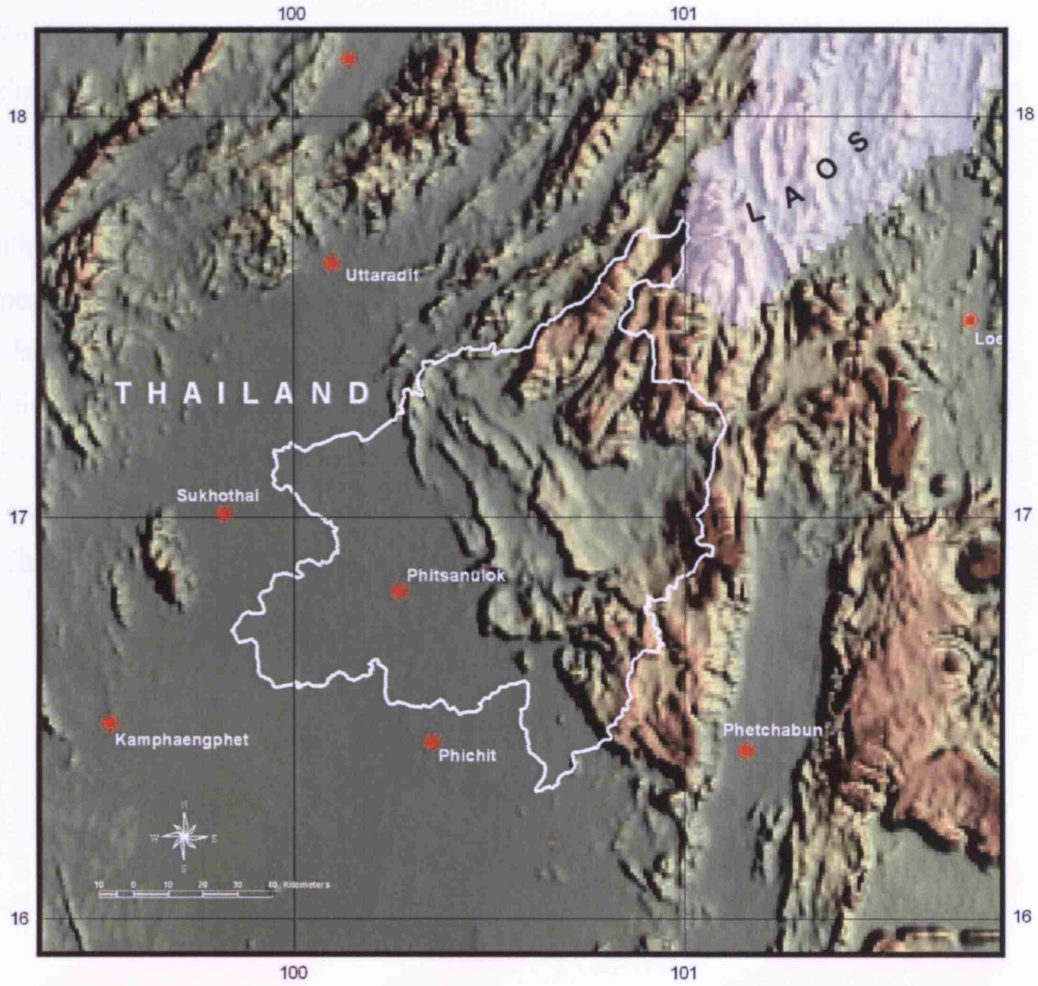


Figure 6.10 Topography of the Phitsanulok Province

farms.

The plain on the west side of the province is called 'Chao Phraya River Plain' and the land above this plain is a river basin area of major rivers, small tributaries, swamps, natural levees, and alluvial terraces.

Phitsanulok province has two major rivers – the Nan and Yom – which both rise in the mountain ranges in the north of Thailand, pass across the province and then join with the Ping River becoming the Chao Phraya River in Nakorn Sawan province. Minor rivers and small waterways can be found across the entire plain, and the profusion of swamps and flood plains provides fertile land suitable for many agricultural activities, with over 35 percent of the land in cultivation (see Figure 6.10).

Population

Table 6.4 presents the population data in 2003, which is also mapped in Figure 6.11. Phitsanulok had a total population of 867,356 at that year (429,182 males and 438,174 females) and a total number of households of 253,554. Overall population density was 63 persons per square kilometre. The most populated district was Muang Phitsanulok with a total population of 270,239 and 94,064 households. Muang Phitsanulok was also the densest area of this province with 360 persons per square kilometre. The second most populated district was Wang Thong with 124,485 persons, but the density here falls to 74 persons per sq km. The least populated areas contain less than 38,000 persons and include the Chat Trakan and Wat Bot districts.

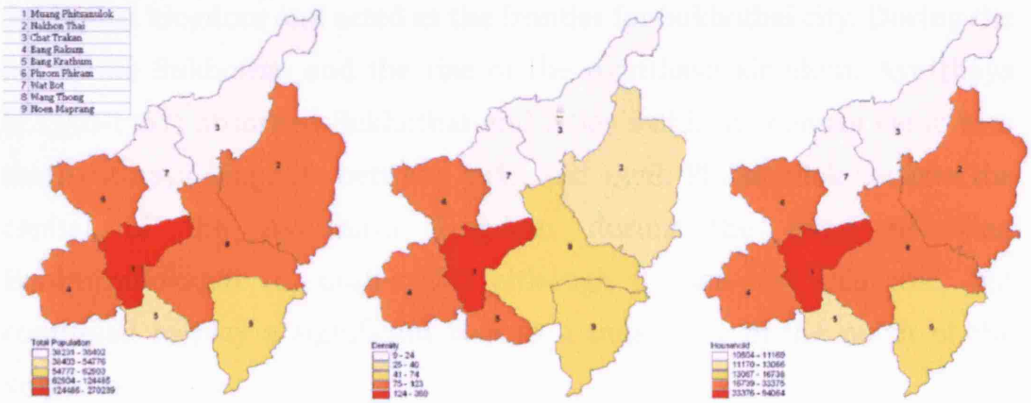


Figure 6.11 The Distribution of Population, Density and Households in the Phitsanulok Province

Table 6.4 Population in 2003 in Phitsanulok Province

Province/Districts	Male	Female	Total	Household	Area	Density
Phitsanulok province	429,182	438,174	867,356	253,554	13,816	63
Muang Phitsanulok	132,391	137,848	270,239	94,064	751	360
Nakhon Thai	44,029	43,947	87,976	23,952	2,220	40
Chat Trakan	19,388	18,843	38,231	10,804	1,586	24
Bang Rakum	48,659	49,964	98,623	25,447	936	105
Bang Krathum	26,860	27,916	54,776	13,066	447	123
Phrom Phiram	45,048	46,673	91,721	24,939	833	110
Wat Bot	18,958	19,444	38,402	11,169	4,326	9
Wang Thong	62,182	62,303	124,485	33,375	1,687	74
Noen Maprang	31,667	31,236	62,903	16,738	1,030	61

(Source: Department of Provincial Administration 2003)

The City

Brief History and Background

Phitsanulok city is known by another name as 'Muang Song Kaew' - the city with two major rivers, and like Chiang Mai, it is one of the most important historical cities in Thailand. Before the period of the Sukhothai Kingdom (c.1238-1438) Phitsanulok was a city controlled by the Khmer Empire which is clear from various historical evidence, for example in the construction of the Wat (Temple) Chulamanee. During the reign of King Ramkhamhaeng the Great (c.1239-1317), the city became a part of Sukhothai kingdom and acted as the frontier for Sukhothai city. During the fall of the Sukhothai and the rise of the Ayutthaya kingdom, Ayutthaya (c.1350-1767) absorbed Sukhothai and cities within its control came into the Ayutthaya kingdom between 1365 and 1378. Phitsanulok became the capital of the Ayutthaya kingdom during the reign of King Boromtrailokanat (c. 1448-1488) although it then lost this role, but continued to play a significant role as a major city in the north of the kingdom.

During the reign of King Naresuan the Great (c.1590-1605) who born in Phitsanulok, the city was used as the military training base for training

armies to capture Ayutthaya back from the Burmese. Phitsanulok continued to play a part in the principal events in Thai government, a role that it still has.

The old city centre was situated around Wat Chulamanee close to the Nan River, approximately 5 kilometres southwest from the modern Phitsanulok city centre. The old city was settled above a natural levee on both sides of Nan River and was bounded with earth and brick walls and moats, constructed for defence from the enemy. The protective walls were not built in a symmetric rectangular shape like Chiang Mai city; instead, they were constructed along the river as shown in Figure 6.12. The city originated from this area but as it has developed, it has spread all around its core on both sides of the river.

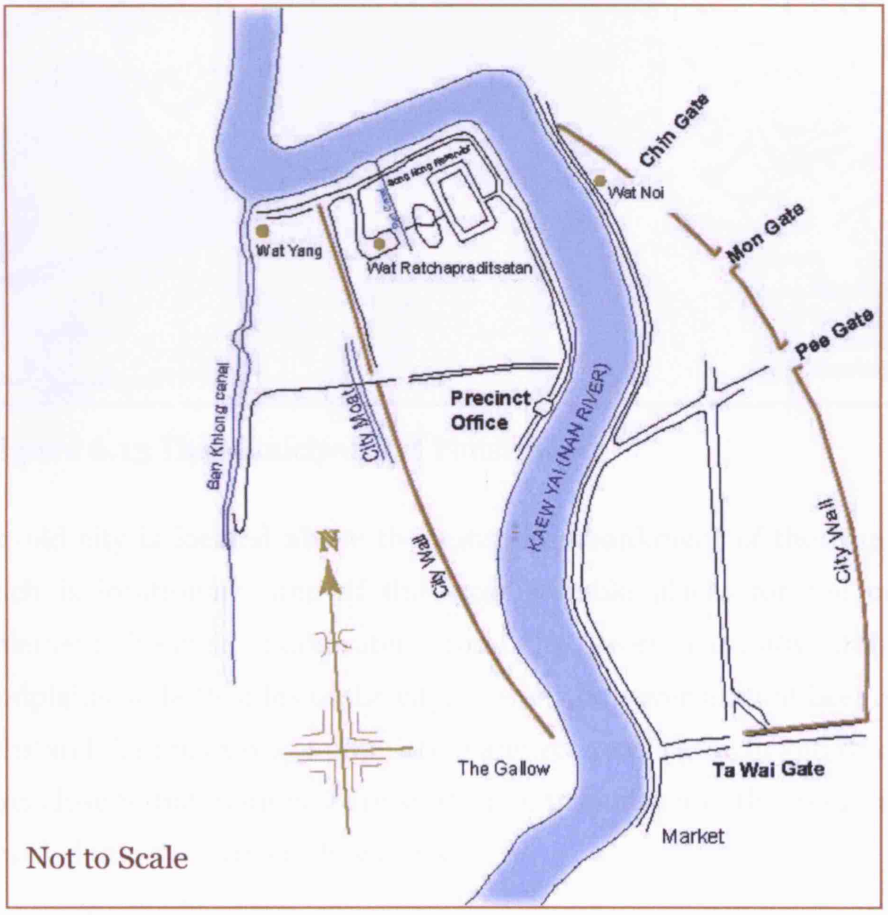


Figure 6.12 Phitsanulok: The Old City

Urban Morphology and Growth

Phitsanulok city today officially covers only one sub-district of Muang Phitsanulok which is the Tambon, Nai Muang. The city is connected to the surrounding sub-districts of Hua Ror and Aranyik in the north, Bung Pra in the south, Aranyik in the east, and Ban Khlong and Wat Chan in the west with both sides of the river fully occupied (see Figure 6.13).

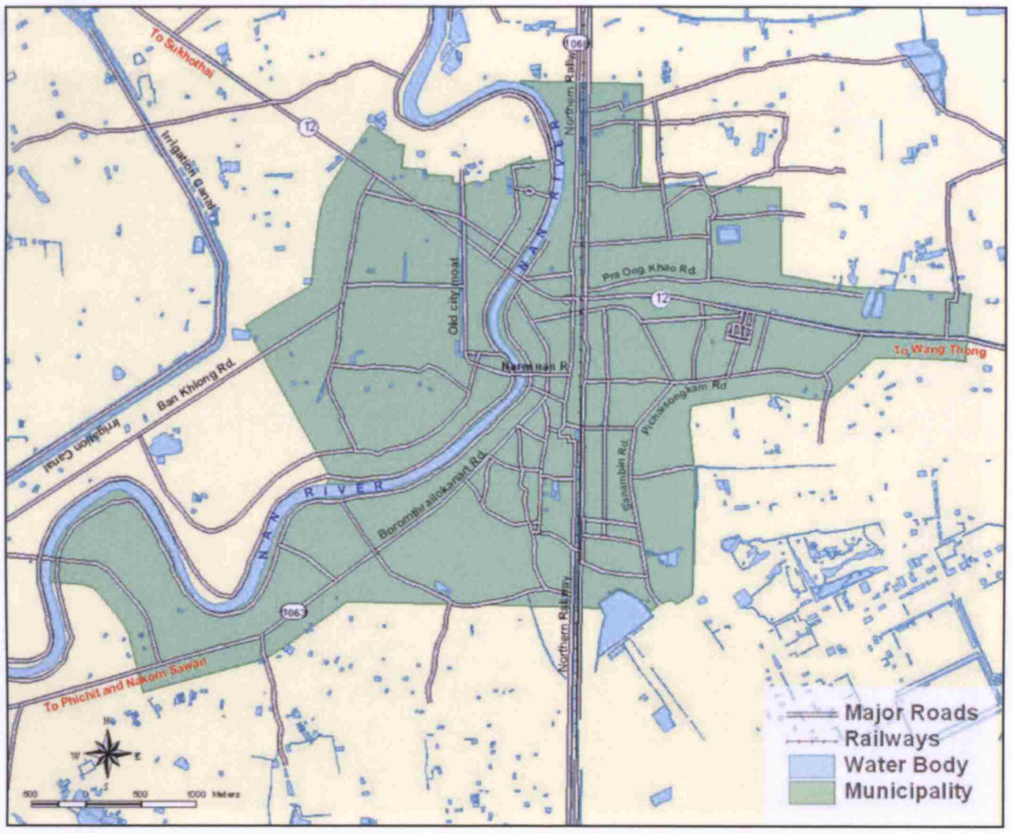


Figure 6.13 The Municipality of Phitsanulok

The old city is located above the natural embankment of the Nan River which is locationally one of the most suitable places for the original settlement because floodwater from the river naturally drains to floodplains on both sides of the city. The city however has not been able to withstand the pressures of population and economic growth and thus rural areas close to the main city are starting to urbanise and the main city has sprawled into the surrounding areas.

In 2003, total population in the Muang Phitsanulok district was 270,239 persons, approximately 31 percent of all the provincial population. 83,956 persons lived in the Phitsanulok municipality, giving a density of about 4,538 persons per sq km. Dense population within the city and rapid economic development has led to dramatic expansion on the periphery of the city and the number of households in Phitsanulok municipality has risen by 50 percent in 10 years, from about 20,000 in 1993 to 30,000 households in 2003.

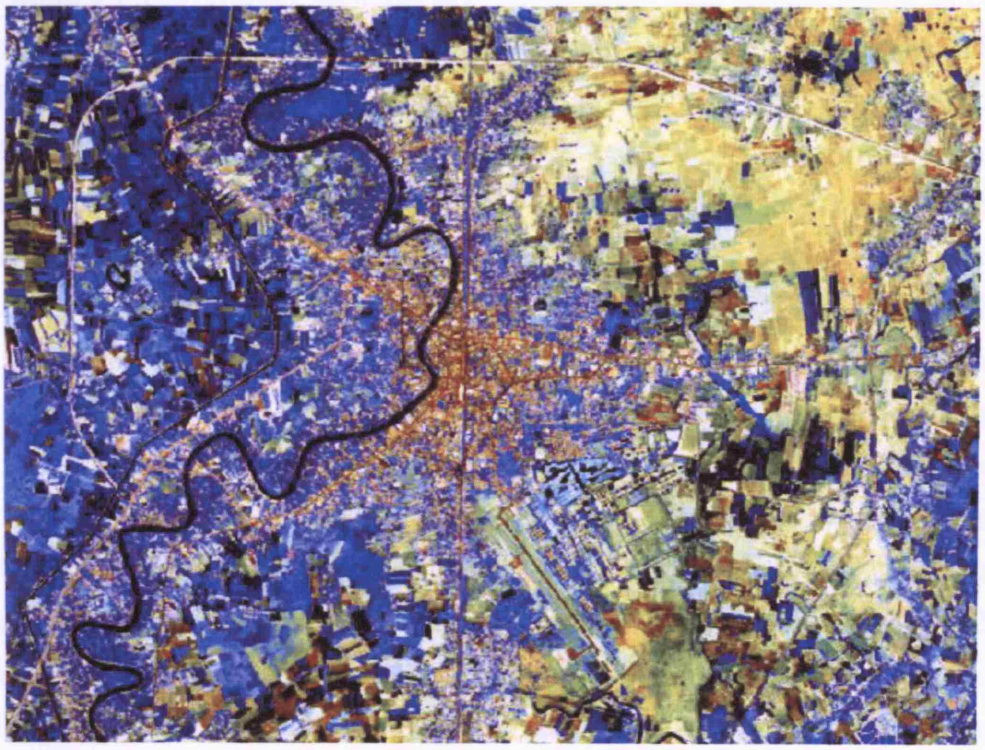


Figure 6.14 Satellite Image of Phitsanulok City in Year 1989

In addition to various national decentralisation policies that have defined Phitsanulok city as one of the growth poles of Thai cities since 1977, at the regional scale, it is the favourable location of Phitsanulok city that is one the main factors causing the city to have become one of Thailand's major cities. Considering the distance between Phitsanulok city and other major cities, it has become a hub connecting the principal towns and cities in three regions: Northern, North-eastern and Central Thailand (see Figure 6.1). Moreover, when considering international trades between Thailand, Myanmar, Laos, Vietnam and the South of China, Phitsanulok city is very

well located and likely to become a main international trading centre. This could well occur and if this massive potential is realised, it will certainly affect the future development of the city.

The morphology of Phitsanulok city and its interrelations with surrounding areas is illustrated using a 2002 multi-spectral Landsat satellite image as shown in Figure 6.14. This can be roughly interpreted in the same fashion as we did for the image of Chiang Mai city shown in Figure 6.8 above. Built-up areas are shown in the light orange colour at the centre of the image, and these pixels represent the main development in Phitsanulok city. The River Nan separates Phitsanulok city to its west and east and this is shown in black as indeed are other water bodies such as swamps and the irrigation canal to the west of the river. Dense vegetation areas along Nan River are represented in the dark blue colour. The lighter blue clusters over the surrounding areas are agriculture. Finally, the bright colours of yellow and white represents bare (vacant and uncultivated) land and transportation. According to the image, the dense built-up areas are compactly concentrated on the east side of the river. Beyond this compact core of the city, urban growth can be seen as ribbon development in form. And at some distance from the city centre, the urban area is surrounded by a circular ring road. Unlike Chiang Mai with its three nested ring roads with the first ring very close to the main city, the circular ring road of Phitsanulok city is constructed far from the centre. Thus it appears to have had less influence on the growth of the city, at least so far, but if the city grows, it is bound to become more important as a driver of urban growth.

Due to the fact that much of the topography around the city is composed of floodplains and swamps which are costly and difficult to build upon, this seems to have forced urbanised areas to be built close to existing transportation routes, thus reinforcing the pattern of ribbon development. The image shows that urban areas are growing from the city centre along three major radial roads that link the core of the city to the outskirts in the northwest, the east, and the south. All these roads are then connected together by the circular ring road which lies approximately 15 km from the

city centre. The urbanised area is still expanding along these three directions.

3. Data Exploration

The previous section has overviewed the two case-study cities and allowed us to present some imagery about the importance of both cities from the past to the present. Both cities have played similar important although slightly different roles in Thailand's regional and national development. As these two cities have been chosen as case studies for testing the simulation model, to perform and verify the model, data layers need to be prepared in the same fashion as the data used in the experiments discussed in the last chapter. Unlike these fabricated datasets, these data now need to be collected, managed and organised using various geographical techniques in order to transform and output all data into the model format. The data were collected from several sources both in hardcopy and digital form as 1: 250,000 maps, 1: 50,000 topographic maps, city maps, non-spatial data from the provincial database, and land use/cover from the multi-spectral Landsat satellite images. All map data were manipulated using the ArcView GIS package and the images prepared and interpreted using the Erdas Imagine package.

In this section, we will visualise separately the map data of Chiang Mai and Phitsanulok city which we will put into the form that is ready to use in the model in the next chapter. We will first explain the scope and specific model areas chosen from the entire areas which enable us to produce effective visualisations of the model. Land use/cover changes are the next issue we will present; as we have temporal image data, this section will also discuss matters of interpretation, analysis and comparison of the images which are essential information for validating the model. Finally, the section will summarise all the map data visually in the form in which it will be directly used in the model in the next chapter.

Chiang Mai

The Case Study Field Data

Due to limitations in model computation and the visualisation of its inputs and outputs, it is not possible to picture the entire area of the Chiang Mai province in the model interface. This is due to the fact that there are limits on the size of problem that can be effectively handled due to the number of individual cells required in processing and of course, limits on the size of map that can be easily visualised. The model is meant to study the growth of urbanising areas with their interactions in their surrounding hinterlands. Hence, the Chiang Mai case-study field was restricted to a fixed distance from the existing city centre. For purposes of model visualisation, a trial 12-kilometre radius was fixed from the centre and all administrative territories (sub-districts) that fell within the main city and this hinterland became part of the model area defined by its cellular space. Figure 6.15 depicts the region selected within 12 kilometres from the city centre. The Chiang Mai case-study field covers approximately 864 square kilometres with 58 sub-districts which in terms of Table 5.1 cover the Muang Chiang Mai, Hang Dong, Mae Rim, San Sai, Doi Saket, San Kamphaeng and Saraphi districts.

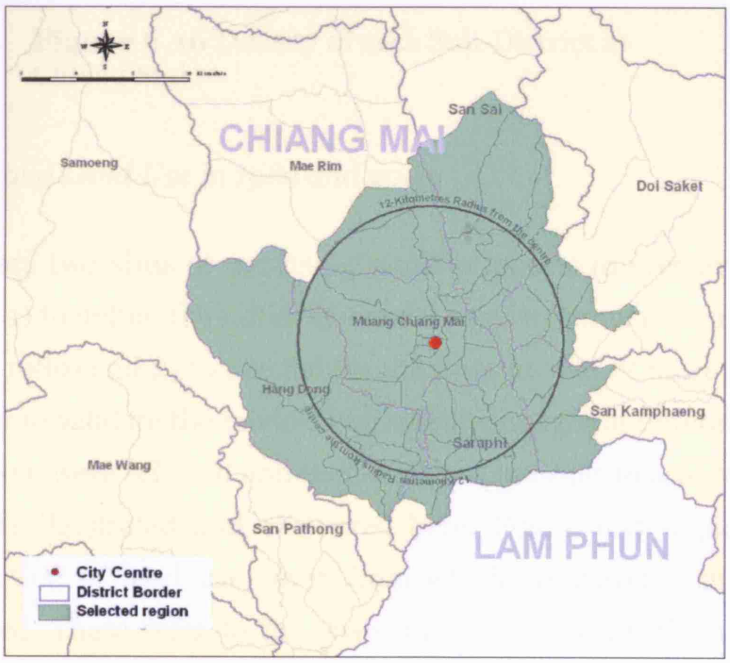


Figure 6.15 Case-Study Field of Chiang Mai city

In the year 2002, the most populated area was the Su Thep sub-district with 37714 persons. While the least populated sub-district was Pa Lan in the east of the study area, the densest areas with over 6000 persons per sq km were Tambon Sri Phum (~7000 ppkm²), Tambon Chang Moi (~6000 ppkm²), Tambon Pra Sing (~7500 ppkm²) and Tambon Hai Ya (~6700 ppkm²), which are all areas in Chiang Mai's city centre (see Figure 6.16 below). The density map clearly shows this variation in densities in the city which reveals the sorts of high demand for land in the centre.

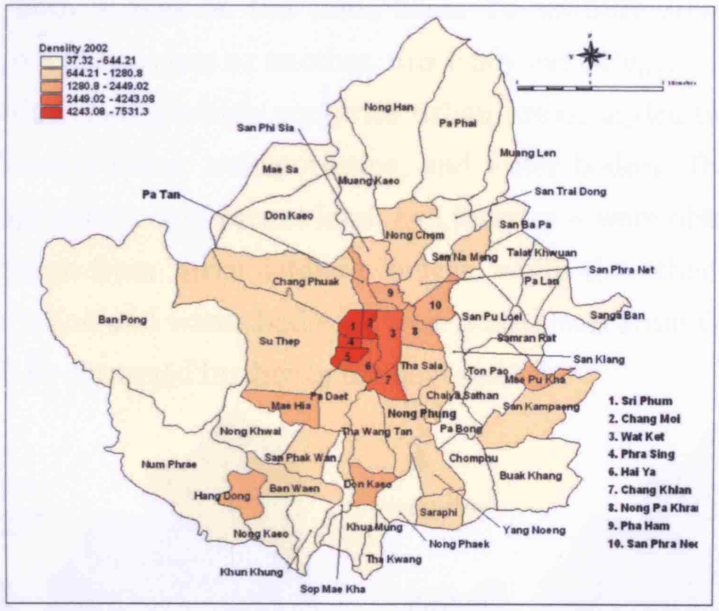


Figure 6.16 Density of each Sub-District in Chiang Mai

Urban and Land Use in 1989 and 2003

There are two aims of our using satellite images to prepare data for this model: 1) to define the initial state of the urban simulation at t = 0 (actual time = 1989) and 2) to use the results from images at more than one date in order to validate the model. Two satellite images at different dates, 1989 and 2003, were selected and then analysed in order to achieve these goals; both are illustrated and compared here. Figure 6.17 depicts the actual distribution of land use/cover from which comparable changes can be extracted. These refer to the years 1989 and 2003 with the 1989 image from Landsat5 TM with a 28.5-metre resolution, and the 2003 image from

a Landsat 7 ETM image with 25-metre resolution. In this case, it is therefore necessary to resample to 28.5-metre to match the 1989-image. Both images are represented with false colour composite with band 7 as the red, band 5 as the green and band 4 as the blue colour. The images obviously show approximately the different size of the Chiang Mai urban area which has clearly grown and broadened into its hinterlands in every direction during the last 15 years.

To detect land use/cover changes during this period, it is necessary to classify both images on the same basis. To simplify areas that are not clearly part of one class or another, this study has categorised the land into 5 definitive classes which comprise urban areas, agriculture and vacant areas, forestry areas, transportation, and water bodies. The first three – urban, agriculture and vacant land, and forestry – were obtained by direct classification from these satellite images, while the other two classes – transportation and water bodies – were added later from GIS data layers. This will be discussed further in the next section.

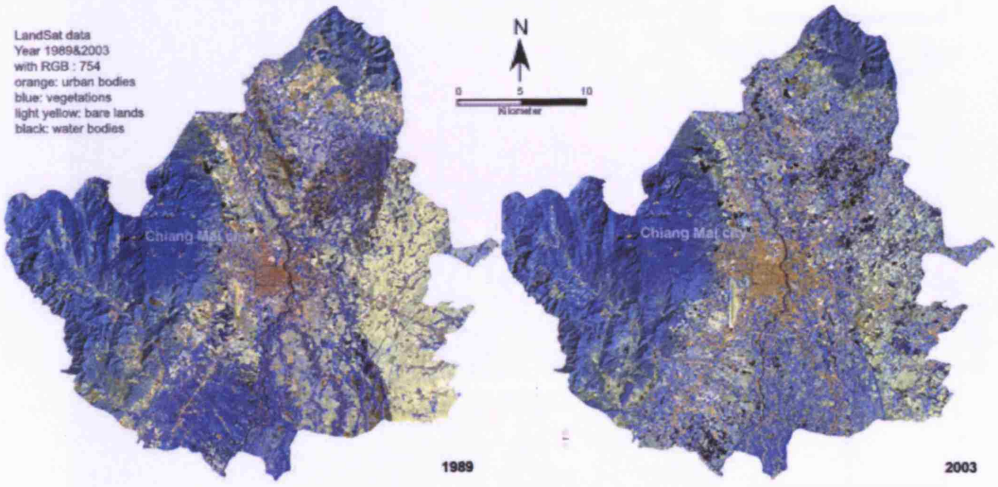


Figure 6.17 Satellite Images covering Chiang Mai Case-Study Area 1989 and 2003

The method used in this study was a supervised classification based on the Maximum Likelihood Classification. To achieve the supervision, a training area (or so called area of interest: AOI or region of interest: ROI) with land use/cover data collected from the field was imposed on the images in order to gather statistical information associated with the image-pixels for each

class. These statistics were then used to evaluate the goodness fit and to enable the process of classification to be developed.

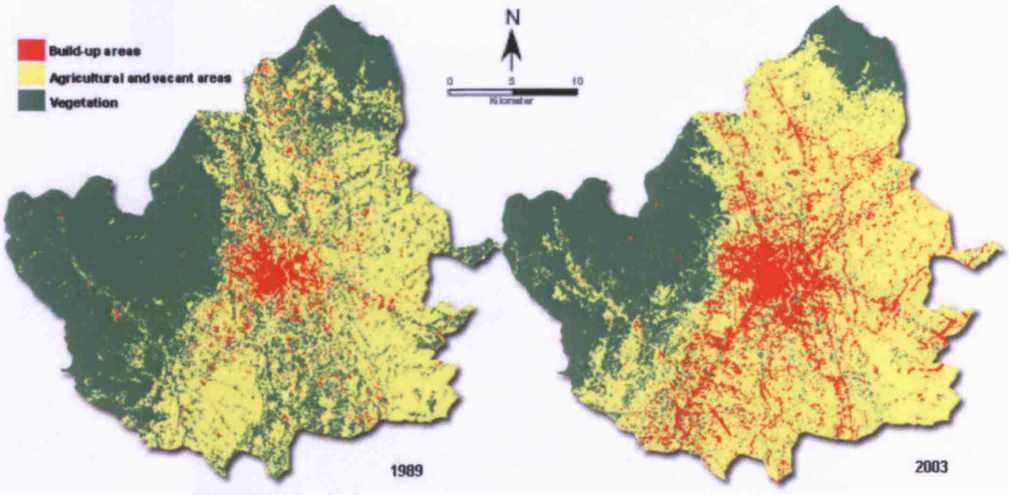


Figure 6.18 Classified Images for the Chiang Mai Case Study Area 1989 and 2003

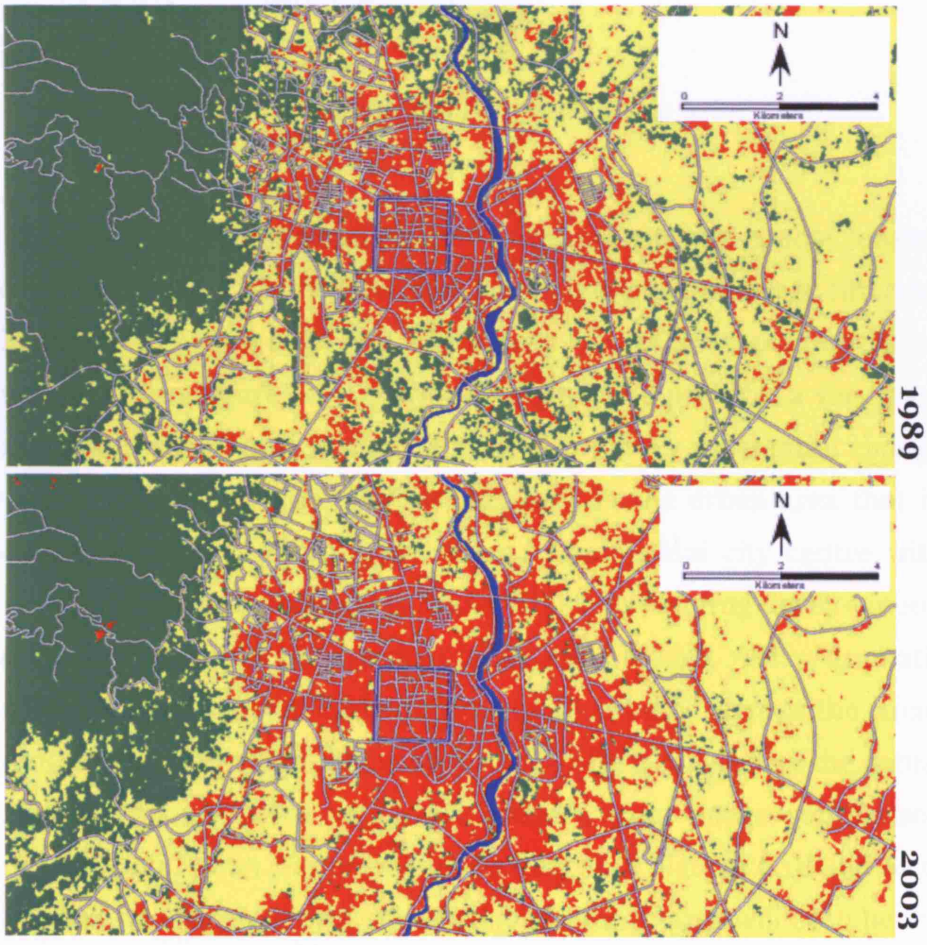


Figure 6.19 Chiang Mai City: 1989 and 2003

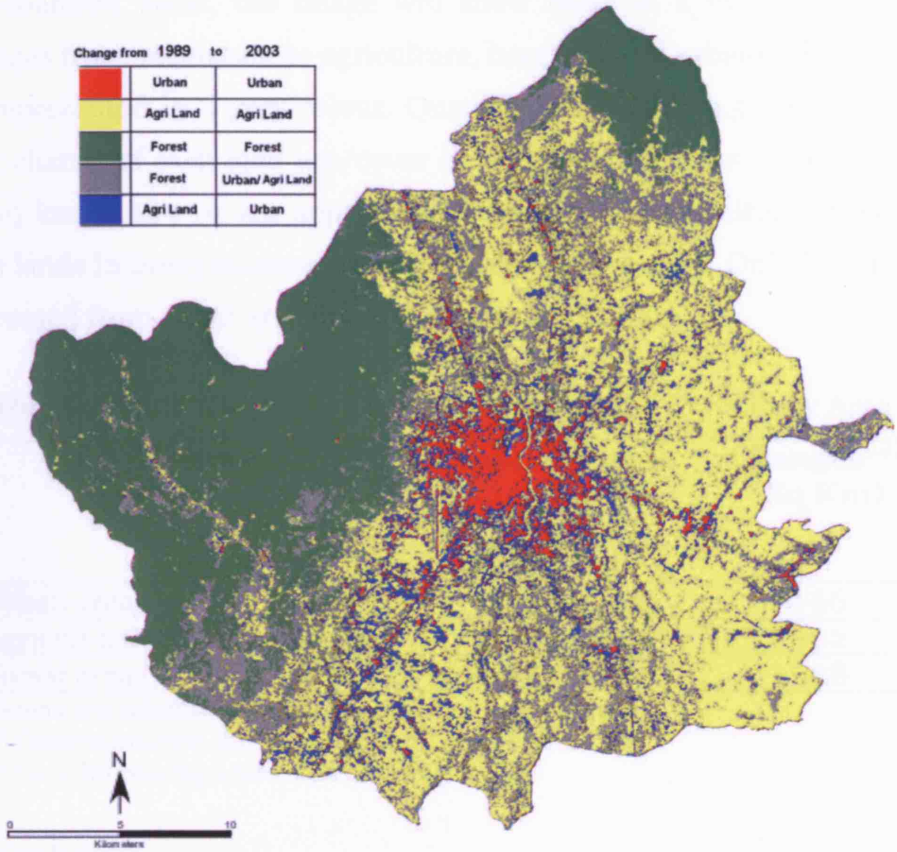


Figure 6.20 Land Use/Cover Changes between 1989 and 2003

Both the classified images which resulted from the various remote sensing procedures of image classification, class combination, and image filtering, are illustrated in Figure 6.18 with a blown-up view of the same images for Chiang Mai city in Figure 6.19. Obviously Figure 6.18 provides a very good picture of land use change in that there is clearly very substantial change between the two dates. The clearest change is in the urban area that in 1989, was compactly concentrated around Chiang Mai city centre with some evidence of areas that had grown as ribbon or leapfrog development. By 2003, the city centre had become larger and broader, with a dramatic expansion of leapfrog and ribbon patterns distributed across the areas surrounding the city centre. Figure 6.20 and Table 6.5 visualise the actual changes in land use/cover which are counted from image comparison between 1989 and 2003. According to the images in Figures 6.18 and 6.19, if there were no change between the two dates, the images will both be the same colour: i.e. identical. On the other hand, if agricultural areas change

to urbanised areas, the image will show these in a blue colour while changes from vegetation to agriculture, bare lands or urbanised areas, will be represented in a grey colour. Quantitatively, Table 6.5 represents the area change of each land use/cover class. Urbanised areas increased from 43 sq km to 109 sq km approximately, while the agricultural areas and bare lands in 2003 increased about 92 sq km from 1989. Only forest areas decreased from about 404 sq km to 246 sq km in 2003.

Table 6.5 Land Use/Cover Changes in Chiang Mai Case-Study Area

Features	Approximate area* (Sq Km)		Changes*** (Sq Km)
	1989	2003	
Urban areas	43	109	+66
Agriculture and vacant lands	417**	509**	+92
Forest areas	404**	246**	-158

*Areas are calculated based on 28.5 meters resolution
** Overestimation due to small number of classes
*** + = Increasing and - = Decreasing

Phitsanulok

The Case-Study Field Data

In the same way, we defined the model space for Chiang Mai city, and Figure 6.21 below represents a 12-kilometre radius drawn from the city centre. Within the circle, we have 23 sub-districts from most parts of the Muang Phitsanulok district, with one sub-district from Phrom Phiram, two sub-districts from Bang Rakam, and another two from Wang Thong. The total area is approximately 1005 square kilometres.

The most populated area was the Nai Muang sub-district with 84,343 persons in 2002, while the least populated area was Chom Thong sub-district in the north of Nai Muang sub-district. The densest area was the Tambon Nai Muang with 4,420 persons per square kilometre which is located in the city centre. The next most dense areas with density ranges from 500 – 1000 persons per sq km were Tambon Ban Khlong (~1000 pp sq km), Wat Chan (~900 pp sq km) and Arunyik (~800 pp sq km) respectively. These areas are neighbouring sub-districts on the east and

west sides of the city centre (see Figure 6.22). From this density map, a fairly useful picture of the urbanisation trends in this area can be extracted.

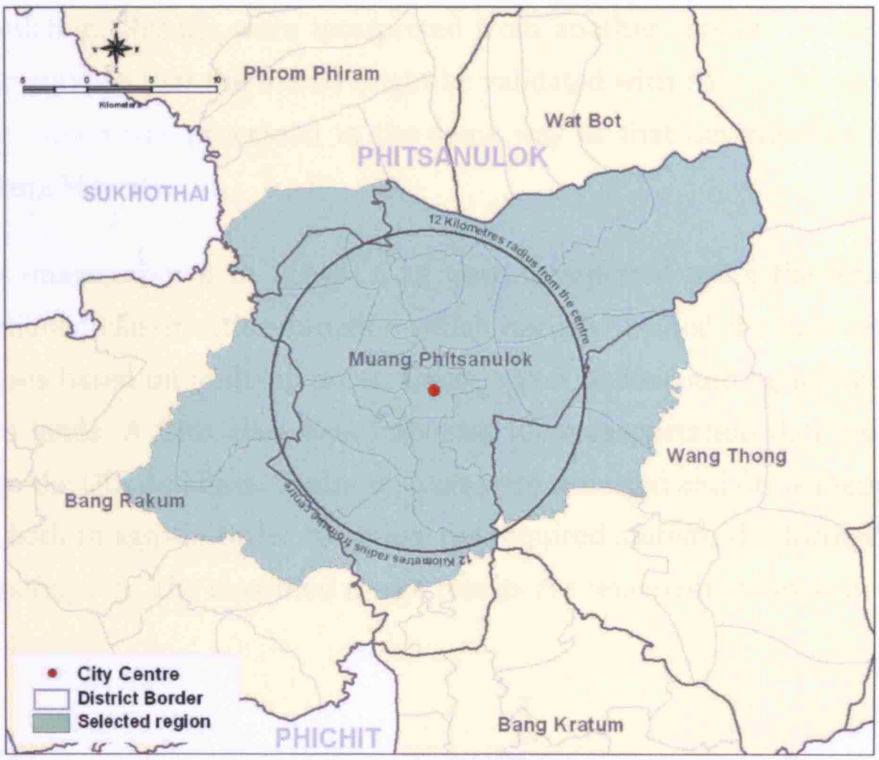


Figure 6.21 Case-Study Field of Phitsanulok

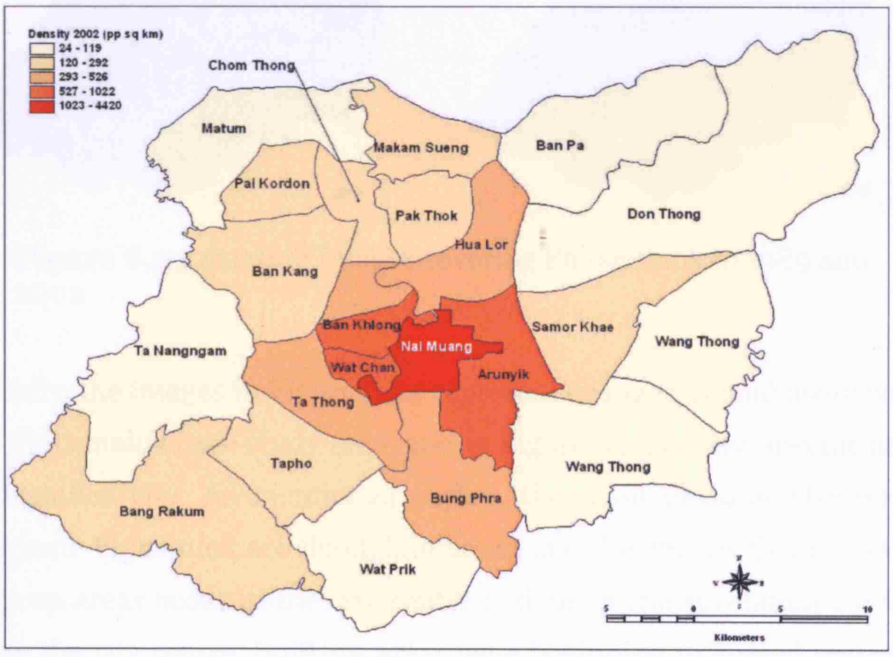


Figure 6.22 Density of each Sub-District in Phitsanulok

Urban and Land Use in 1989 and 2002

A satellite image covering the city area in the year 1989 was selected, analysed and generalised as the initial state for the Phitsanulok urban simulation. Results were interpreted from another satellite image in the year 2002 so that the model might be validated with data at two points in time which was processed in the same way as that described in case of Chiang Mai city.

The images shown in Figure 6.23 were interpreted using the maximum likelihood classification function which initially divided the data into four classes based on built-up areas, forest, water bodies, and agricultural and bare lands. A fifth class was imported for transportation data extracted from the GIS database. Training areas were collected and drawn separately for both images in order to gather the required statistical information for classification. The classified image results are represented below in Figure 6.24.

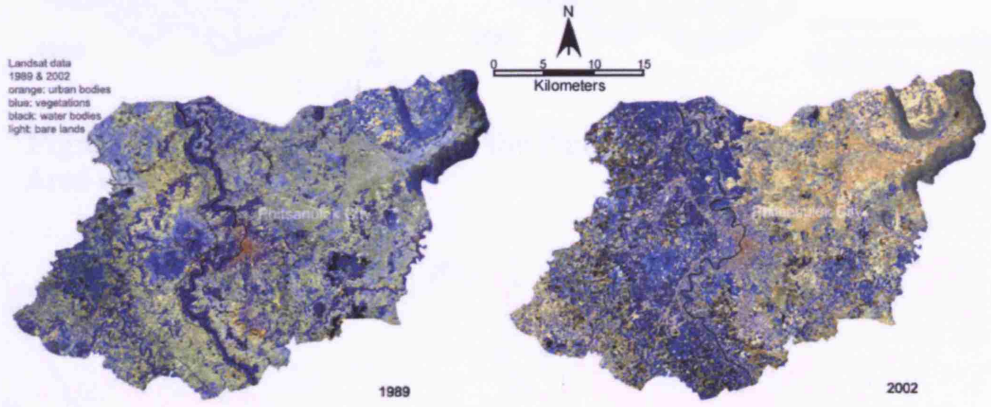


Figure 6.23 Satellite Images covering Phitsanulok in 1989 and 2002

Visually, the images in Figure 6.24 represent changes in land use/cover for the Phitsanulok case-study area, and in Figure 6.25 for the specific area of Phitsanulok city. As mentioned earlier, the main physical features that dominate Phitsnulok are floodplain areas, and this means that most urban built up areas occur in the city centre and along transportation corridors. From the city centre, built-up areas were beginning to spread and sprawl

in all directions from the city as the 1989 image reveals with the ribbon characteristics of this development becoming much clearer in the 2002 image. In addition to the ribbon development pattern that appears at some distance from the city centre, the urbanised areas also broaden out as a comparison between 1989 and 2002 shows in Figure 6.25. Table 6.6 depicts the change in these areas detected from both images. The table summarises approximate overall changes in these areas; built-up areas increase from 53 sq km to 116 sq km; there is a little change from agricultural areas and bare lands from 809 sq km to 827 sq km; and forest land decreases from 143 sq km to 62 sq km.

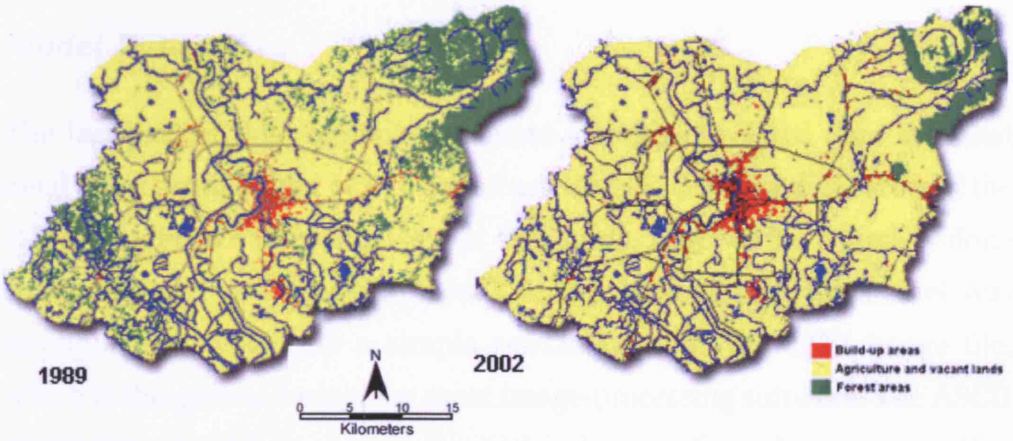


Figure 6.24 Classified Images in the Phitsanulok Case-Study Area 1989 and 2002

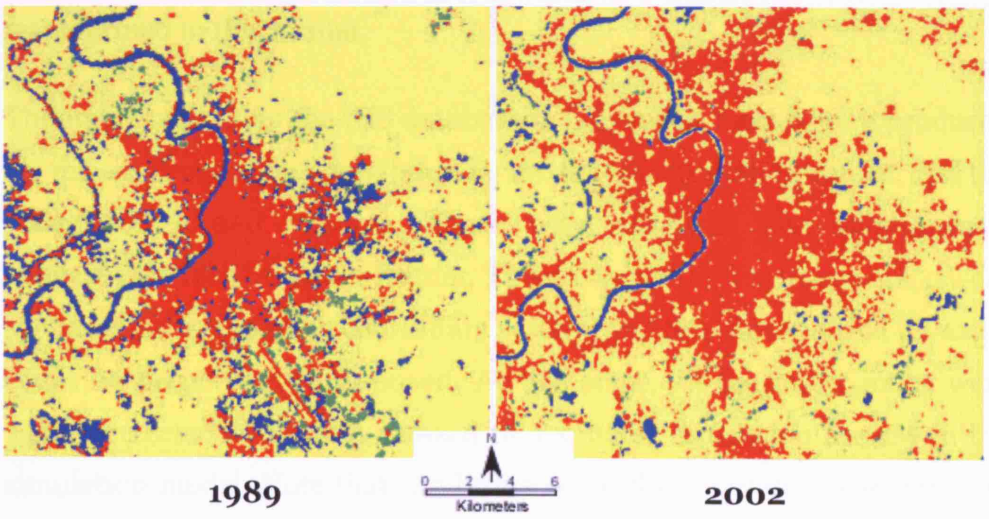


Figure 6.25 Phitsanulok City: 1989 and 2002

Table 6.6 Land Use/Cover Changes in Phitsanulok Case-Study Area

Features	Approximate area* (Sq Km)		Changes*** (Sq Km)
	1989	2002	
Urban areas	53	116	+63
Agriculture and vacant lands	809**	827**	+18
Forest areas	143**	62**	-81

*Areas are calculated based on 28.5 meters resolution
** Overestimation due to small number of classes
*** + = Increasing and - = Decreasing

Model Datasets

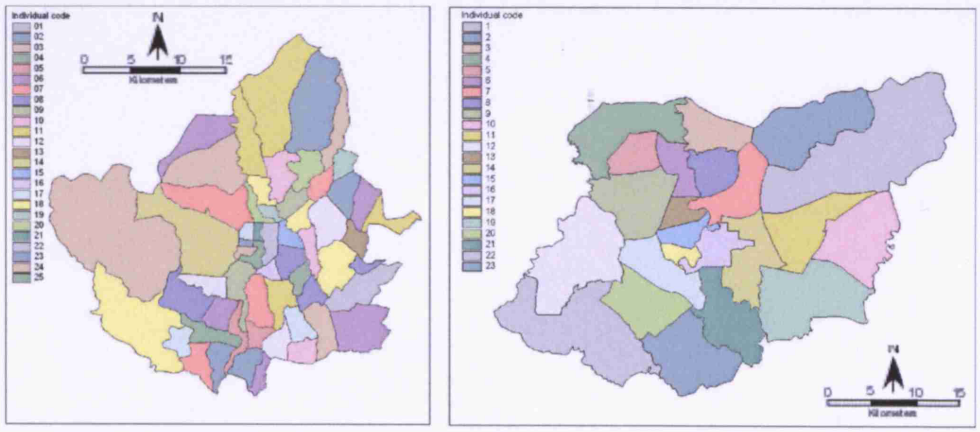
The last part of this chapter introduces a series of related data sets that need to be manipulated before each case study is developed in terms of the DSSM model. We will visualise this data in the ways we have already done so in this and previous chapters. The input module of the model was developed to allow only a simple readable format in ASCII image file, which is the general format for most image-processing software. The ASCII image format is similar to a 2-dimensional array of numbers representing all values of an image (see the example of such an ASCII file format in Appendix II). All images mentioned in the previous sections are thus transformed to this format.

Thematic maps from the GIS database in GIS vector format were produced for the administrative areas/boundaries, the transportation routes, and the water bodies and streams. These were classified and subsequently converted to the GIS raster format. The vector and raster image maps were prepared using the same coordinate system, and this ensured that all maps could be properly superimposed. At this stage, all the raster maps were again corrected and finally resized to a suitable dimension for use in the simulation model. Note that the limits set in the computer program were that any application should not be greater than 122,500 pixels or 350x350 pixels. All Chiang Mai data were thus resized to 340x322 pixels and for Phitsanulok data, this was set at 411x282 pixels (about 120 metres: 4 times

less than the resolution of the original data). The slope layer was constructed from raster DEM data which in turn is created from the contour data of 1: 50,000 maps. Based on 3D GIS functions, the slope layer was directly created, and it was masked and resized to the same scale as the converted GIS raster maps. The last two data layers required are the urban seed and land use/cover. The urban seed layer was constructed by extracting only the urban cells from the classified 1989 image illustrated in the last section. Most of features in the land use/cover layer were obtained from the classified image. Two classes - transportation and the rivers – were added to the land use/cover data from the GIS database. Finally all the data were converted to the ASCII image format.

We show visualisations of all data layers for both cities below, and these are those that are directly input for validation and application of the model in the next chapter. The layers include rivers and water bodies, transportation, land use/cover, administrative area and their boundaries, population in the year 1989, slopes, and urban areas in the year 1989. Finally we show the urban seed sites which are used to drive the simulations. These seeds are those that we consider areas that will develop at the start of the simulation period.

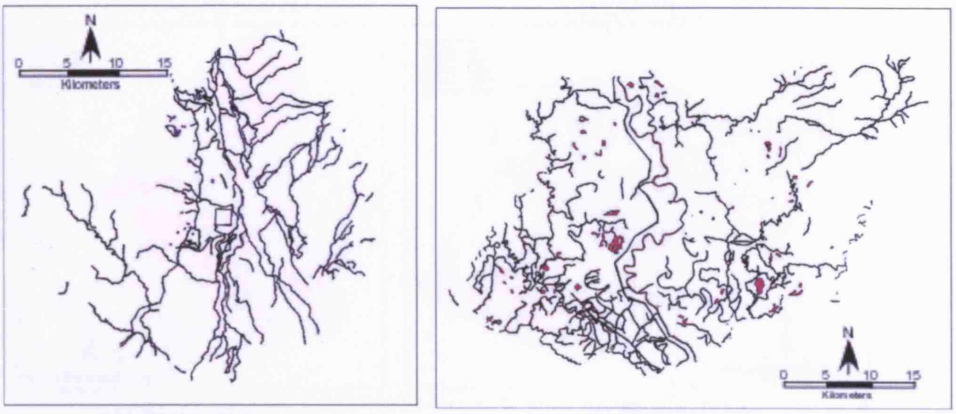
Administrative Areas and Boundaries



a) Chiang Mai b) Phitsanulok

Figure 6.26 Administrative Boundary Maps of the Case Studies

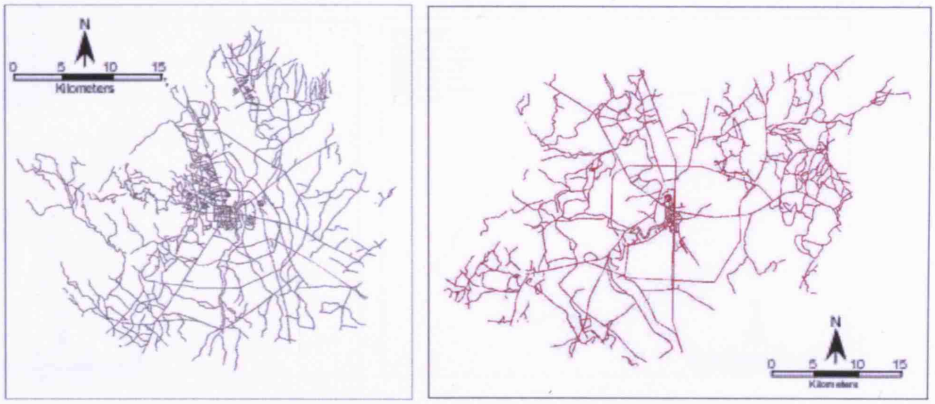
Rivers and Water Bodies



a) Chiang Mai b) Phitsanulok

Figure 6.27 River Maps of the Case Studies

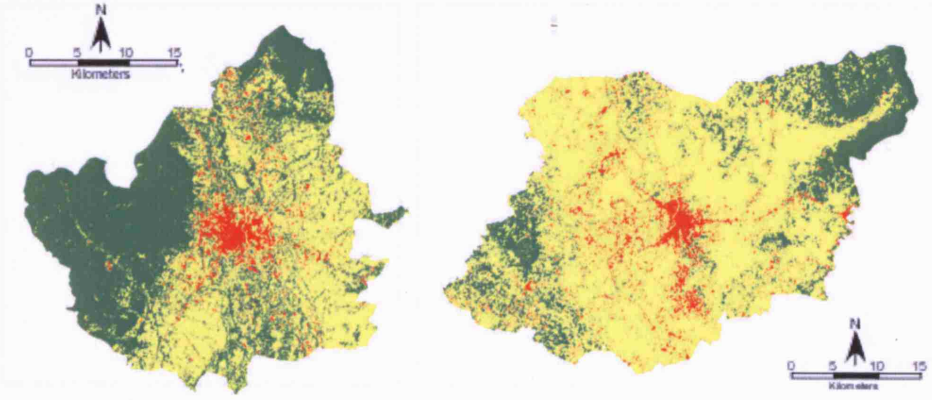
Transportation



a) Chiang Mai b) Phitsanulok

Figure 6.28 Transportation Maps of the Case Studies

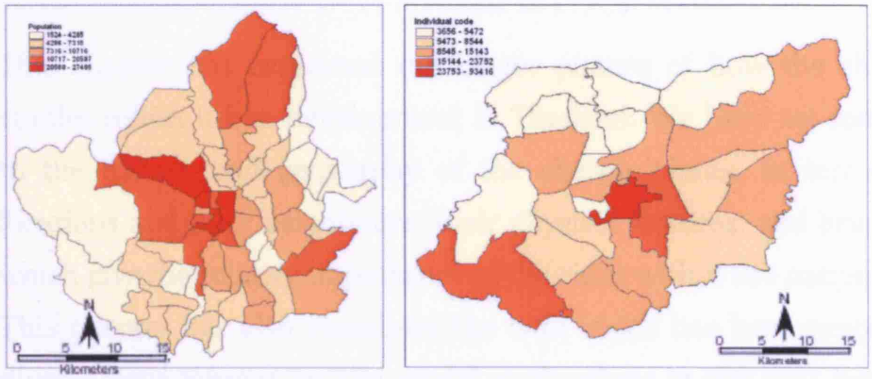
Land Uses



a) Chiang Mai b) Phitsanulok

Figure 6.29 Land Use/Cover Maps of the Case Studies

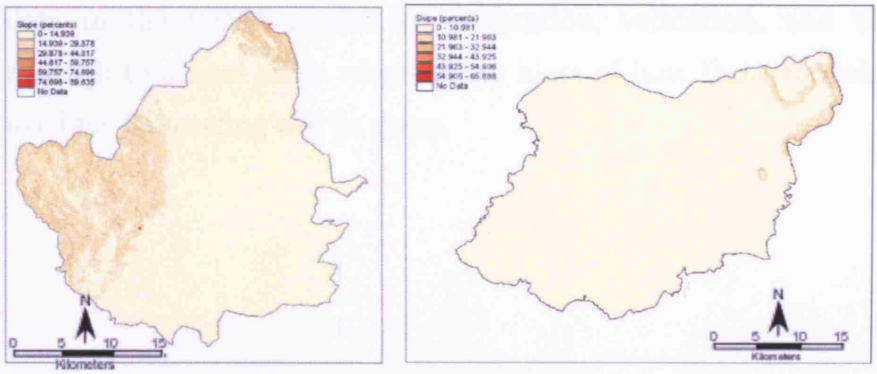
Population



a) Chiang Mai b) Phitsanulok

Figure 6.30 1989-Population Maps of the Case Studies

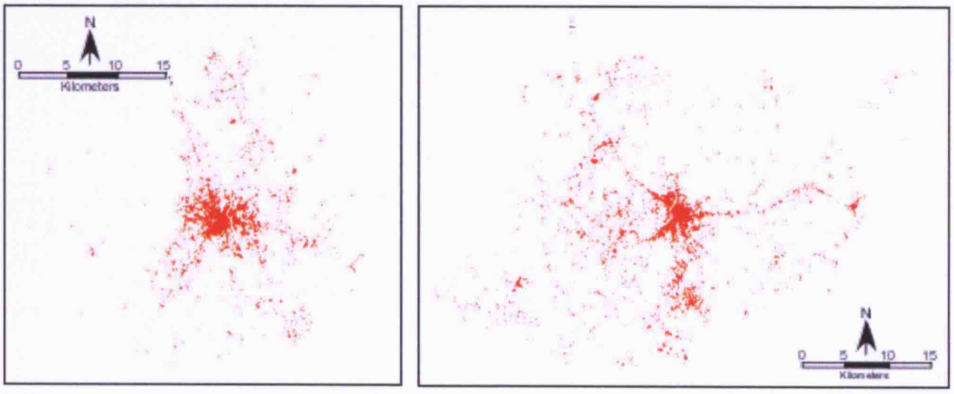
Slope



a) Chiang Mai b) Phitsanulok

Figure 6.31 Slope Maps of the Case Studies

Urban Seeds



a) Chiang Mai b) Phitsanulok

Figure 6.32 Urban Seeds of the Case Studies

4. Conclusions

This chapter has explained the bigger picture of how the chosen case studies reflect urban development in Thailand. We have set some context to the history and geography of the chosen places, in terms of their locations and their neighbours, their physical features, and brief histories which give the relative importance of the cities within the national picture. This chapter has also visualised the data which has been prepared from diverse data sources for the model applications in the next two chapters and various methods for transforming these data into a suitable form for the model have been explained. All this will be useful in understanding and interpreting the results associated with the simulations at the next stage. The next chapter will continue these descriptions through the use of this data in the DSSM model for calibration, validation, and verification, through to a final stage where some hints of how these models might be used for forecasting will be given.

CHAPTER SEVEN
City Case Studies: Evaluations and Applications

1. Introduction

We have already established the general context for these case studies in the last chapter and this one will continue the series of experiments with the model, first introduced in Chapter 5, but now with the real data that was illustrated in the last section of Chapter 6. This chapter is organised into two main sections which deal with the application of the model to the two case study cities. We refer to the first stage as model evaluation in that this is the stage in the model building process that deals with calibration. We then deal with model applications which involve some tentative experiments in using the two models for the two cities in forecasting over medium term periods of 20 or so years.

So far the model has been tested on arbitrary trial city spaces with arbitrary data and the focus has been on reporting and illustrating answers to two primary questions: how the various algorithms within the model work and the extent to which the hypothetical examples are good mirrors of what is happening in terms of urban development in the real world (see the experiments at both the coarse and detailed scales in Chapter 5). The first section of this chapter will attempt to focus on the latter question. Unlike experiments with the test data, the input and use of various spatial features in real cities is much more complicated with respect to the workings of the model. Our experiments at this stage are designed to use the spatial features from 1989 as the initial state for performing these simulations but as we will see, many parameters also have to be set to ensure that the initial conditions are as close as possible to what we observe in reality.

Two other issues need to be considered involving model parameters and the simulation time. As we noted in Chapter 5, the model is composed of a

very large number of model parameters which all are important in driving the simulation. However, it is not possible to test all possible combinations of values of these parameters in testing the model of the real city, so a limited but plausible set of parameter values will be used and reported here. Time, on the other hand, is another important parameter which influences the simulation procedures. As demonstrated in Chapter 5, changes in the simulation parameters generate different growth rates for urban cells in the CA space. Hence, a part of this chapter will discuss the measurable period of time over which urban growth can be observed and modelled in reality, and this leads quite naturally to issues about how we need to transform the model iterations into the temporal parameters used in the simulations.

As described in Chapter 6, the multi-temporal classified images for both case study areas are essential to this stage. We are clearly fixed in terms of the real times we have available and a period of some 13 years (1989 to 2002/2003) for both cities seems reasonable enough to give us the kind of observable changes that we are able to model. We therefore set the simulation times at which the model would output data to the unit times implied from this data – thus providing outputs at every year, although the various internal parameters (iterations) of the simulation were set at a finer interval so that the diffusion and CA processes could operate as we illustrated in Chapters 4 and 5. The model thus returns a sequence of data and their images synchronously with real world time, and these images are used to make comparisons with the observed data so that the performance of the model can be evaluated.

We used two approaches in order to evaluate the results of our experiments in Chapter 5 based on visual comparisons and simple quantitative measurements, and we will employ the same strategy here. For visual comparison, the results will be set against the classified image maps. In case of quantitative comparisons, we will introduce a number of statistical methods to evaluate the model including the accuracy assessment used in image processing, correlation statistics, and regression analysis. With these techniques, we can easily measure how accurate the

model is, and then suggest how the model can be used to simulate other city development processes. In general, this first stage of model evaluation through calibration sets the stage for use of this model in other predictive contexts including its generalisation to the generation of 'what-if' scenarios central to a variety of decision and planning support systems.

This chapter will also demonstrate two other sets of experiments with the model based on the real data sets by establishing various growth scenarios that the model is able to test. First, a simple scenario based on the prediction of urban growth and land use/cover change for Chiang Mai city over 20 years from 2003 was run, based on the parameters established during the model evaluation stage. Another experiment was then set up using the 'what-if' component of a spatial decision support system for the Phitsanulok case study. Connecting the model to such a system is one of most important steps in getting these kinds of techniques closer to the real users of spatial information technology in urban planning. In this chapter, a feasible 'what-if' scenario was designed and the simulations then executed for visualising and describing future outcomes in an analytical fashion. Our use of the model in this context is very much something that relates to its future use and we have not focussed this thesis at all on the actual use in planning although the model has been developed in sympathy with the kind of technological support that the urban planning process requires. We will pick up on this issue in our conclusions but it is enough to say that we consider DSSM to be entirely consistent with the general class of CA models that have been suggested as tools in decision and planning support (Geertman and Stillwell 2003).

2. Model Evaluation and Calibration

In this section, we will begin to evaluate the model in detail through the data we have defined for the two case study areas which we structured into a form suitable for the model in the last chapter. Prior to starting the simulations and engaging in analysis that will lead to an evaluation of the model, the simulation parameters and the time variables will be

reconsidered for the purposes of defining the scope and range of sets of model parameters by their values while the simulation iterations will be matched up to the real time structure in the data. After we have defined these parameters and time iterations, we are in a position to run the model to produce outcomes, which we will then compare visually as we have done in previous chapters. At the same time, we will use their raw data from the various outcomes for measurement of the fit of the model in a statistical fashion.

Defining Parameters

Table 5.1 in Chapter 5 clearly shows that there are a large number of parameters to be defined in terms of their values prior to performing any simulation. These model parameters can be classified into 3 groups: 1) parameters for the CA simulation such as thresholds for CA transition rules and diffusion; 2) parameters for the constrained map dynamics which affect the construction of the dynamic maps at each time state or period; and 3) parameters for the hierarchical node and potential surface map generation such as distance measures and functions which are needed to create nodes and fashion the measurement of potential. As seen in Chapter 5, there were various possibilities for combining these parameters in order to get different results (outcomes) from the simulation model. The first parameter group associated with the CA simulation is not only important for driving the simulation, but also intimately relates to the time parameter. Hence, changing parameters within this group directly affects the simulation, real time matching, and growth rates produced by the model. The second and third parameter groups both concern the generation of the constraining maps which influence the CA simulation and bind the cellular allocation processes to the socio-economic dynamics.

Block 7.1 represents a set of CA transition rules defined initially for this experiment and this is consistent with the CA rules tested in Chapter 5. The rule set will now be used as the default for every simulation in the first calibration experiment. Furthermore, some of these parameters were set as constant as we note in Table 7.1; as a result, they are completely

associated with the simulation time that is initially needed to harmonise the time periods defined by the real data.

Table 7.1 also provides the major parameter values for the simulations used in the experiments for both case studies. For these experiments, four scenarios were delineated, setting different values of these parameters in order to evaluate differences in outcome. As seen in Table 7.1, these experiments were implemented so that different results could be associated with different contexts. The outcome associated with the first scenario was defined as an archetype used for comparison with all the others. The second scenario was used to test changes in the relative density that is the major parameter used in constructing the population density map. The third experiment deals with the transformation of the distance function that affects the generation of centrifugal surface map. And in the last experiment, the results from the archetypal example from the first experiment were compared to outcomes from the simulation based on different weight scores associated with the constraining map layers. Using the same parameters for both case study areas also gave us another way to compare the differences between Chiang Mai and Phitsanulok, differences between the cities in terms of the standard model, in contrast to differences in the parameterisation of the models fitted to the same cities.

Once all parameters are identified, growth rates of the urban cells in both case studies could then be estimated. This is then matched against the real growth of the urban areas using multi-date satellite imagery and its interpretation. The next section will explain this in more detail.

Table 7.1 Defining Parameter Values and Rule Thresholds for the Calibration Experiments

Model parameters		**Values and Thresholds for Scenario:**				
		Time	**1**	**2**	**3**	**4**
Raster Map	Numbers of row Numbers of column Resolution	Chiang Mai: 322 and Phitsanulok: 283 Chiang Mai: 340 and Phitsanulok: 412 120 m.				
Accessibility	Search radius for accessibility	300 m.				
Constrained CA thresholds	From dead cell to developing state	0.6				

Model parameters		Values and Thresholds for Scenario:				
		Time	**1**	**2**	**3**	**4**
	From developing state to developed cell	0.6				
Diffusion	Threshold for generate diffused cells on CA space	0.2				
	Maximum number of diffused seed cells	5				
	Range for diffusion period	0-5				
Forest areas change	Threshold-- if majority of neighbouring cells is agriculture	0.1				
	Threshold-- if majority of neighbouring cells is urban	0.3				
Relative density	Urban	**0.8**	0.8	**0.4**	0.8	0.8
	Agriculture	**0.15**	0.15	**0.4**	0.15	0.15
	Forest	**0.05**	0.05	**0.2**	0.05	0.05
Population growth rate	Random ranged from 0 to	0.3				
Node and Hierarchy	Distance to generate node	5,000 m.				
	Distance for creating centrifugal surface	**25 km.**	25 km.	25 km.	**10 km.**	25 km.
Weight scores with transportation	Centripetal surface	**1**	0.5	0.5	0.5	**2**
	Centrifugal surface	**2**	2	2	2	**1**
	Accessibility	**5**	3	3	3	**3**
	Topography	**2**	2	2	2	**2**
	Land use	**0**	0	0	0	**0**
	Land demand	**0**	2.5	2.5	2.5	**2**
Weight scores without transportation	Centripetal surface	**0**	1	1	1	**2**
	Centrifugal surface	**0**	2	2	2	**1**
	Accessibility	**0**	0	0	0	**0**
	Topography	**8**	3	3	3	**2**
	Land use	**0**	1	1	1	**1**
	Land demand	**2**	3	3	3	**4**
Land demand with urban cells existing	Weight for urban	0.8				
	Weight for agriculture	0.2				
Land demand with no urban cells existing	Weight for agriculture	0.6				
	Weight for forest	0.1				

From dead cells $$u_{t+1} = \begin{cases} 0 : u_t = 0; \omega_1 = 0 \\ 1 : u_t = 0; \omega_1 \geq 2; \text{random} \end{cases}$$

Developing cells $$u_{t+1} = \begin{cases} 1+ : u_t = 1; \omega_2 \geq 5; \text{random} \\ 1- : u_t = 1; \omega_2 < 5; \text{random} \\ 10 : u_t = 1; \omega_2 \geq 5; t - t_u \geq p; \text{random} \end{cases}$$

Developed cells $$u_{t+1} = \begin{cases} 1- : u_t = 1; \omega_3 \leq 4; \text{random} \\ 10 : \text{otherwise} \end{cases}$$

t: time state
u: urban cells
ω: neighbourhood function
state 0: dead cells
state 1: initial state of developing cells
state 1+: level-up developing cells
state 1-: level-down developing cells
state 10: developed cells

Block 7.1 General CA Transition Rules for the Experiments

Defining Time: Simulation and Reality

One of the general problems of all spatio-temporal models which involve iteration in the allocation of activities to space is to measure or to match the simulation time with the reality. In particular, the time scale of the simulation is set by the 'iterations' which are associated with the set of program algorithms that will be executed in one cycle with all the changes sequentially appearing after the cycle has been completed. The key question involves how many days or months or years one iteration represents, and this will define a cycle. On the other hand, another way of looking at this issue is how many simulation periods correspond to a day or a month or a year, and this of course will define an iteration.

To resolve these definitional issues of temporality, it is necessary to find out what the relationships are between these two temporal scales. To do so, this section firstly shows the quantitative measurement of image results

classified from the time date satellite images thus obtaining the change in urban areas between 1989 and 2003 for Chiang Mai city. The changes in the urban area evaluated from these images is then used to set a parameter threshold which measures each cycle in the simulation.

As mentioned in the previous section, a set of parameter values concerning the cellular automata processes was initially defined in order to generate a rate and pattern of growth for the simulation which was consistent with that observed. At this stage, the simulations were executed using controls on weighting scores associated with the constraining maps as shown in the scenario 'Time' in Table 7.1 above. New cells generated by the model were counted until this number reached the amount of transformed urban cells measured from the previous time state; the total number of simulation time states was then reported and then it was approximately matched to the real time in question. This was simply a way of apportioning urban growth equally within the time periods associated with the simulation but matching the difference between the urban imagery at 1989 and 2003. For instance, if the total change in urban cells from 1989 to 2003 is, say, 1000 cells, and the total time from running a simulation until reaching 1000 cells is 200 iterations, then this means that we assume 200 iterations to be 14 years—with, in this case, about 14 iterations corresponding to one year in reality (that is, 14 iterations x 14 years = 196 iterations ~ 200 iterations). In this experiment, after defining the parameters in the model as well as defining the total amount of cell changes between the two dates, because the simulation is based on random processes, we ran the model ten-times and then took the mean number of iterations so that we could get some estimate of how many iterations needed to be associated with a typical or average run of the model. This of course is still very much an approximation but as far as we aware, this is the first time in any cellular automata modelling of this kind, this kind of inquiry into the choice of time parameters has been made. It is of course occasioned by the fact that in this model, we used the simulation itself to generate total growth and change which still needs to be constrained in some way to what we observe.

Table 7.2 Changes in the Urban Areas of Chiang Mai and Phitsanulok from 1989 to 2003/2002

Urban Areas	Approximate area (Sq Km)		Changes (Sq Km)	No. of pixels at Resolution 120 m.
	1989	2003/2002*		
Chiang Mai	58	108	+50	3472
Phitsanulok	47	73	+26	1805

* 2003 for Chiang Mai and 2002 for Phitsanulok

As can be seen in Table 6.5 in the previous chapter, Table 7.2 again summarises the change in urban areas for both case studies. The data used in the model was prepared using the 120 metre resolution pixel grid; based on this pixel size, the amount of pixels shown in the last column of the Table 7.2 is computed. Tables 7.3 and 7.4 represent results of 10 trials of the model in terms of the total number of iteration cycles processed from the simulation, where each simulation was executed under the parameter sets shown in Table 7.1. The means of these iterations for each case study were then computed as shown in Tables 7.3 and 7.4.

For Chiang Mai area, the 10 simulations produced 137 iterations as the mean which corresponds approximately to 10 iterations corresponding to one year in real time. On the other hand, the mean of the iterations for the Phitsanulok trials was 52. One year in reality is then estimated as 4 iterations of simulation time for this city. These two values are then used as the time parameter for all scenarios that we have developed for this model in terms of each case study in the rest of this chapter.

Table 7.3 Simulation (Iteration) Times for Chiang Mai City 1989-2003

Trial	No. of Cells	Period (years)	Iterations
1	3480	14	135
2	3484	14	140
3	3481	14	141
4	3485	14	130
5	3490	14	132
6	3471	14	141
7	3493	14	146
8	3482	14	137
9	3494	14	138
10	3490	14	133
Average			**137**

Table 7.4 Simulation (Iteration) Times for Phitsanulok City 1989-2002

Trial	No. of Cells	Period (years)	Iterations
1	1810	13	53
2	1802	13	52
3	1805	13	54
4	1815	13	52
5	1801	13	50
6	1811	13	53
7	1806	13	52
8	1809	13	48
9	1814	13	56
10	1801	13	53
Average			**52**

Simulating Chiang Mai 1989-2003

To implement a real case simulation for evaluating the model, it is also necessary to dynamically reassign some map data layers illustrated in the previous chapter to obtainable appropriate temporal datasets. For instance, existing roads were, in reality, constructed in different time periods; defining existing roads as they evolved through the simulation process would therefore provide much more valid results than using only a single static transportation map fixed for the beginning of the simulation, at 1989 in these cases. Furthermore, we will clearly use the actual population records which illustrate far better the population growth characteristics than random growth rates used in the theoretical experiments with the model in Chapter 5.

In case of Chiang Mai city, the existing road layers were attributed into 3 classes for the years 1989, 2000 and 2002. In terms of the growth rates, instead of randomisation in each simulation period, these experiments apply actual 1989-2003 differences in population derived from demographic data from 1993-2002. In short, we associate population change then with changes in the pixels which show urban growth and in this way match up the two different sets of data. Note that for 1989-1992 and for 2003, population is backcast and then forecast for these missing data using a regression equation. Its form is not important here but it is the most consistent way of manufacturing a proper population time series.

According to the time parameters calculated in the previous section, the model was then refined to input sequentially both transportation and population data into the simulation processes at appropriate time points. That is, the population map is updated every 10th simulation iteration which is the equivalent of one year while at the 100th and 130th iterations, the road features which are attributed to the years 2000 and 2002 are redefined respectively.

1. Scenario 1: Base Scenario

Figure 7.1 demonstrates the sequence of images that represent the predicted growth of urbanised areas in Chiang Mai city and its surrounding region from 1989 to 2003 using the temporal structure defined at the first stage. It is clear that urbanised areas tend to expand from the centre of the city to the east and south based on the probabilities calculated from the dynamic map. As mentioned earlier, the weight scores defined in Table 7.1 significantly affect the probability of generating cells in the space. In this scenario, the most important factor was accessibility which was weighted as the highest score relative to others as seen in Table 7.1.

As seen in Figure 7.1, it is clear that the small urbanised area to the south becomes denser since 1994 when compared to the 1989 image. Whereas small clusters to the north of the city were still less dense, some have become denser after 1998 which then begin to expand. While the leapfrogging of development has become the dominant pattern occurring around the far periphery of the city, the centre also grew into its surrounding areas, especially to the east and south of the city. For example in the east, cells were first generated close to the first circular road called 'superhighway' (see development between 1991-1994) and vacant lands between the superhighway and the city have been infilled since the year 1996.

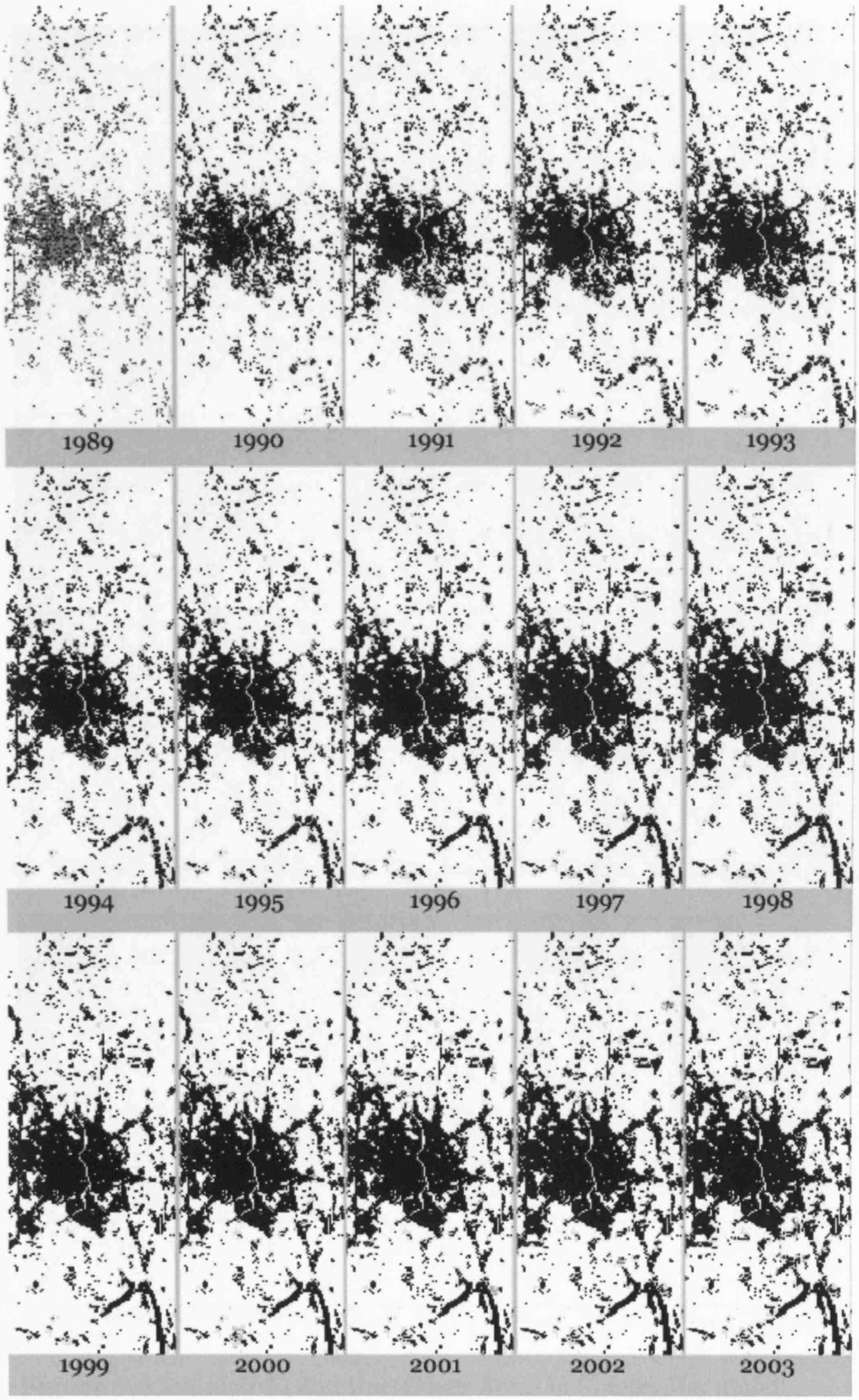


Figure 7.1 Urbanised Areas of Chiang Mai 1989-2003 predicted under Scenario 1

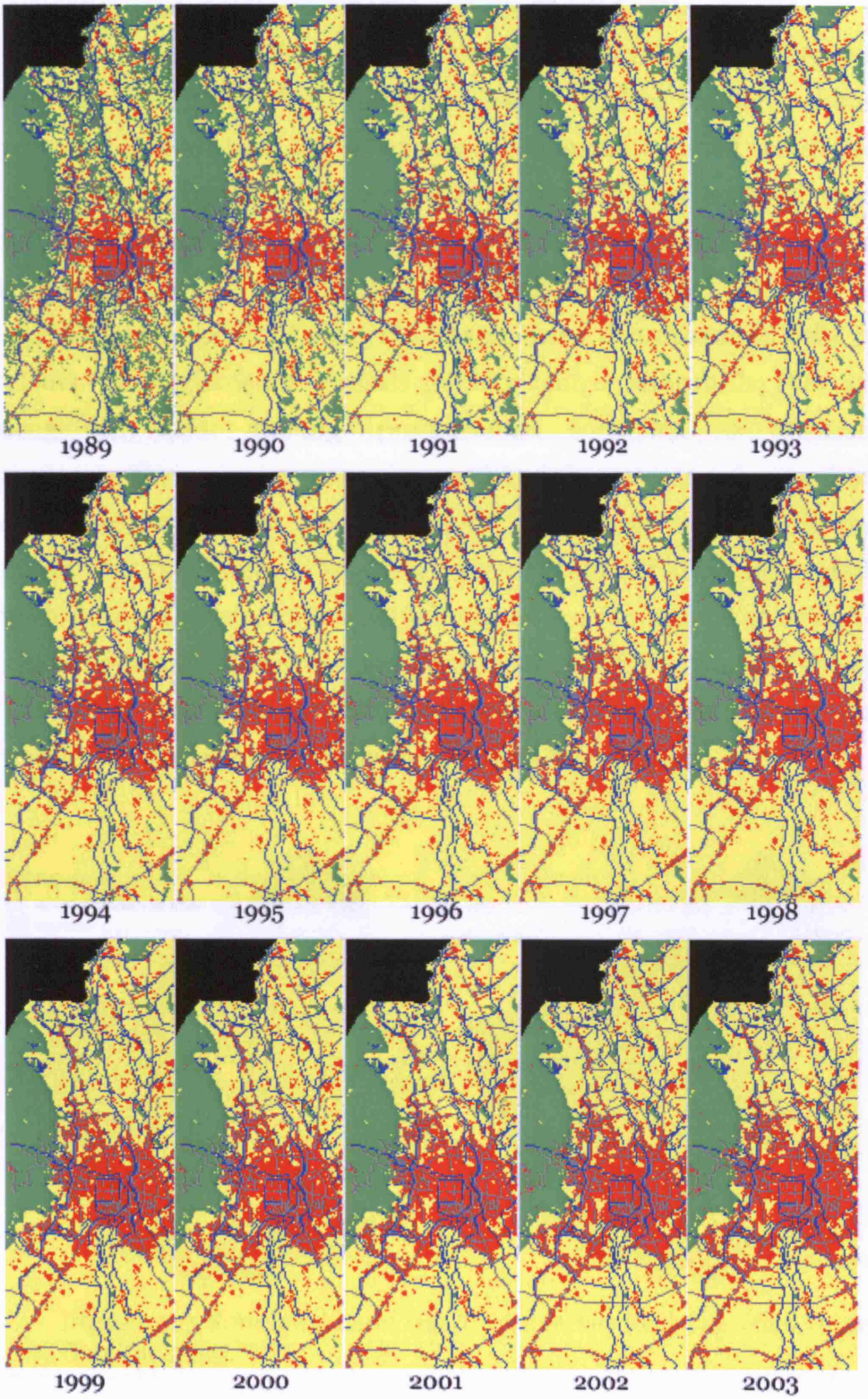


Figure 7.2 Simulated Land Use/Cover Areas in Chiang Mai under Scenario 1

Figure 7.2 shows complementary outcomes generated from the simulation. Sequential land use/cover images are constructed from changes in the urbanised cells within the CA space from transition rules that apply to the land use layer itself. These images represent gradual temporal changes from one class of land use to another. For instance, small forest areas (small groups of green cells) on the plain to the north and south of the city in 1989 gradually decline and transform to agriculture and vacant lands. Most of these areas have become completely transformed since the year 1994. From this point of view, the images show that urbanised cells (red colour) first agglomerate towards the city centre. Afterwards, they grow along the major existing transportation corridors. As mentioned previously, the second circular ring road and the outer circular road have only been usable since 2000 and 2002 respectively as shown in these images. For this reason, these two ring roads do not affect urbanisation much during this simulation period and only a few groups of urbanised cells were generated near the second circular road to the south of the city. However, it is possible that both will have an influence on the city when running the model into the near and medium term future.

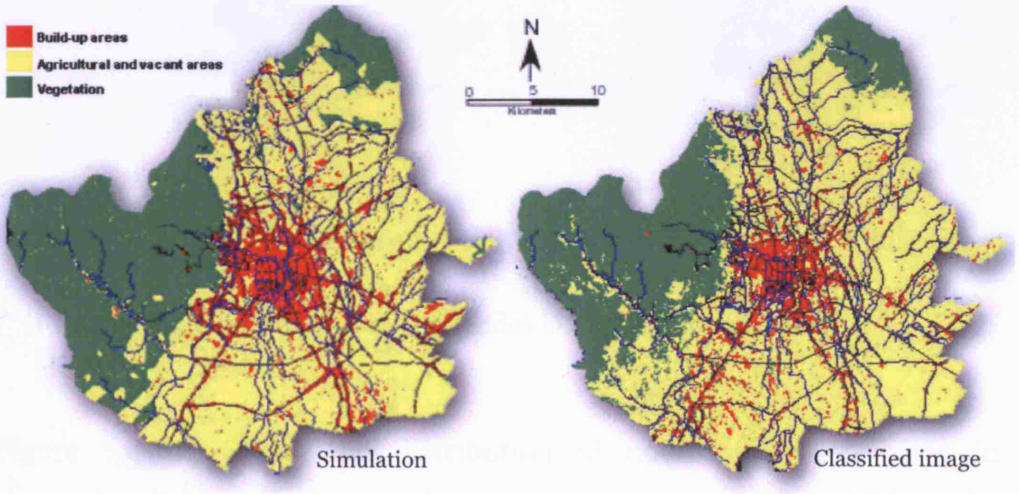


Figure 7.3 Simulated Land Use/Cover versus Real Classified Image Data for 2003

It is important to compare these simulation results to the real world data in validating and calibrating the model. Figures 7.3, 7.4 and 7.5 visualise the results of land use/cover, node distributions and potential surfaces for

the simulation at iteration 140 (year 2003) compared to the same spatial characteristics created from the actual data in year 2003.

The land use/cover maps were classified into 5 types based on urbanised areas (red), agriculture or vacant land (yellow), forest area (green), road cells (black), and water bodies (blue). After 140 iterations of the simulation, two large forest areas remained in the north and the west of the city, which are close to the forest features classified from the satellite image in 2003 (see the green areas in Figure 7.3). The urbanised areas, on the other hand, were distributed around the city centre and developed along the main transportation routes in all directions especially through to the south and the east of the city. From the actual data, the image clearly shows that the urbanised area has developed to the south along Chiang Mai-Hang Dong road and this is consistent with the model.

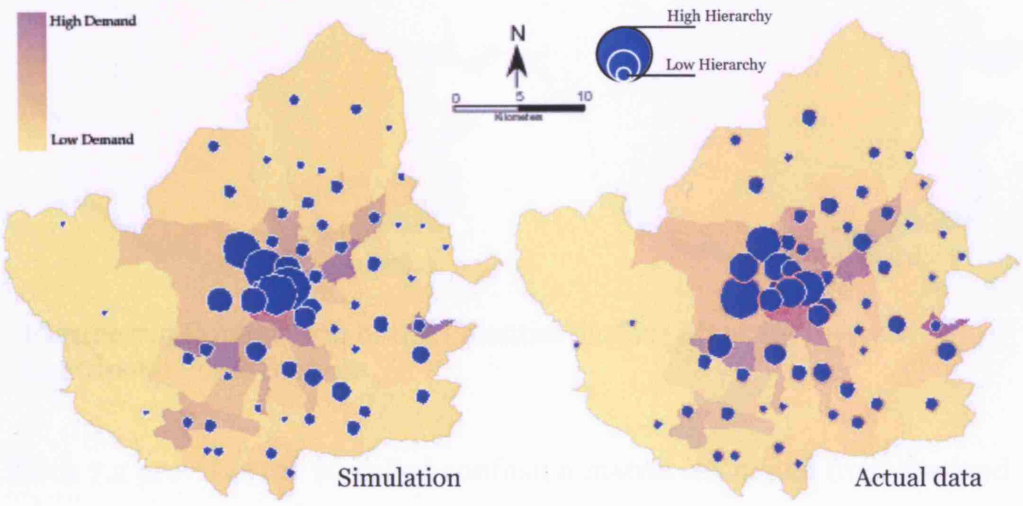


Figure 7.4 Comparison of the Nodal Distribution and Land Demand

Figure 7.4 represents the distribution of nodes and their sizes in comparing the simulation results with the actual data. In addition, the background images illustrate the intensity of land demand in each sub-district. As seen in this figure, the node distribution from the simulation looks pretty similar to the nodes generated from the real dataset especially in the vicinity of the city centre and the area in the south. Considering the

land demand map, each sub-district also reflects a similarity of spatial pattern between the simulation results and the actual data, and this gives further confidence that we have chosen reasonable parameter values and that the structure of the model reflects more or less what is happening in the real world.

The two potential surface maps created from the simulation and the actual data are shown in Figure 7.5. Both images indicate the concentration of high potential in the same areas: around the city centre and to the west of the city, again confirming the quality of this simulation.

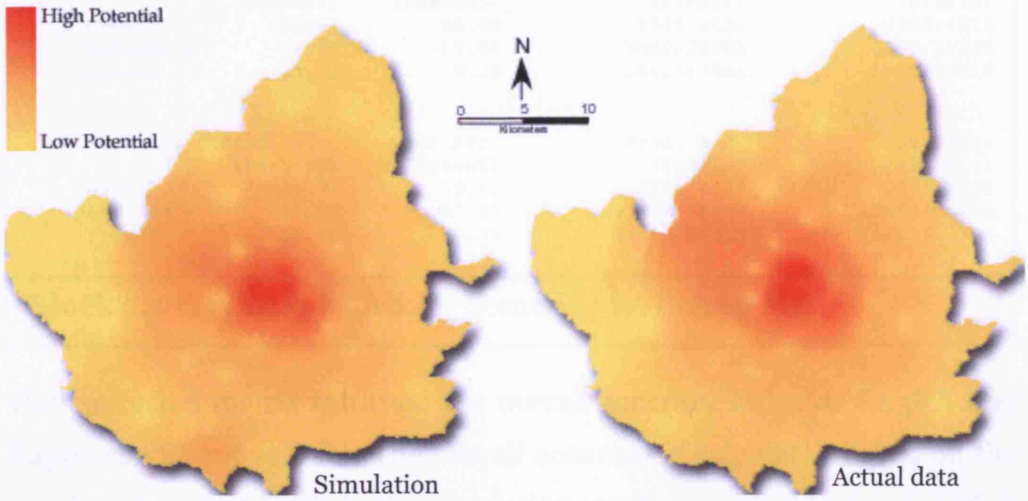


Figure 7.5 Comparison of the Potential Surface Maps for the Centripetal Force Surface

Block 7.2 provides the so-called confusion matrix calculated from the land use/cover image created from the model as an input image and the actual image classified from the satellite image as the ground truth image. First, the two indexes - overall accuracy and Kappa coefficient - are computed in order to summarise the accuracy of the outcome from the simulation compared to actual data. Other parts depict in detail the various elements of these quantitative indexes from the confusion matrix which can indicate actual accuracies for each land use class.

Confusion Matrix: [Chiang Mai] (340x322x1)

Overall Accuracy = (42578/52319) 81.3815%
Kappa Coefficient = 0.6630

Ground Truth (Pixels)

Class	Urban	Agri&Bare	Forest	Total
Urban	2276	3338	206	5820
Agri&Bare	2212	25492	1294	28998
Forest	27	2664	14810	17501
Total	4515	31494	16310	52319

Ground Truth (Percent)

Class	Urban	Agri&Bare	Forest	Total
Urban	50.41	10.60	1.26	11.12
Agri&Bare	48.99	80.94	7.93	55.43
Forest	0.60	8.46	90.80	33.45
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	60.89	49.59	3544/5820	2239/4515
Agri&Bare	12.09	19.06	3506/28998	6002/31494
Forest	15.38	9.20	2691/17501	1500/16310

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	50.41	39.11	2276/4515	2276/5820
Agri&Bare	80.94	87.91	25492/31494	25492/28998
Forest	90.80	84.62	14810/16310	14810/17501

Block 7.2 Confusion Matrix for Scenario 1 for Chiang Mai

The confusion matrix returned the overall accuracy index as 81.38% and Kappa coefficient as 0.66. The overall accuracy is calculated based on the number of correct pixels in the simulation result compared to the classified image with respect to the total number of pixels in these images (see Equation 7.1). This means that when considering all classes in the image, 81.38% of the cells in the simulation have been predicted correctly. On the other hand, the Kappa coefficient is an index indicating the classification accuracy using information from the confusion matrix. The Kappa coefficient is a ratio representing difference between observed and expected sets of data. It can be computed based on the total number of pixels in all the ground truth classes, the sum of the confusion matrix diagonals and the sum of pixels of all classes from both the simulation image and the ground truth image as shown in Equation 7.2. (see more detail of the confusion matrix in Jensen 1996). In this case, with a Kappa coefficient of 0.66, this means that there is a 66% probability that the result from the simulation matches the classes in the actual land use map.

$$OAc = \frac{\sum_{i=1}^{n} P_{ii}}{N_c} \qquad (7.1)$$

where
OAc : Overall accuracy
P : Matched pixels
N : The total number of pixel in the matrix
i : row in confusion matrix

$$K = \frac{N\sum_{i=1}^{t} P_{ii} - \sum_{i=1}^{t} P_{i+}P_{j+}}{N^2 - \sum_{i=1}^{t} P_{i+}P_{j+}} \qquad (7.2)$$

where
K : Kappa index
P: Pixel
t : the number of rows in the error matrix
N : the total number of training data in the error matrix
i : row in confusion matrix
j : column in confusion matrix
+: column or row sum

The confusion matrix also provides the detailed accuracy for each class as shown in the next section of the matrix in Block 7.2. The first two matrices represent the quantities counted as pixels and computed in percentage terms. The last two matrices provide other detailed indexes including errors of commission and omission, and the producer and user accuracy. These portray other important information about each class in rather different fashion.

Consider the ground truth column in the first and second matrices. For the urban class, the ground truth indicates that there are 4,515 pixels in this class. The simulation was able to generate 2,276 pixels or 50.41 percent of these pixels properly but 2,212 pixels or 48.99 percent were classified as agriculture or vacant land with only 27 cells or 0.6 percent forest area. This means that the simulation model yields approximately 51 percent accuracy for the urban class, 81 percent for agriculture or vacant areas, and 91 percent accuracy for the forest class.

Errors of commission represent pixels that belong to another class that are labelled as belonging to the class of interest. As seen in Block 7.2, the urban class has a total of 5,820 pixels where 2,276 pixels are simulated correctly while 3,544 other pixels, the residuals, are classified incorrectly as urban. Then the error of commission can be calculated as a ratio between incorrect pixels and total number of pixels in the ground truth. For the urban class, the error of commission was 3544/5820 or 61%. The errors of commission in the agriculture and forest classes were 12% and 15% respectively. In contrast, errors of omission represent pixels that belong to the ground truth class but which the model has failed to generate in their proper class. The urban class has a total of 4,515 pixels where 2,276 pixels are generated correctly while 2,239 ground truth pixels are classified incorrectly. Therefore the error of omission is 2,239/4,515 or 50%.

The producer accuracy is a measure indicating the probability that the model has generated an image pixel in Class A, say, given that the ground truth is also in Class A. The urban class has a total of 4,515 ground truth pixels where 2,276 pixels were generated correctly. The producer accuracy is thus the ratio 2,276/4,515 or 51%, whilst the producer accuracies of agriculture and forest classes are 81% and 91% respectively. Lastly, the user accuracy is another measure indicating the probability that a pixel is in Class A given that the model has generated the pixel in Class A. The model has generated a total of 5,820 pixels as the urban class of which a total of 2,276 pixels are classified correctly. The user accuracy of the urban class is thus the ratio 2,276/5,820 or 39%. We now examine the three other experiments that make variations on this benchline scenario.

2. Changes in the Dasymetric Parameters

The second experiment was set up by defining the relative density in the dasymetric mapping module in a different way. In this case, the relative density for urban and agriculture/vacant land use classes were set at 0.4 which means that 40 percent of all population in each sub-district lives in

urbanised areas and 40 percent in the agricultural-rural areas. Another 20 percent of population will live in the forest areas.

In the same fashion, the simulation generates the following outcomes which are visualised below. These results can also be used in calculating and constructing the confusion matrix to assess the accuracy as we show in Block 7.3.

Confusion Matrix: [Memory1] (340x322x1)

Overall Accuracy = (43303/52319) 82.7673%
Kappa Coefficient = 0.6797

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	1905	2225	120	4250
Agri&Bare	2583	26507	1299	30389
Forest	27	2762	14891	17680
Total	4515	31494	16310	52319

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	42.19	7.06	0.74	8.12
Agri&Bare	57.21	84.17	7.96	58.08
Forest	0.60	8.77	91.30	33.79
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	55.18	57.81	2345/4250	2610/4515
Agri&Bare	12.77	15.83	3882/30389	4987/31494
Forest	15.77	8.70	2789/17680	1419/16310

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	42.19	44.82	1905/4515	1905/4250
Agri&Bare	84.17	87.23	26507/31494	26507/30389
Forest	91.30	84.23	14891/16310	14891/17680

Block 7.3 Confusion Matrix for Scenario 2 for Chiang Mai

Figure 7.6 illustrates the series of images generated from this scenario for an east-west slice across the study area. As can be seen in these images, changes in the parameters generating the dasymetric maps over the entire time period affects some of the constraint maps through the dynamic map module as we show for the slices in Figure 7.7. When the relative density of urban areas is reduced from 0.80 to 0.40, this decreases the concentration of population in the city areas and means that the agriculture/vacant land use areas are allocated much more population for these are weighted the same as urban.

Because this scenario was set up by changing the dasymetric map parameters, it generates rather different dasymetric maps from the first scenarios. Figure 7.7 represents this population density map comparison between the first two scenarios, which obviously illustrates these differences. With equal relative densities of urban and agriculture in Scenario 2, the population does not concentrate exclusively in urbanised areas but is also located in agricultural areas. Consequently this influences the land demand which is also different; the growth of urbanised cells then differs from the first scenario for the constrained probability calculations are affected by these differences.

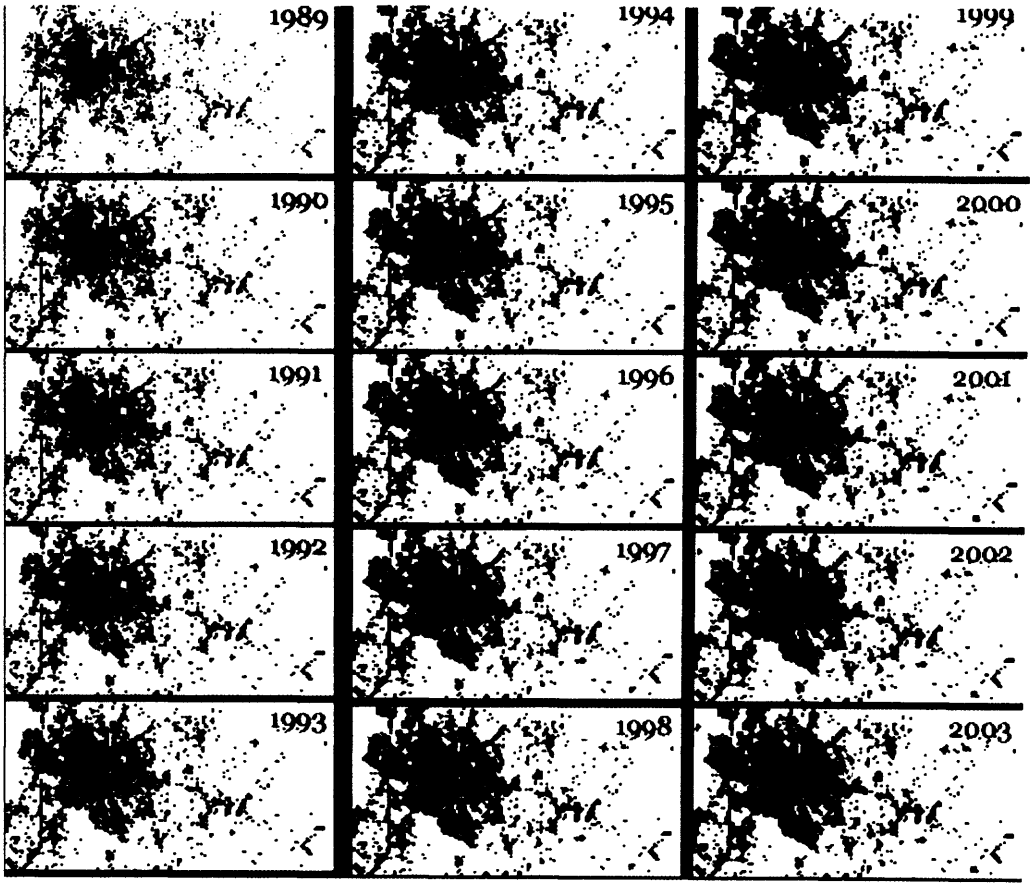


Figure 7.6 Urbanised Areas of Chiang Mai 1989-2003 from Scenario 2

Urbanised cells tend to agglomerate in areas surrounding the city centre. Only a few are distributed at the far side but close to transportation features. A small-urbanised cluster to the east of the city centre begins to grow up after 1995, and until 2003, this growth pattern is similar to the actual data shown in Figure 7.8. Since the year 2000, the images clearly

show a trend linking the urbanised area of the main city with this small cluster to the east.

The overall accuracy for this scenario was approximately 82.76% and the Kappa coefficient was 0.68. These represent the basic degrees of error in the overall land use/cover generated by this simulation. Although these summary indexes show acceptable degrees of accuracy, they are not sufficient to summarise how good or bad the model is, and it is still necessary to consider the other detailed indexes as we did in the previous scenario.

For each class, Block 7.3 also gives these other detailed indexes. In this case, the simulation was able to generate 1,905 cells from the total of 4,515--or only 42.19% -- for the urban class. Whereas when we compare this with the ground truth, 2,583 cells or 57.21% were classified as agriculture or vacant areas, and 27 cells or 0.6% as forest areas. From the first two matrices, we see that there is an accuracy of 42.19% for urbanised areas, 84.17% for agriculture or vacant lands, and 91.3% for forest areas.

The ratio between incorrect pixels and total number of pixels—or the error of commission was approximately 55% for the urban class, 13% for agriculture, and 16% for the forest class. This shows that there are 55% of pixels from other classes labelled as belonging to the urban class. On the other hand, the error of omission for each class were 58% for the urban class, 16% for agriculture, and 9% for the forest class. 58% of the error of omission means that there are 58% of the total pixels that belong to the ground truth but which the model has failed to generate in the urban class.

The producer accuracy in the last matrix gives 42 percent. This means that if there are 100 urban pixels in the ground truth, 42 pixels are generated correctly from the simulation model, whereas the user accuracy for urban class was approximately 45 percent. Thus if the model generated 100 pixels as urban cells, then 45 urban pixels would be correct.

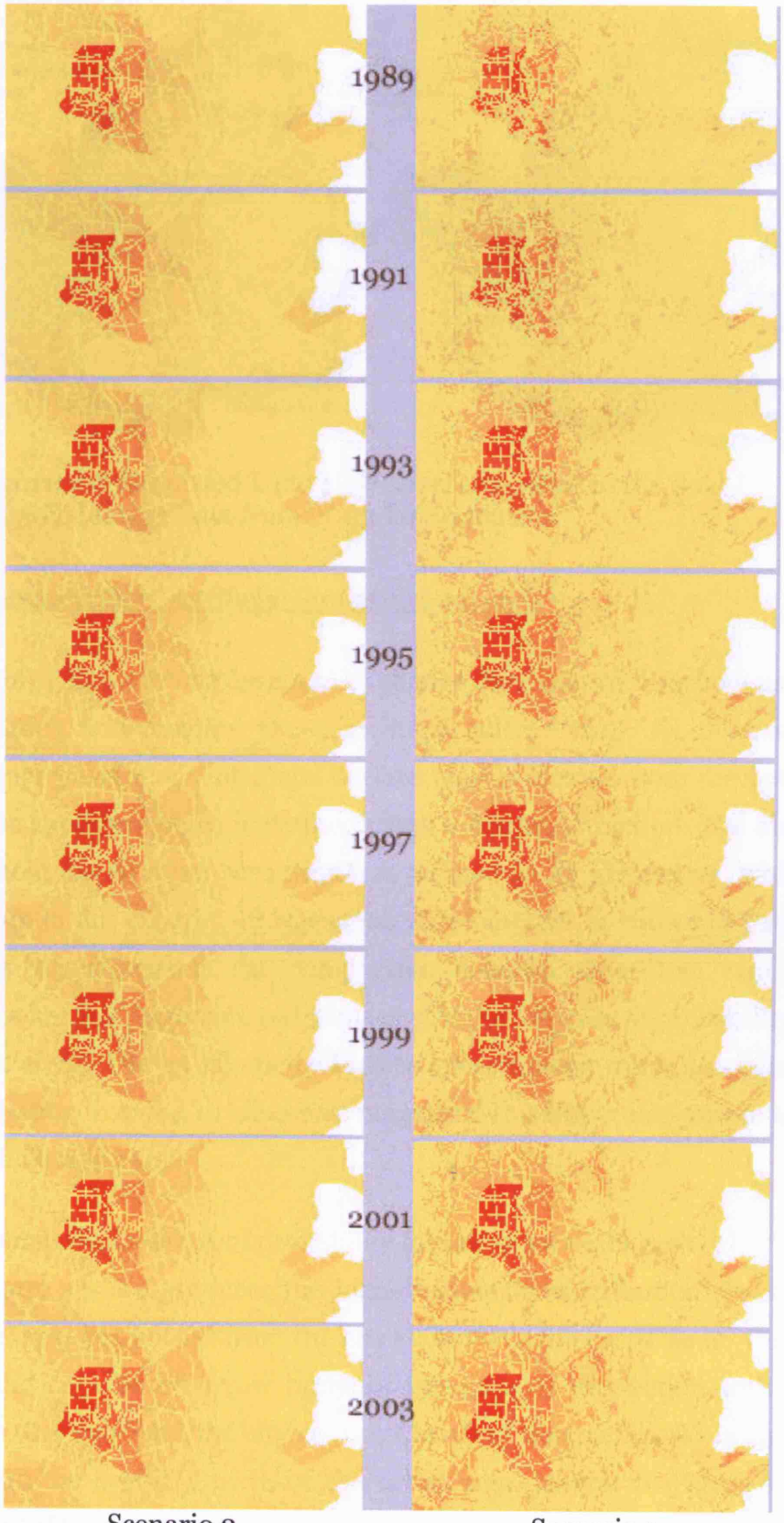


Figure 7.7 Comparison of the Dasymetric Maps between the Two First Scenarios

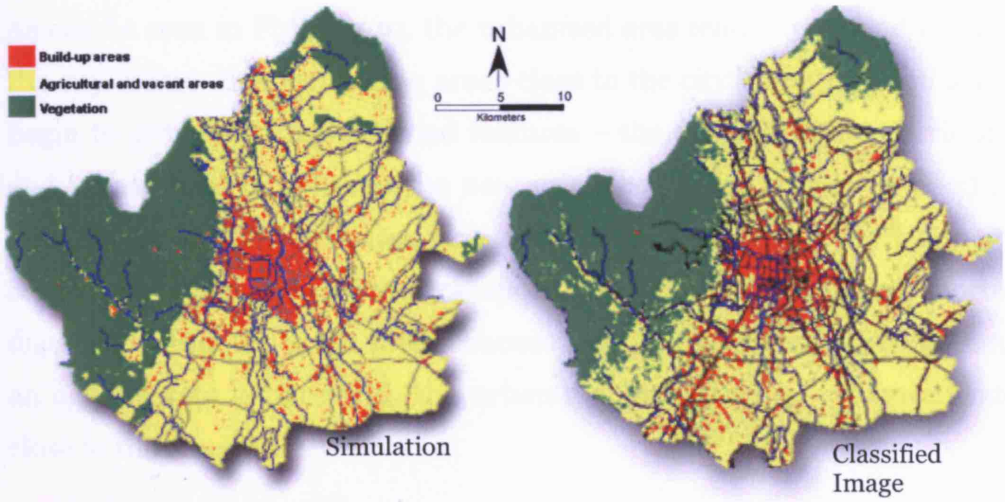


Figure 7.8 Simulated Land Use/Cover compared to the Real Classified Image Data from 2003 for Scenario 2

3. Changes in the Centrifugal Surface Parameters

This third scenario was designed to change the distance threshold used for calculating and creating the centrifugal surface map. As discussed in Chapters 3 and 4, a centrifugal surface map is created from the potential surface approach which is defined using a distance function. The distance threshold for all other scenarios was defined as 25 kilometres, while the distance in this experiment was set as 10 kilometres as shown in Table 7.1. Figure 7.9 represents the comparison between these two centrifugal surface images. As shown, red areas give higher degrees of probability than yellow areas. The 25-kilometre surface gives a much more homogeneous distribution than the 10 kilometre range, and this uniformity covers almost the whole study area.

Intensities depict levels of probability for urbanisation in each cell. The 10-kilometre surface produces the highest intensity distribution around the hinterlands but not far from the city centre. We thus assume in this case that the city has a narrow range of influence or interaction with other towns that surround the city centre. Changing this parameter in this way leads to the overall growth of the urban area. Images in Figure 7.10 as shown below represent the sequence of urbanised areas obtained from the simulation.

As can be seen in Figure 7.10, the urbanised area tends to expand around the city centre. In surrounding areas close to the city, the urbanised areas begin to grow up along the road features – the transportation corridors, and vacant land in between the new urbanised areas is then in-filled in later stages of the simulation. Urbanised areas at the far side of the city centre tend to grow along the transportation corridors. Assigning shorter distance thresholds to the model causes the city to expand much more as an agglomerate pattern with the urbanised areas tending to concentrate close to the city centre.

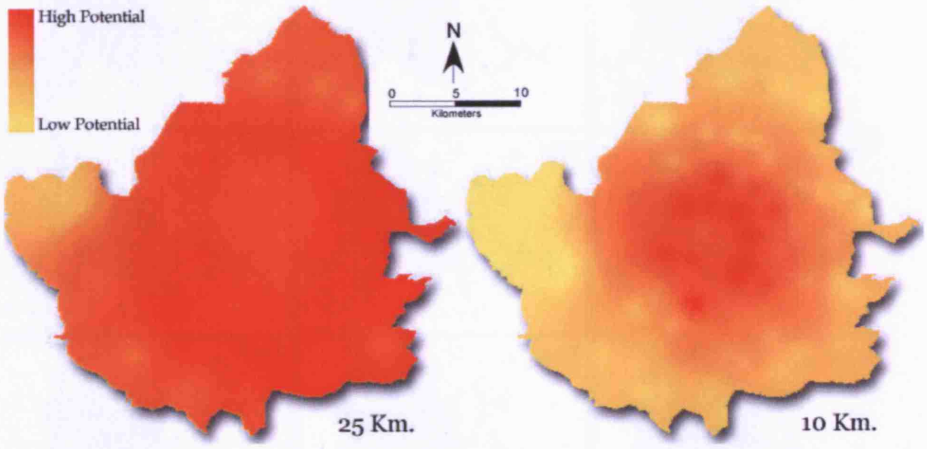


Figure 7.9 Centrifugal Surface Maps: 25 and 10 Kilometres

Figure 7.11 represents the comparison between the simulation results at year 2003 and the actual data in terms of land use/cover. The images clearly show that urbanised areas agglomerate in or close to the city centre, whereas the actual image shows, in contrast, that these urbanised areas appear all over the region but close to transportation features.

Quantitatively these two images were again measured using the confusion matrix as shown in Block 7.4. The overall accuracy of all classes for this simulation was 82.74%, whereas the Kappa coefficient was 0.68, again pretty close to the two sets of earlier results For each class on the other hand, the simulation generated 2,021 urban cells correctly from 4,515 pixels—or 44.76 percent. 2,453 pixels or 54.55 percent was generated as agriculture, and 31 pixels or 0.69% were forest areas. From the matrices, we can say simply that there is an accuracy of 44.76% for the urban class, 83.83% for agriculture, and 91.16% for forest areas.

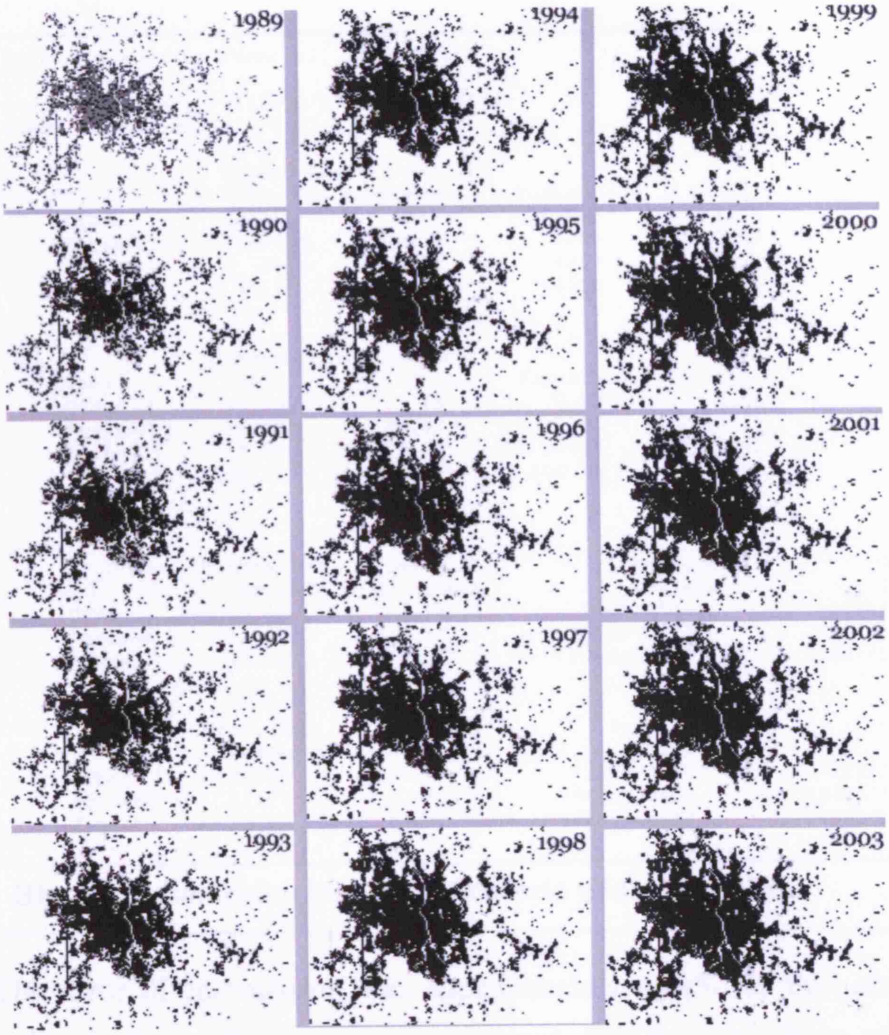


Figure 7.10 Urbanised Areas of Chiang Mai 1989-2003 from Scenario 3

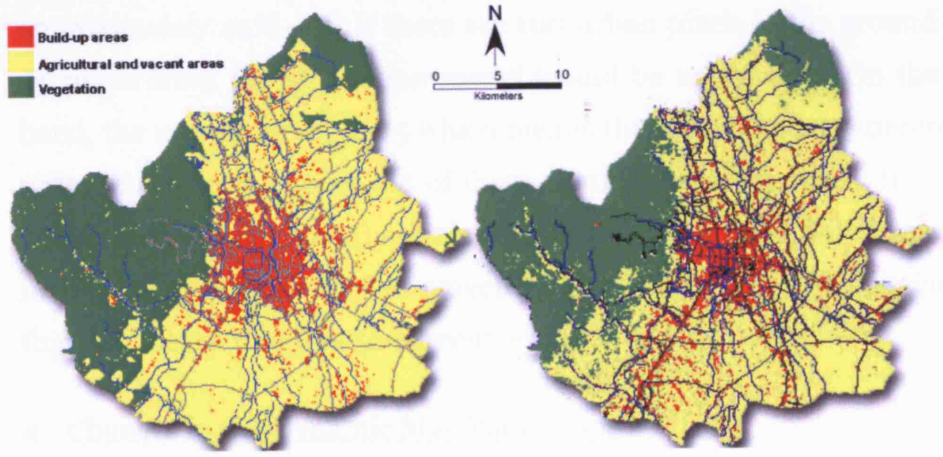


Figure 7.11 Simulated Land Use/Cover compared to the Real Classified Image Data of 2003 for Scenario 3

```
Confusion Matrix: [Memory1] (340x322x1)

Overall Accuracy = (43290/52319)  82.7424%
Kappa Coefficient = 0.6807

                Ground Truth (Pixels)
    Class         Urban     Agri&Bare     Forest        Total
     Urban         2021          2357        151         4529
     Agri&Bare     2463         26400       1290        30153
     Forest          31          2737      14869        17637
        Total      4515         31494      16310        52319


                Ground Truth (Percent)
    Class         Urban     Agri&Bare     Forest        Total
     Urban        44.76          7.48       0.93         8.66
     Agri&Bare    54.55         83.83       7.91        57.63
     Forest        0.69          8.69      91.16        33.71
        Total    100.00        100.00     100.00       100.00


       Class   Commission      Omission          Commission           Omission
                (Percent)     (Percent)            (Pixels)           (Pixels)
     Urban        55.38         55.24          2508/4529          2494/4515
     Agri&Bare    12.45         16.17          3753/30153         5094/31494
     Forest       15.69          8.84          2768/17637         1441/16310


       Class   Prod. Acc.     User Acc.          Prod. Acc.           User Acc.
                (Percent)     (Percent)            (Pixels)           (Pixels)
     Urban        44.76         44.62          2021/4515          2021/4529
     Agri&Bare    83.83         87.55         26400/31494        26400/30153
     Forest       91.16         84.31         14869/16310        14869/17637
```

Block 7.4 Confusion Matrix for Scenario 3 for Chiang Mai

The error of commission was approximately 55.4% for the urban class quite similar to the previous experiment. For the agriculture and forest classes, these were 12% and 16% respectively. The error of omission for the urban class was 55.24%. The producer accuracy for the urban class was approximately 45% -- or if there are 100 urban pixels in the ground truth, 45 pixels from the simulation model would be urban cells. On the other hand, the user accuracy is 45 which means that if the model generates 100 urban pixels in the space, 45 of these would be correct pixels. It is worth noting at this point that these results are satisfactory but only just so. The model is basically about "50 percent" right in terms of it predictions but this also means that it is 50 percent wrong.

4. Changes in the Dynamic Map Parameters

The dynamic map module is an important set of subroutines in the model used for calculating the constrained probabilities of each cell through the

predetermination of different weight scores for each map layer. The first three experiments were tested using the same weight scores defined using a trial and error method noting the results of the hypothetical experiments in Chapter 5. In this case, the weight scores of all the map layers were adjusted as shown in Table 7.1 above.

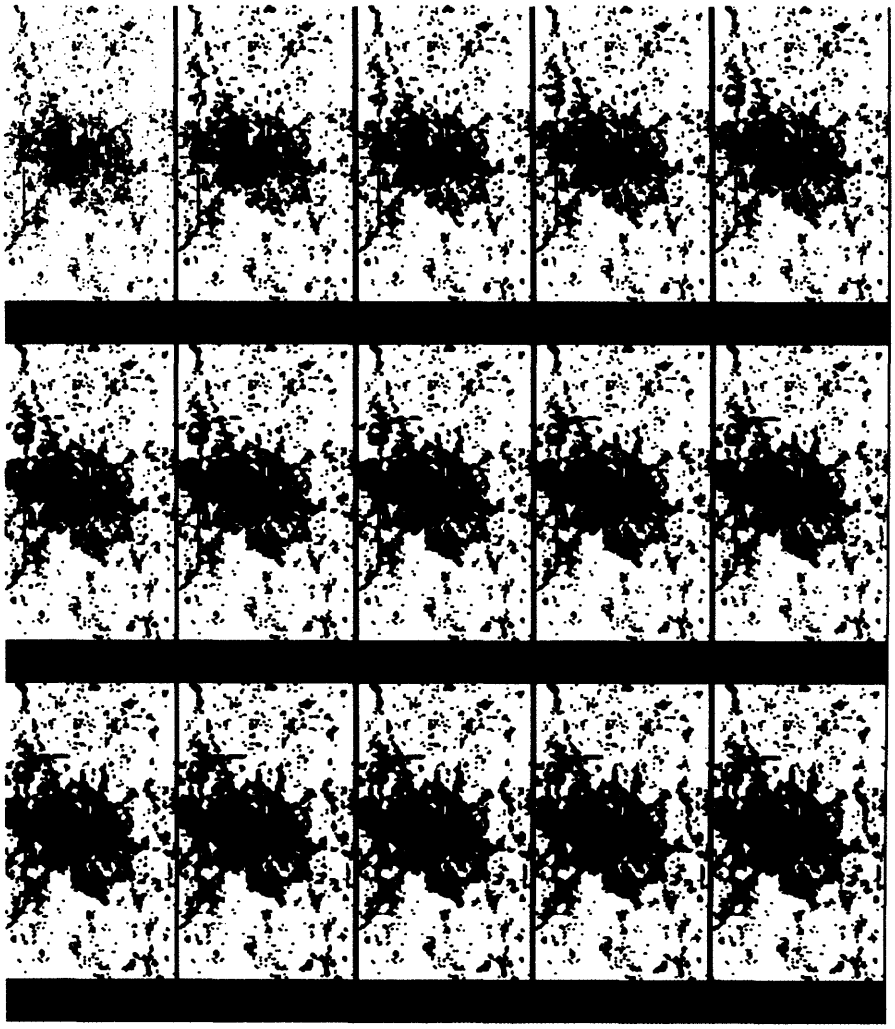


Figure 7.12 Urbanised Areas of Chiang Mai 1989-2003 from Scenario 4

Figure 7.12 illustrates the sequence of images resulting from this simulation using the given parameter sets for scenario 4. Urbanised cells still tend to appear close to road features but they also grow up faster in the area around the city centre. This causes urban agglomeration. Figure 7.13 illustrates the comparison between the land use/cover image result and the actual data, and here the pattern of urbanised areas around the city centre looks similar when compared to the actual pattern. However as

we reduce the weight score for the centrifugal surface, the images in both Figure 7.12 and 7.13 clearly show that a few urbanised areas grow far from the city centre.

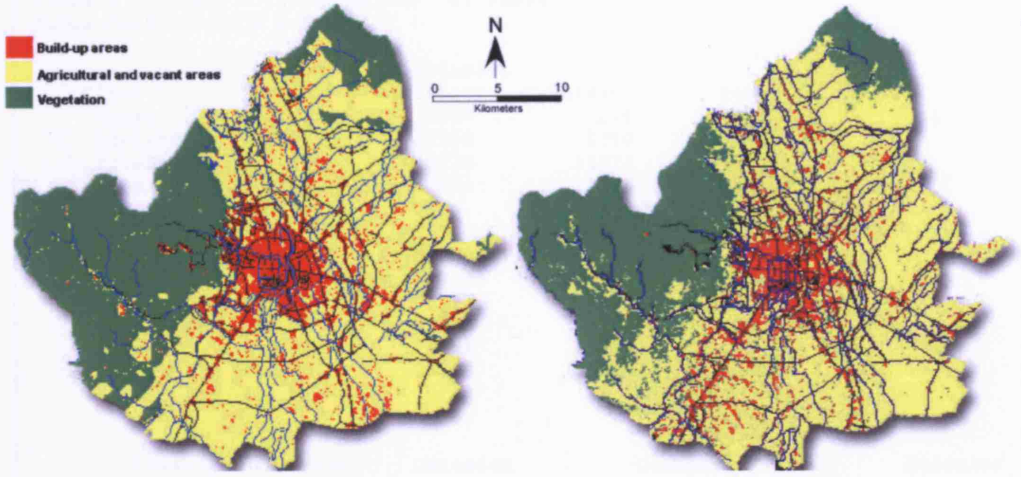


Figure 7.13 Simulated Land Use/Cover compared to the Real Classified Image Data of 2003 for Scenario 4

The overall accuracy shown in the confusion matrix in Block 7.5 is approximately 82 percent while the Kappa coefficient is computed as 0.67. For each class, the simulation is able to generate 2,149 correct urban pixels from a total of 4,515 ground truth pixels or 47.60% and the simulation generates 51.75% correctly as agriculture and 64% as forest areas. For other classes, the simulation generates the agriculture land use correctly as 82.43% compared to the ground truth data. The best accuracy was the forest class at approximately 90 percent.

The error of commission for the urban class was about 59%. On the other hand, the errors of commission in the agriculture and forest classes is still low at only 12% and 16% respectively. The error of omission is approximately 52% for the urban class, in contrast to the errors of omission for the agriculture and forest classes which are 17% and 9% respectively. The producer accuracy is around 47 percent, which indicates that almost 50% of the urban pixels in ground truth were matched to cells generated from the simulation. On the other hand, the user accuracy shows that if there are 100 urban pixels created from the simulation,

around 41 cells fit the pixel distribution in the ground truth. Both indexes show a very high accuracy for both the agriculture and forest classes.

Confusion Matrix: [Memory1] (340x322x1)

Overall Accuracy = (42882/52319) 81.9626%
Kappa Coefficient = 0.6699

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	2149	2807	226	5182
Agri&Bare	2337	25959	1310	29606
Forest	29	2728	14774	17531
Total	4515	31494	16310	52319

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	47.60	8.91	1.39	9.90
Agri&Bare	51.76	82.43	8.03	56.59
Forest	0.64	8.66	90.58	33.51
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	58.53	52.40	3033/5182	2366/4515
Agri&Bare	12.32	17.57	3647/29606	5535/31494
Forest	15.73	9.42	2757/17531	1536/16310

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	47.60	41.47	2149/4515	2149/5182
Agri&Bare	82.43	87.68	25959/31494	25959/29606
Forest	90.58	84.27	14774/16310	14774/17531

Block 7.5 Confusion Matrix for Scenario 4 for Chiang Mai

Discussion

As seen in previous chapters and noting the parameters shown in Table 7.1 above, there are several ways in which we can mix all the parameters in the model, and generate quite similar results. It is thus still very difficult to design a scheme to identify the best parameter values for there are many features that need to be evaluated and many of these must be accomplished in purely visual rather than numerical terms. Even if we had such a scheme for selectively moving through the parameter space, it is still hard to judge how to significantly choose parameters that gain the best results that are closest to the real world phenomenon. This study has used a series of trial and error methods to accomplish this choice and in the case of the Chiang Mai area, all these comparisons of the scenarios are summarised in Figure 7.14 and Table 7.5.

Figure 7.15 shows the results at each five year simulation for each scenario in visual terms for the urbanised areas. It clearly shows that the urbanised cells in the first scenario are distributed all over the space, whereas in the others, they are more concentrated close to the city centre. Only in the last scenario is there a slight trend for their distribution further out from the existing city. Compared to the pattern of urbanised cells (red cells) in the 2003 classified image in Figures 7.3, 7.8, 7.11 and 7.13, it appears that the most similar is our baseline, benchmark or archetypal scenario, the first one.

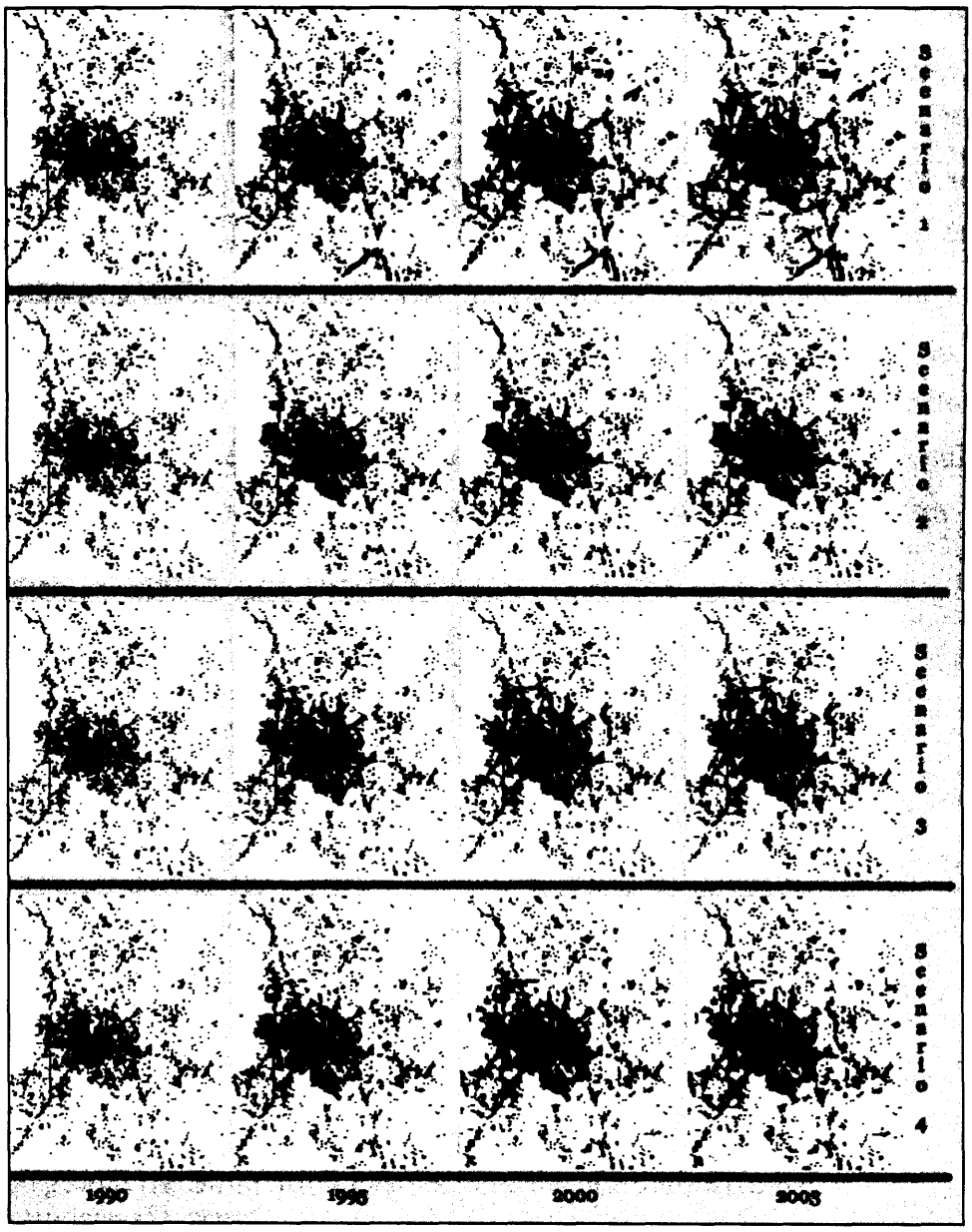


Figure 7.14 A Visual Comparisons of all Four Scenarios

Table 7.5 shows that all results are acceptable in terms of the confusion indexes. The overall accuracy and the Kappa coefficient average 82 percent and 0.67 respectively. This gives an acceptable performance in terms of the overall simulation generated by the model which we judge to be satisfactory. Nevertheless, when considering more detail in each class, we find some differences between the scenarios. In case of agriculture or the bare land and the forest classes, all the scenarios show a very high percent accuracy; approximately 80% for agriculture and 90% for forest class (see the producer accuracy in Table 7.5 below).

Table 7.5 Summary of the Accuracy Measures for all Scenarios for Chiang Mai

Scenario	Overall Accuracy	Kappa	Error of Commission			Error of Omission			Producer Accuracy			User Accuracy		
			Urban	Agri	Forest	Urban	Agri	Forest	Urban	Agri	Forest	Urban	Agri	Forest
1	81.38	0.66	60.89	12.09	15.38	49.59	19.05	9.2	50.41	80.94	90.8	39.11	87.91	84.62
2	82.77	0.67	55.18	12.77	15.77	57.81	15.83	8.7	42.19	84.17	91.3	44.82	87.82	84.23
3	82.74	0.68	55.38	12.45	15.69	55.24	16.17	8.84	44.76	83.83	91.16	44.62	87.55	84.31
4	81.96	0.67	58.53	12.32	15.73	52.4	17.54	9.42	47.6	82.43	90.58	41.47	87.68	84.27

Considering the accuracy measured for the urban class in all the scenarios, it is clear that the highest producer accuracy is 50.41% for scenario 1. Although very low compared to other classes, it is the most important measurable quantity that can be used to distinguish or qualify the results from each scenario. The second best is scenario 4 with a 47% accuracy of urbanised cells.

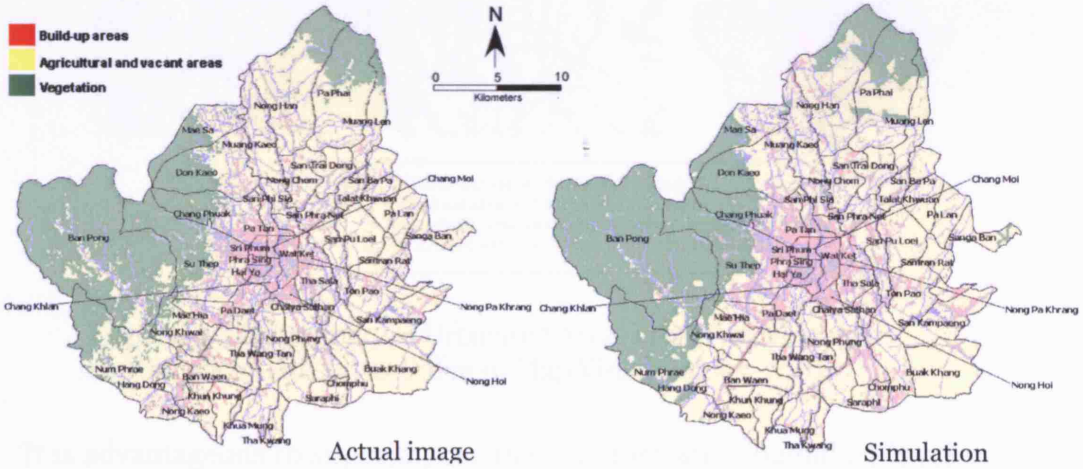


Figure 7.15 Comparison of Land Use/Cover between the Actual Image and the Simulation Results in terms of Administrative Boundaries in 2003

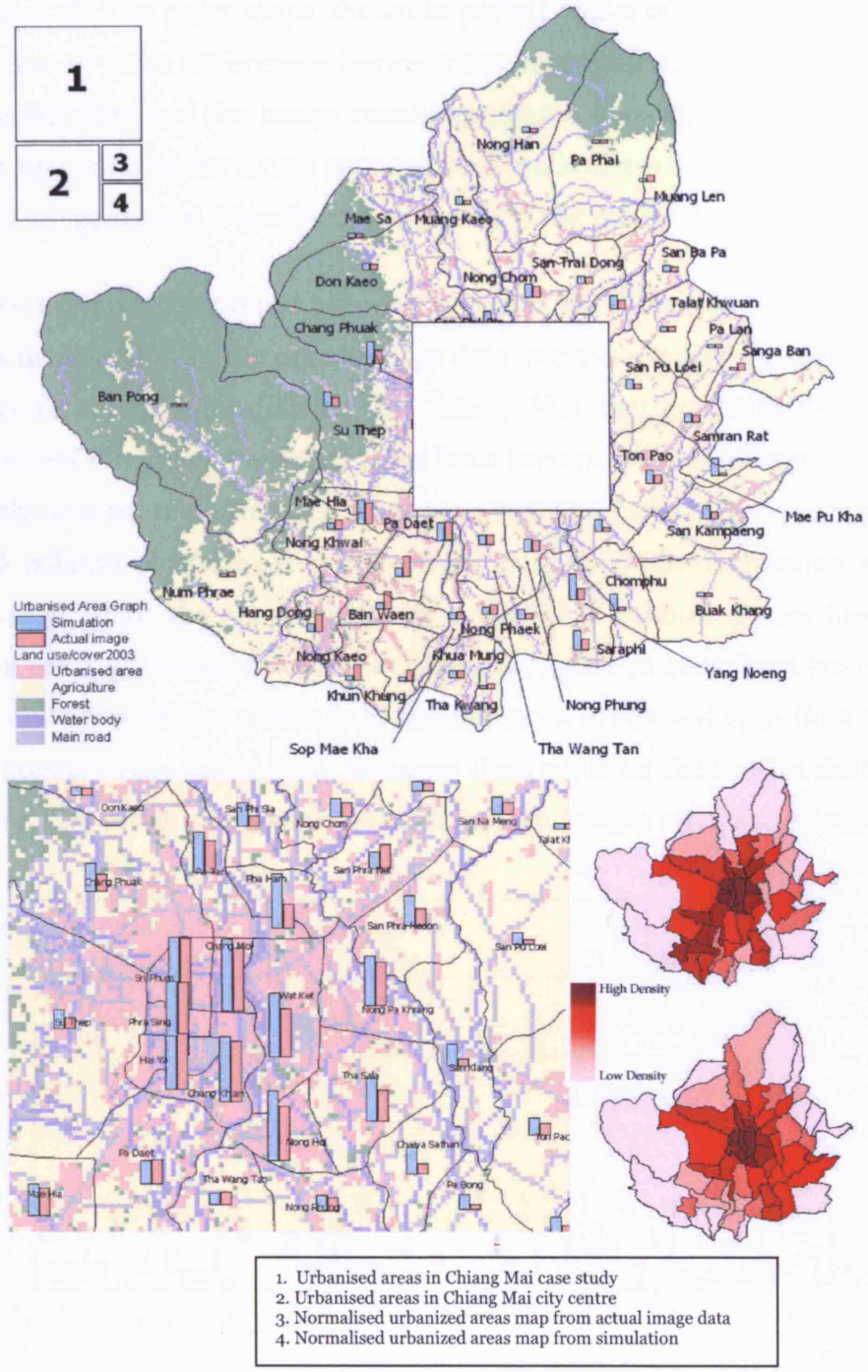


Figure 7.16 Summary of Urbanised Areas in the Actual Image and Simulation Result as Different Map Visualisations, 2003

It is advantageous to superimpose the administration boundary layer onto both the actual image and the simulation results so that we can evaluate the model in detail as we show in Figure 7.15. Moreover, other information can be visualised using the various geographical techniques shown in

Figure 7.16. The chart maps shown in parts 1 and 2 of Figure 7.16 clearly visualise all these differences between the urbanised areas obtained from the actual data and the image results generated from the simulation. The other two parts of the figure represent normalised graduated colour maps which are generated from the actual data and the model respectively.

Figure 7.17 provides overall information about the differences in urbanized areas measured from the actual image data and the simulation results. The x-axis represents sub-districts in Chiang Mai region while the y-axis represents the areas measured from both images, with the longer the x-y bar, the greater the difference for the particular sub-district in question. A blank column illustrates that the measured areas in the simulation result are bigger than the actual one, while the filled column shows that the actual areas are bigger than the simulation. As can be seen from the chart, the simulation model normally generates more urbanised cells than those that appear in the actual data. However if we consider the images shown in Figure 7.18, urbanised areas measured in both images are quite similar.

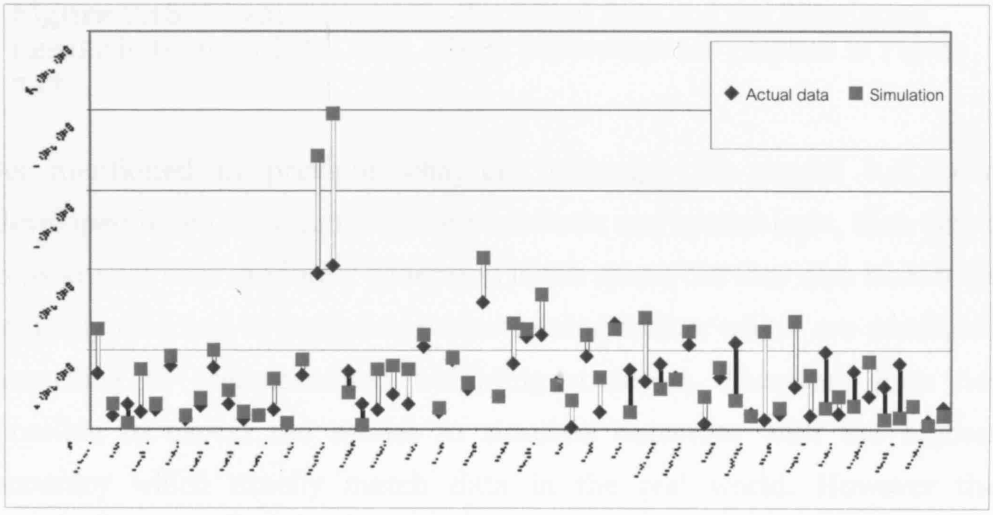


Figure 7.17 Differences in Urbanised Areas between the Actual Data and the Simulation Results for Administrative Areas

As seen in parts 3 and 4 of Figure 7.16, the model is able to provide outcomes quite close to the city centre in the area in the Chang Moi, Wat Ket, Chang Khan, Pha Sing, Hai Ya, Chang Phuak and SuThep sub-districts. The chart in map 1 of Figure 7.16 shows that most regions close to the city centre and through to the north and east are almost the same. Only

one sub-district, Mae Phu Kha, in the east of the city has a very different allocation of urban cells between the simulation results and the actual data. Most of the errors occur in the southwest of the city, in the San Pak Wan, Ban Waen, Hang Dong and Nong Keao sub-districts. These sub-districts have lower urban densities in the simulation compared to the real situation.

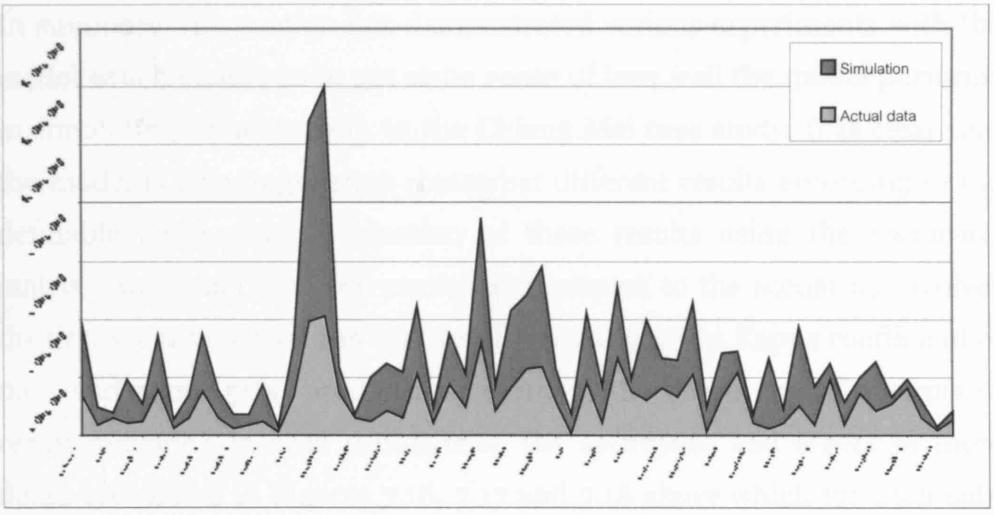


Figure 7.18 Urbanised Areas in the Actual Data and the Simulation Results in terms of Total Area, whose Differences are graphed in Figure 7.17

As mentioned in previous chapters, although the model has been developed based on a numerical probabilistic and spatial logic, then active cells are not only randomly generated in the space, but they also have to be logically matched to predefined constraining factors which are combined according to predetermined weighting strategies. Therefore it is not possible to expect the model to simulate outcomes with the highest accuracy which exactly match data in the real world. However the simulation results provide a similar trend to that shown in Figure 7.18 which is encouraging.

To compare the two data sets in quantitative fashion, the urbanised areas for each sub-district of both data sets have been normalised using the percentage of urbanised area as differences between the actual and simulated images. These data were paired and used in calculating the

correlation coefficient. The correlation was approximately 0.59, which represents only 35 percent of the variance in the differences in the images being explained. On the other hand, if we normalise the urbanised areas in terms of the total areas in each sub-district, the actual and simulation results have a much higher correlation which is approximately 0.80 or an explanation of some 64 percent of the variance.

In summary, this section has demonstrated various experiments with the model which enable us to get some sense of how well the model performs in simulating urban growth in the Chiang Mai case study. It is clear that the model is able to generate somewhat different results according to the definable parameters. Evaluation of these results using the confusion matrix shows that the best results with respect to the scenarios involves the first scenario which has an overall 81% accuracy, a Kappa coefficient of 0.66 and about 51% producer accuracy in the urban class. In terms of result matching in each sub-district, the accuracies and errors in more detail are shown in Figures 7.16, 7.17 and 7.18 above which for each sub-district, give a satisfactory correlation in terms of the urbanised areas of 0.58.

Simulating Phitsanulok 1989-2002

In this section, we will present the experiments for our other case study area in much the same way we have presented the Chiang Mai case study, but we will proceed a little more rapidly as we have already explained much of the basis for our empirical evaluation. As mentioned earlier in Chapter 6, Phitsanulok city occupies an important role in the regional economy of lower Northern Thailand; although the size of the city is smaller than Chiang Mai city, there has still been rapid urban growth in the region and this will represent another test of the model, albeit at a slightly smaller scale.

As in the Chiang Mai case study, four scenarios were prepared in order to create four different sets of results from which a suitable calibration could be evaluated. The population of each sub-district was dynamically defined

for every year and as we showed earlier, we set equal to 4 iterations per time cycle. The results generated from the simulations were evaluated with respect to the actual data in the same fashion as in the Chiang Mai case study.

The results shown in Figure 7.19 should be read in the same way as in the Chiang Mai experiments. For our base scenario 1, urbanised areas tend to expand along transportation routes. As seen in the figure, urbanised cells are generated around the edges of the city centre, but by the year 2002, it becomes clear that there are at least three agglomerated urban clusters which have been generated along the main route from the city centre.

Scenario 2 was simulated based on changes in the parameters associated with the generation of the dasymetric density map. The result shows distinct centralisation of urbanised areas associated with the density of the land demand in Muang Phitsanulok sub-district. Some urbanised cells are generated along the transportation routes but when we reduce the influence of the distance threshold from 25 to 10 km for scenario 3, Figure 7.19 shows that the urbanized areas expand due to the effect of the centripetal and centrifugal potential surfaces. It then appears that the urbanised cells are generated quite close to the city centre. Using a lower weight score for the centrifugal surface but a higher one for the centripetal surface map causes more cells to be created around the city as we show in scenario 4. A few urbanised cells are created along transportation routes that lie far from the city centre.

As discussed in the Chiang Mai case study, land use/cover maps generated from each scenario also have to be compared and evaluated in a quantitative manner. In Figure 7.20, we show land use/cover images for the year 2002 from the simulations in compared with the actual image data. The confusion matrices for each of the four scenarios are calculated as shown in Blocks 7.6, 7.7, 7.8 and 7.9.

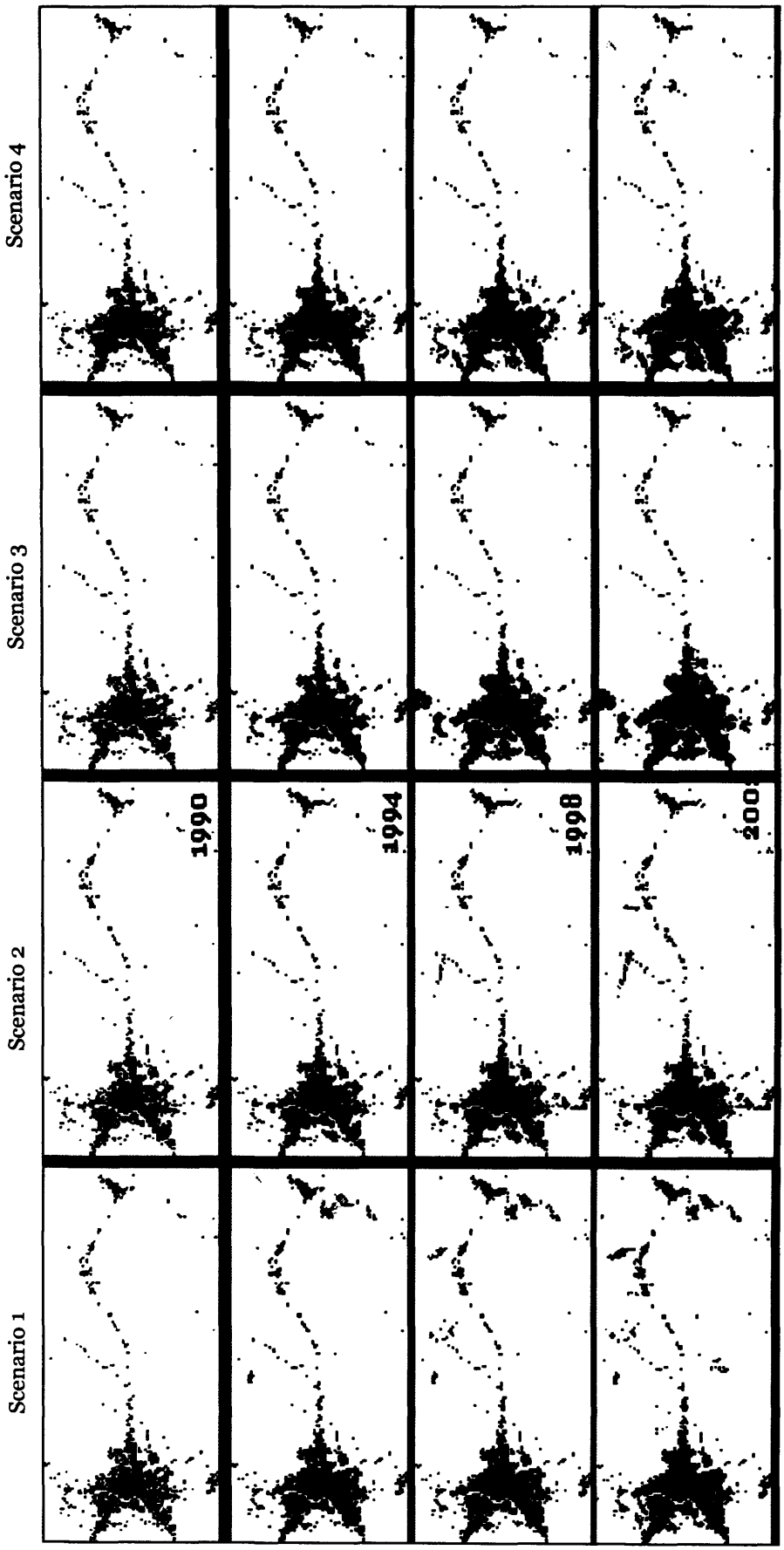


Figure 7.19 Comparisons of Urban Areas in the Four Scenarios for Phitsanulok

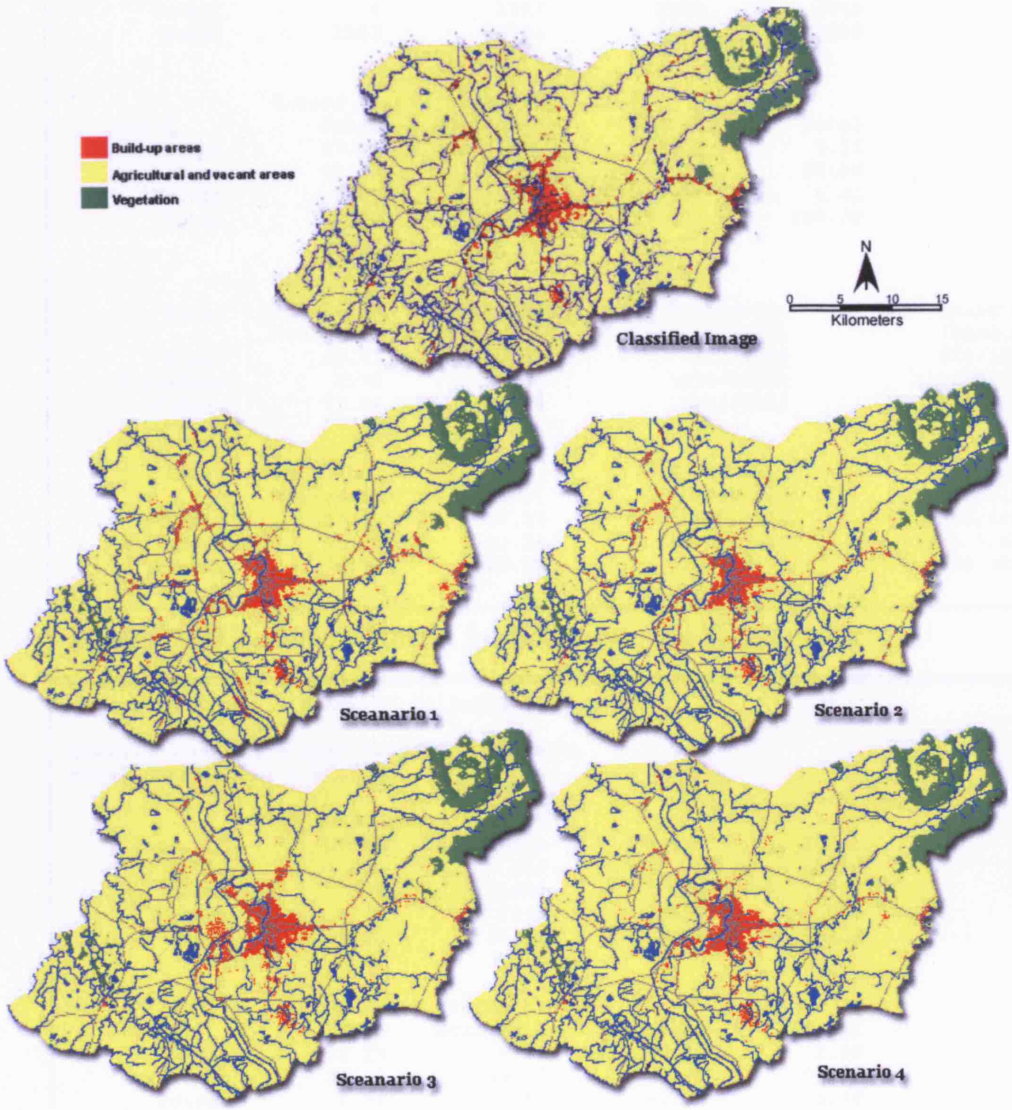


Figure 7.20 Land Use/Cover Images from the Actual Data and the Four Simulations in 2002

Confusion Matrix: [Memory1] (412x283x1)

Overall Accuracy = (54842/58408) 93.8947%
Kappa Coefficient = 0.6459

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	1085	799	2	1886
Agri&Bare	814	51205	640	52659
Forest	4	1307	2552	3863
Total	1903	53311	3194	58408

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	57.02	1.50	0.06	3.23
Agri&Bare	42.77	96.05	20.04	90.16
Forest	0.21	2.45	79.90	6.61
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	42.47	42.98	801/1886	818/1903
Agri&Bare	2.76	3.95	1454/52659	2106/53311
Forest	33.94	20.10	1311/3863	642/3194

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	57.02	57.53	1085/1903	1085/1886
Agri&Bare	96.05	97.24	51205/53311	51205/52659
Forest	79.90	66.06	2552/3194	2552/3863

Block 7.6 Confusion Matrix for Scenario 1 for Phitsanulok

Confusion Matrix: [Memory2] (412x283x1)

Overall Accuracy = (54998/58408) 94.1618%
Kappa Coefficient = 0.6516

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	995	522	1	1518
Agri&Bare	902	51444	634	52980
Forest	6	1345	2559	3910
Total	1903	53311	3194	58408

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	52.29	0.98	0.03	2.60
Agri&Bare	47.40	96.50	19.85	90.71
Forest	0.32	2.52	80.12	6.69
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	34.45	47.71	523/1518	908/1903
Agri&Bare	2.90	3.50	1536/52980	1867/53311
Forest	34.55	19.88	1351/3910	635/3194

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	52.29	65.55	995/1903	995/1518
Agri&Bare	96.50	97.10	51444/53311	51444/52980
Forest	80.12	65.45	2559/3194	2559/3910

Block 7.7 Confusion Matrix for Scenario 2 for Phitsanulok

Confusion Matrix: [Memory3] (412x283x1)

Overall Accuracy = (54815/58408) 93.8484%
Kappa Coefficient = 0.6464

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	1136	813	0	1949
Agri&Bare	764	51139	654	52557
Forest	3	1359	2540	3902
Total	1903	53311	3194	58408

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	59.70	1.53	0.00	3.34
Agri&Bare	40.15	95.93	20.48	89.98
Forest	0.16	2.55	79.52	6.68
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	41.71	40.30	813/1949	767/1903
Agri&Bare	2.70	4.07	1418/52557	2172/53311
Forest	34.91	20.48	1362/3902	654/3194

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	59.70	58.29	1136/1903	1136/1949
Agri&Bare	95.93	97.30	51139/53311	51139/52557
Forest	79.52	65.09	2540/3194	2540/3902

Block 7.8 Confusion Matrix for Scenario 3 for Phitsanulok

Confusion Matrix: [Memory4] (412x283x1)

Overall Accuracy = (55038/58408) 94.2302%
Kappa Coefficient = 0.6559

	Ground Truth (Pixels)			
Class	Urban	Agri&Bare	Forest	Total
Urban	1008	530	0	1538
Agri&Bare	892	51459	623	52974
Forest	3	1322	2571	3896
Total	1903	53311	3194	58408

	Ground Truth (Percent)			
Class	Urban	Agri&Bare	Forest	Total
Urban	52.97	0.99	0.00	2.63
Agri&Bare	46.87	96.53	19.51	90.70
Forest	0.16	2.48	80.49	6.67
Total	100.00	100.00	100.00	100.00

Class	Commission (Percent)	Omission (Percent)	Commission (Pixels)	Omission (Pixels)
Urban	34.46	47.03	530/1538	895/1903
Agri&Bare	2.86	3.47	1515/52974	1852/53311
Forest	34.01	19.51	1325/3896	623/3194

Class	Prod. Acc. (Percent)	User Acc. (Percent)	Prod. Acc. (Pixels)	User Acc. (Pixels)
Urban	52.97	65.54	1008/1903	1008/1538
Agri&Bare	96.53	97.14	51459/53311	51459/52974
Forest	80.49	65.99	2571/3194	2571/3896

Block 7.9 Confusion Matrix for Scenario 4 for Phitsanulok

Table 7.6 Summary of the Accuracy Measures for all Scenarios for Phitsanulok

Scenario	Overall Accuracy	Kappa	Error of Commission			Error of Omission			Producer Accuracy			User Accuracy		
			Urban	Agri	Forest	Urban	Agri	Forest	Urban	Agri	Forest	Urban	Agri	Forest
1	93.89	0.64	42.47	2.76	33.94	42.98	3.95	20.10	57.02	96.05	79.90	57.53	97.24	66.06
2	94.16	0.65	34.45	2.90	34.55	47.71	3.50	19.88	52.29	96.50	80.12	65.55	97.10	65.45
3	93.84	0.64	41.71	2.70	34.91	40.30	4.07	20.48	59.70	95.93	79.52	58.29	97.30	65.09
4	94.23	0.65	34.46	2.86	34.01	47.03	3.47	19.51	52.97	96.53	80.49	65.54	97.14	65.99

All the scenarios resulted in almost the same overall accuracies and Kappa coefficients as shown in Table 7.6. This implies that the overall land use/cover of all scenarios generated from the simulations were similar to the actual data with approximately 94% accuracy and 0.64 for the Kappa index. However, comparing all these results, it is necessary to consider the greater detail in the confusion matrix. The results for the first and third scenarios returned highest producer accuracies of approximately 57% and 59% respectively which indicates that the model has generated an image pixel close to the urban classification in the actual data. This measure clearly distinguishes these two scenarios from the others. In the next section, we will additionally consider the detail of these two scenarios by performing an evaluation of the urbanised areas simulated and observed within each sub-district.

Discussion

The results from the actual images and the simulation in 2002 can be both given a more focussed geographic reference by being easily overlaid with other map layers that present more spatial information. In this section, the results from scenario 1 are superimposed on the administrative boundary layers so that we can compare the urbanised areas simulated in each sub-district with the actual data as we show in Figure 7.21 below.

As in the Chiang Mai experiment, Figure 7.22 presents a comparison between urbanised areas in the actual image and in the simulation. Figures 7.22a, b and c illustrate graduated colour maps for normalised urbanised areas of the actual image and the simulation for scenarios 1 and 3 respectively. These maps show the concentration of urbanised areas in the Nai Muang sub-district and others that are close to it. Figure 7.22d

represents comparisons between the total urbanised areas measured from each of these three images. As seen in the chart map, most of the graphs visually represent patterns and trends for the simulation results which are quite compatible to the actual data.

Figure 7.23 illustrates these comparisons in detail. The charts obviously illustrate the errors between actual data and the simulation results comparing the sizes of the urbanised area measured from actual data with the results from scenario 1 and scenario 3 respectively. The bars represent errors or differences between areas measured from the two datasets for each sub-district. In both visual and quantitative terms, it is clear that scenario 1 has slightly higher errors than the scenario 3.

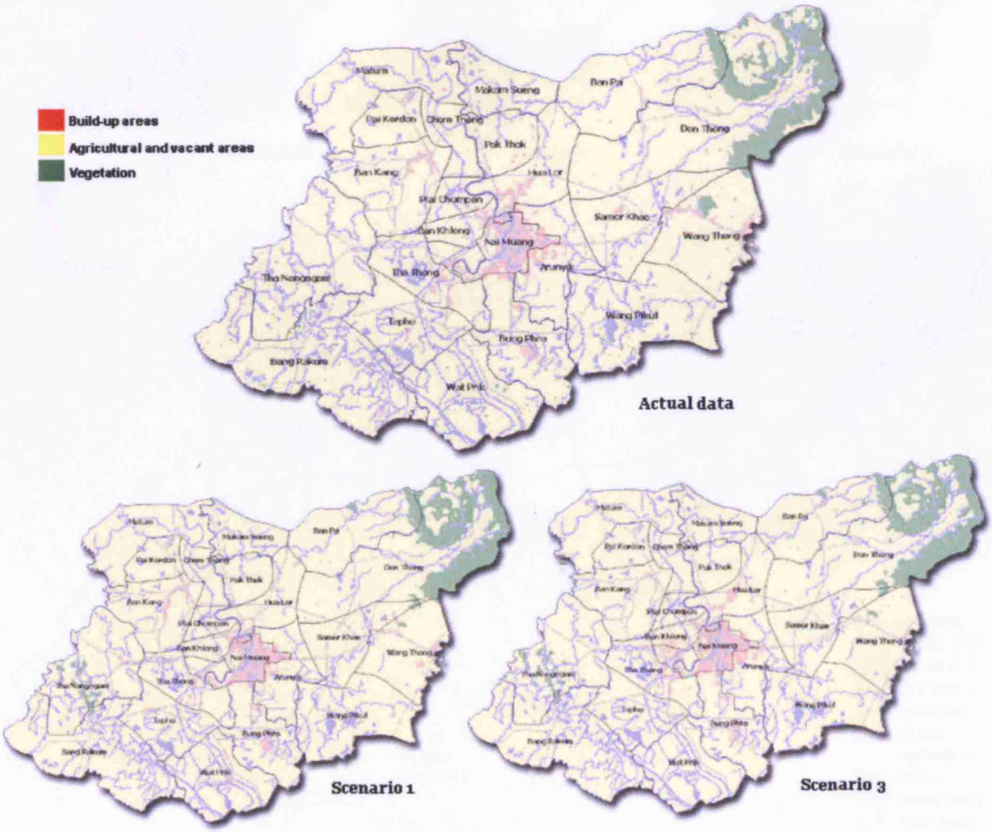


Figure 7.21 Comparison of Land Use/Cover between Actual Image and Simulation Result

The data can also be usefully assessed in a quantitative way as Figures 7.24a and 7.24b show in terms of the two scatter plots of actual urbanised areas against the simulation results. Interactions between the datasets are depicted in form of r-squared values explaining some 93 percent of the variance for scenario 1 (correlation = 0.96) and 92 percent for scenario 3 (correlation = 0.95) respectively. The coefficients show that there are very high positive relationships between the datasets. This emphasizes that the model performs much better than the model for Chiang Mai city which is encouraging because the results for the first city are more disappointing than we originally anticipated.

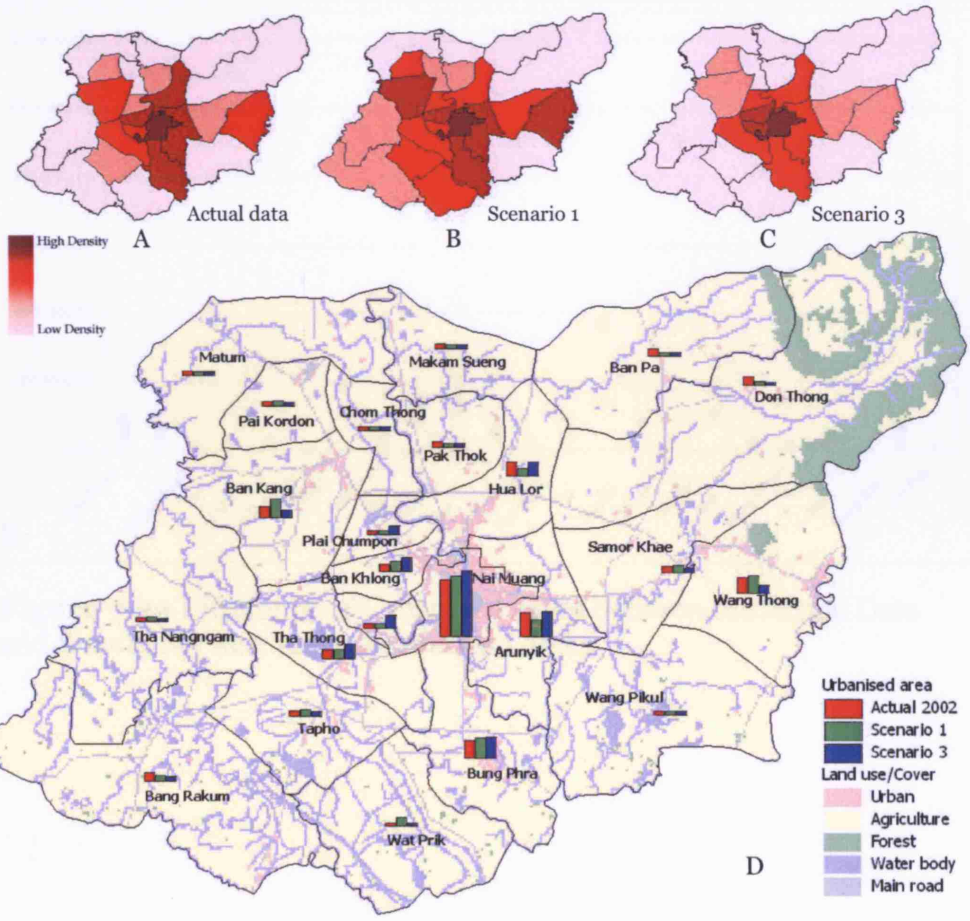


Figure 7.22 Summary of the Urbanised Areas from Actual Image and Simulation Results for the Two Best Scenarios

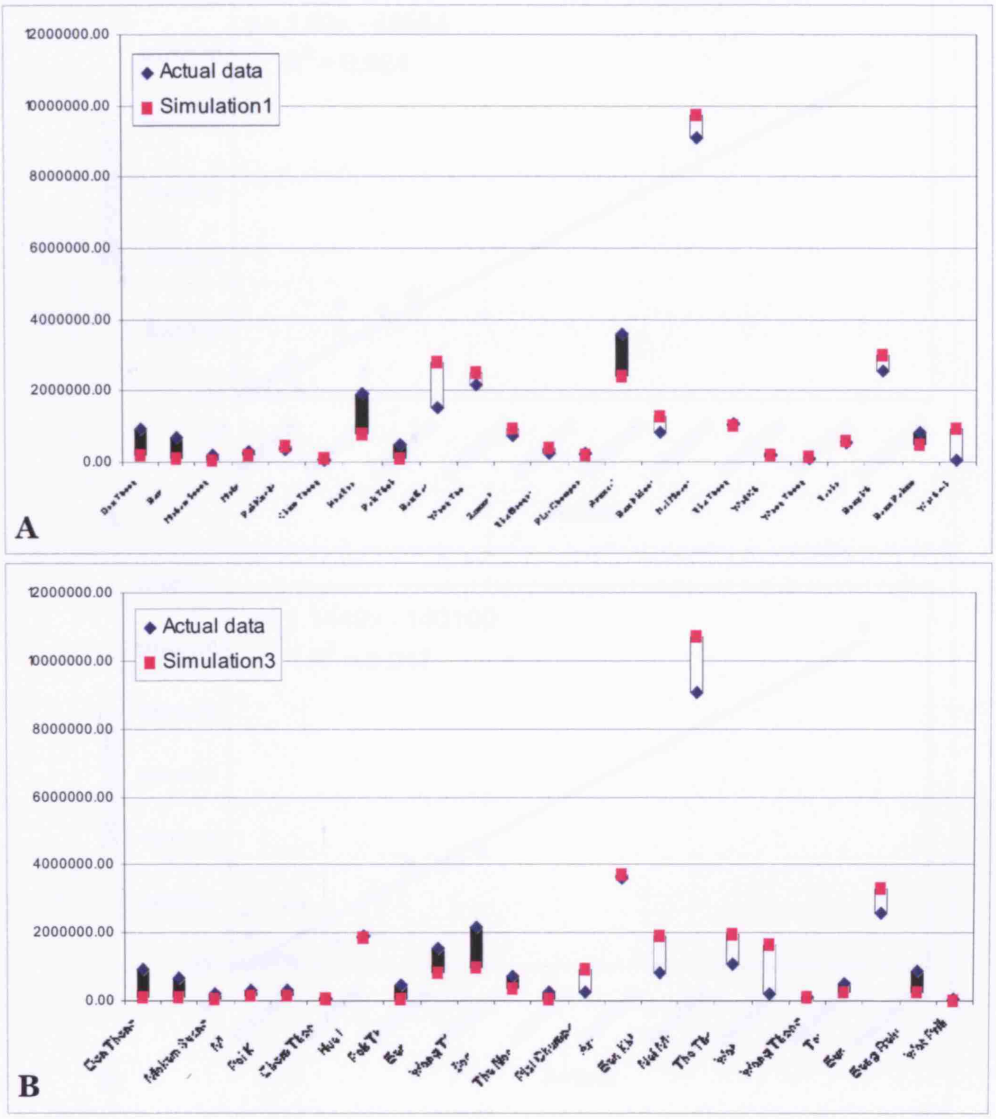


Figure 7.23 Difference of Urbanised Areas between the Actual Data and Simulation Results for Scenarios 1 and 3

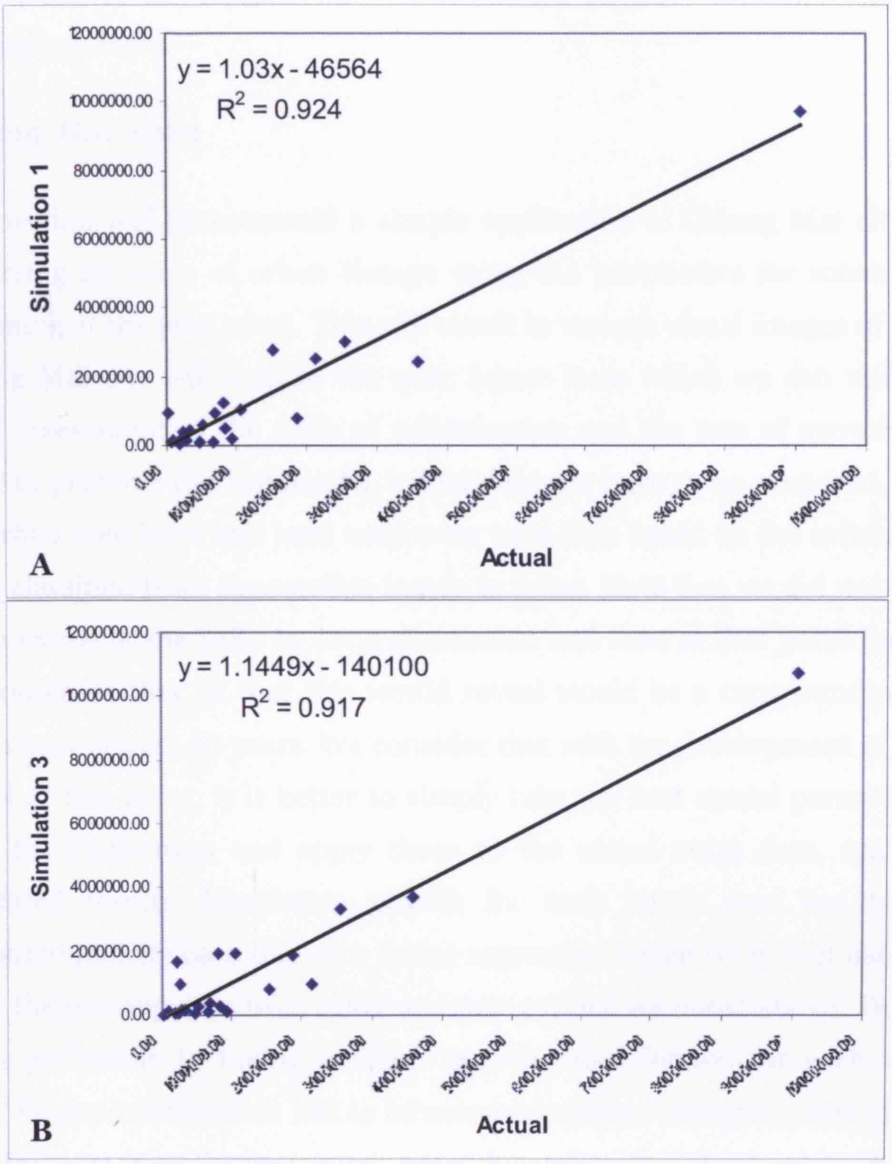


Figure 7.24 Scatter plots of actual data and result from Scenario 1 and 3 for Phitsanulok

3. Forecasting Experiments with the Applicable Models

We now need to complete our development of the model with some short and sharp ideas about how we might use it in a forecasting situation. Our development as we noted at the onset of this chapter, will not be comprehensive but we simply sketch how we might proceed in making

forecasts of urban change in our two case study areas for the next 20 years, the medium term.

Chiang Mai 2023

This section will demonstrate a simple application to Chiang Mai city by predicting 20 years of urban change using the parameters for scenario 1 beginning at the year 2003. This will result in various visual images of how Chiang Mai city will look in the near future from which we can make a rapid assessment of the scale of urbanisation and the rate of growth. In order to perform the simulation, initial datasets have to be prepared, and the urban seed layer and land use/cover were thus based on the urbanised areas classified from the satellite image in 2003. Note that we did not take the outcome of the 1989 to 2003 simulation and start at that point for we are conscious that all that this would reveal would be a compounding of errors over almost 40 years. We consider that with the development of this model at this point, it is better to simply take the best model parameters from the calibration and apply these to the actual 2003 data, not the simulated results. Population growth for each future year has to be computed and we base this on a linear regression which we in fact used to fill in the missing data from 2002 and 2003 which we noted above. This is much preferable to taking random growth rates defined for each time state. Transportation data had to be assumed with no changes in any of the road features from the year 2003, notwithstanding that there are bound to be changes in the road infrastructure to 2023. In fact one possible extension of the model would be to add a layer in the prediction of road infrastructure as a positive feedback from the allocation of population within the model.

Figure 7.25 represents a prediction of the urbanised areas simulated from the model over this 20 year period. Considering the urbanised areas around the city centre in year 2003, the urbanisation is predominantly along the superhighway and inner ring road. As the simulation proceeds, there is urban expansion along the routes to the east and south of the city. After the year 2012, some nodal points where the radial roads intersect the

second ring appear as very clear urban clusters on every side of the city. A little later in the simulation, urban areas at the edges of the city grow up and are finally connected to urban clusters at these nodal points. Some development links the city to nearby small towns, especially in the east and south of the city as shown in Figures 7.25 and 7.26. Between the eastern outskirts of the city and a small town in the east in the Ton Pao sub-district previously separated from the city, both are obviously connected with urbanised areas along the road by 2010. This also happens in other areas to the far south side of the city, for example in the Yang Noeng, Hang Dong and Mae Hia sub-districts.

During the 20 year forecasting period, these results visually show that many urban cells are generated around junctions of main roads with the middle and outer ring roads radiating from the city centre. These results still represent a trend that indicates the possibility that these areas will become urban in the future, similar to the urbanisation that has occurred around the second ring road during the years 2003 to 2014 shown in Figure 7.26.

Although the model generates urbanised cells in the space based on probabilities calculated from the dynamic constraints which adapt themselves during the simulation process, running these simulations with the same parameter sets several times generates outcomes that imply a fairly similar pattern of urbanisation and a standard well-structured urbanisation process. This application however does demonstrate another way in which the model can perform in predicting possible spatial scenarios. Although we are sure that the model is not generating accurate futures, we do consider it is able to generate rather well informed spatial trends in population distribution which in terms of the urbanization in Chiang Mai, would appear to be quite plausible. This means that we consider the model to be a good reflection of the existing processes of land use change in this region and these forecasts give us more confidence in the model's predictive ability.

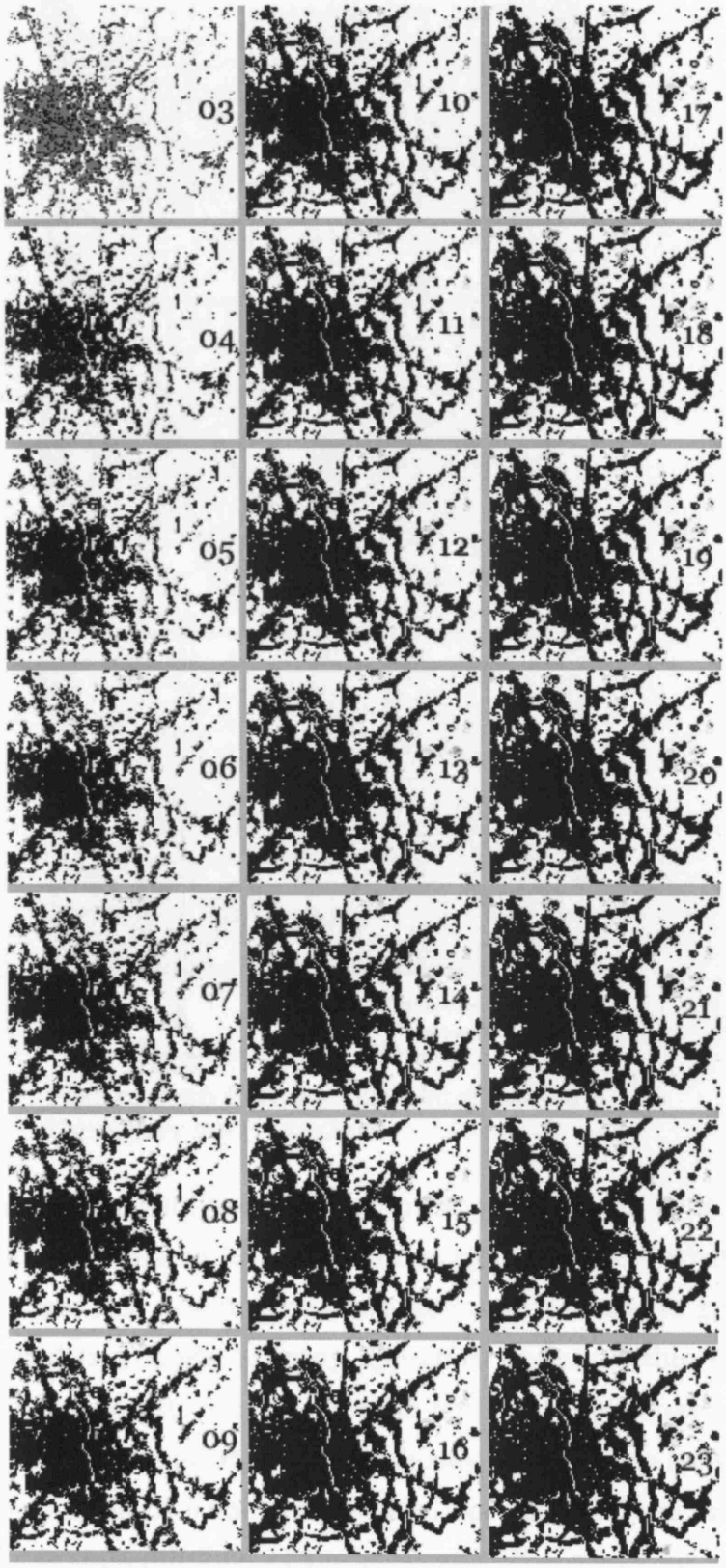


Figure 7.25 Forecasting the Chiang Mai Urbanised Areas 2003 - 2023

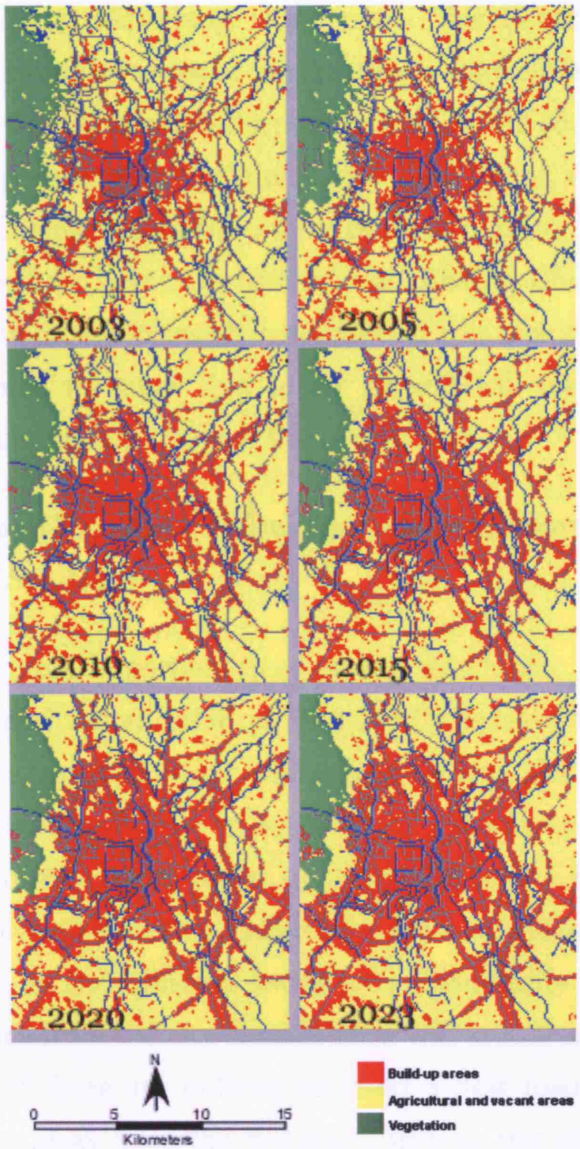


Figure 7.26 Forecasting Chiang Mai Land Use/Cover 2003 - 2023

Chiang Mai is the most important city in the northern region of Thailand. Urbanisation in this area has been intense since 1980 and what we both observe and are able to simulate is that the same processes of growth that have operated from 1980 to 2003 appear to operate within the model when we make forecasts over the medium term future. We may learn from the actual data in year 2003 for it has already yielded some facts attesting to the urbanisation of some sub-districts. For example, the Hang Dong sub-district is presently one of the most attractive zones for real estate business investment. Many light- and medium-density residential units have been constructed along the main transportation artery from the city,

along the Chiang Mai-Hang Dong road. This is probably because this territory is within the range of the city centre, and close to a large shopping centre, with the Chiang Mai Airport located in the south not far from the city. The superhighway or Chiang Mai-Lam Phang road is a new route connecting Chiang Mai to two major provincial cities, Lam Phun and Lam Phang, as well as being the major road linking Chiang Mai to Central Thailand. The model forecasts confirm that it is along these highways and in their immediate areas that urban growth will take place in the near future, continuing the trends of the last decade.

Phitsanulok City: New Transportation Infrastructure: Testing a 'what-if' Scenario

One of the most important uses of the simulation model is to predict 'what-if' scenarios that differ quite markedly in certain specific areas from reality. In the last section our scenario for Chiang Mai is more general and focuses on the broad issue of 'what-if' the current trends in urbanisation continue. However there are many other types of 'what-if' question. For instance, how will growth be affected if there is a plan to construct a new project such as an industrial estate or a community university on a site some distance from the city centre? How will urbanisation in an historical city appear if there are policies to build a new town centre in a lesser favoured city away from the historical sites in the old city? In the same way, what will happen if new roads are planned which in a way is the same as adding new road infrastructure as input to the predictions of general urbanisation in Chiang Mai. Generally, these questions cannot be answered unless we define new planning strategies as input to the model which is one of the time honoured ways of using these models in forecasting.

Constructing the model to gauge the impact of such strategies is a necessary response to these questions. It is clearly safer to examine such plans in the model prior to making a decision to perform such radical action in the real world, and in using the model in this way, we are not only able to learn from model outcomes but we may also realise some other

issues about how well the model works under different scenarios. In fact testing the model in this fashion by defining such a range of scenarios is as fruitful as exploring solutions to the model through variations in the parameter space.

This section will experiment with the Phitsanulok dataset by setting up a 'what-if' scenario to explore how urban form would be changed through the construction of a new road system. To perform this scenario, it is necessary to prepare new transportation datasets. In this case, the Department of Public Works and Town & Country Planning (DPT) of Thailand have designed a new future road layout for Phitsanulok city in the city master plan of 1999 which can be easily adapted to the simulation. Unfortunately, the plan does not quite have the detail that we need for the model in terms of the dates at which different streets are to be constructed. In these circumstances, the year of construction was assumed and then attributed to each road segment to render the simulation closer to the reality which we show in Figure 7.27.

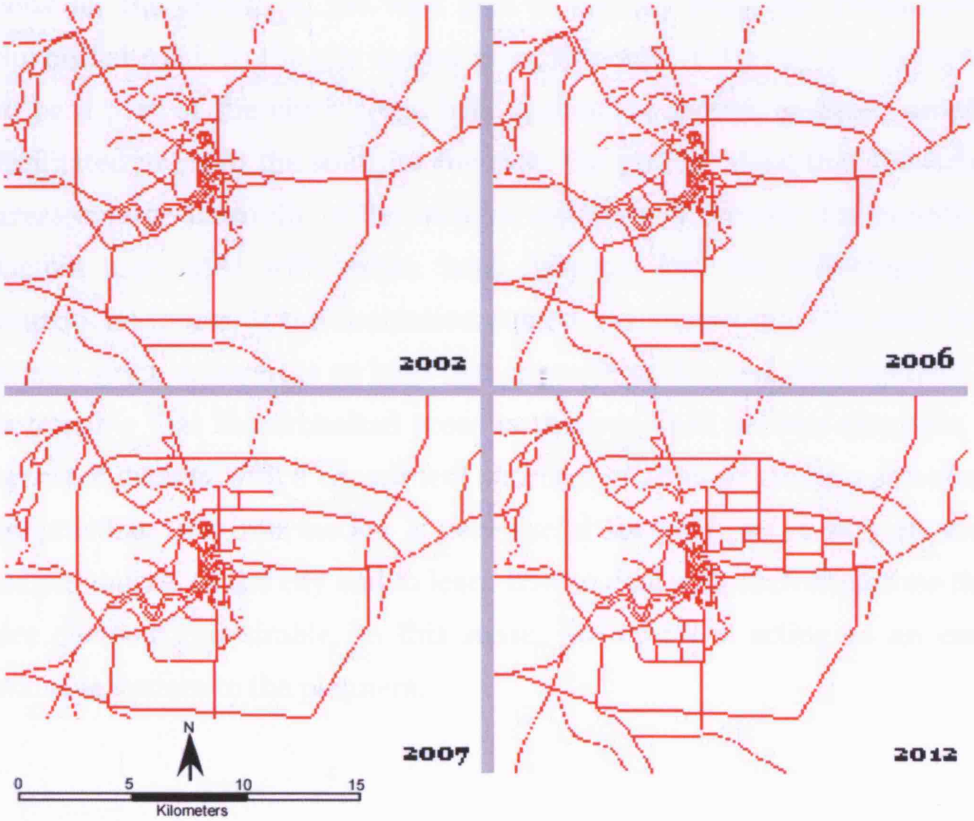


Figure 7.27 A Proposed New Transportation System for Phitsanulok

In fact in terms of the DSSM code, part of this was redefined to enable the program to read these additional attributes of the road layer. Once the simulation reaches the years in question within the simulation, the program repeatedly scans the road map and updates those features of the map that are relevant to new transportation infrastructure within the model space.

Figures 7.28, 7.29 and 7.30 above represent the sequence of growing urban areas and land use/cover resulting from the simulation. Three trends of growth are clearly visible, in the west, southwest and east of the city centre. Growth expands along the Phitsanulok-Sukhothai road to the west of the city (see Figure 7.31a) while a second one appears over the length of Boromtrailokanat road from the city to the southwest (Figure 7.31b). The last one is the urbanised area along the Phitsanulok-Wang Thong road to the east of the city (see Figure 7.31c).

The simulation results indicate how important it is for the city planner to consider the growth in the west part of the city along the Phitsanulok-Sukhothai road. In the city master plan, this side of the city was designed to be a part of the city's green belt. Instead of growth in more suitable facilitated areas in the south or the east, the results show that the urban areas emerge naturally in the western outskirts of the city due largely to factors associated with urban land demand, and the centrifugal and centripetal forces. If the simulation outcome is an accurate one and there are no strict constraints on land that is predicted for such development, it is possible that the urbanised areas in this zone will become excessive in terms of density with a consequent shortage of appropriate infrastructure. In practice, this information is very useful for planners to foresee what might happen to the city and to learn how to deal with such situations that are deemed undesirable. In this sense, the model is acting as an early warning system to the planners.

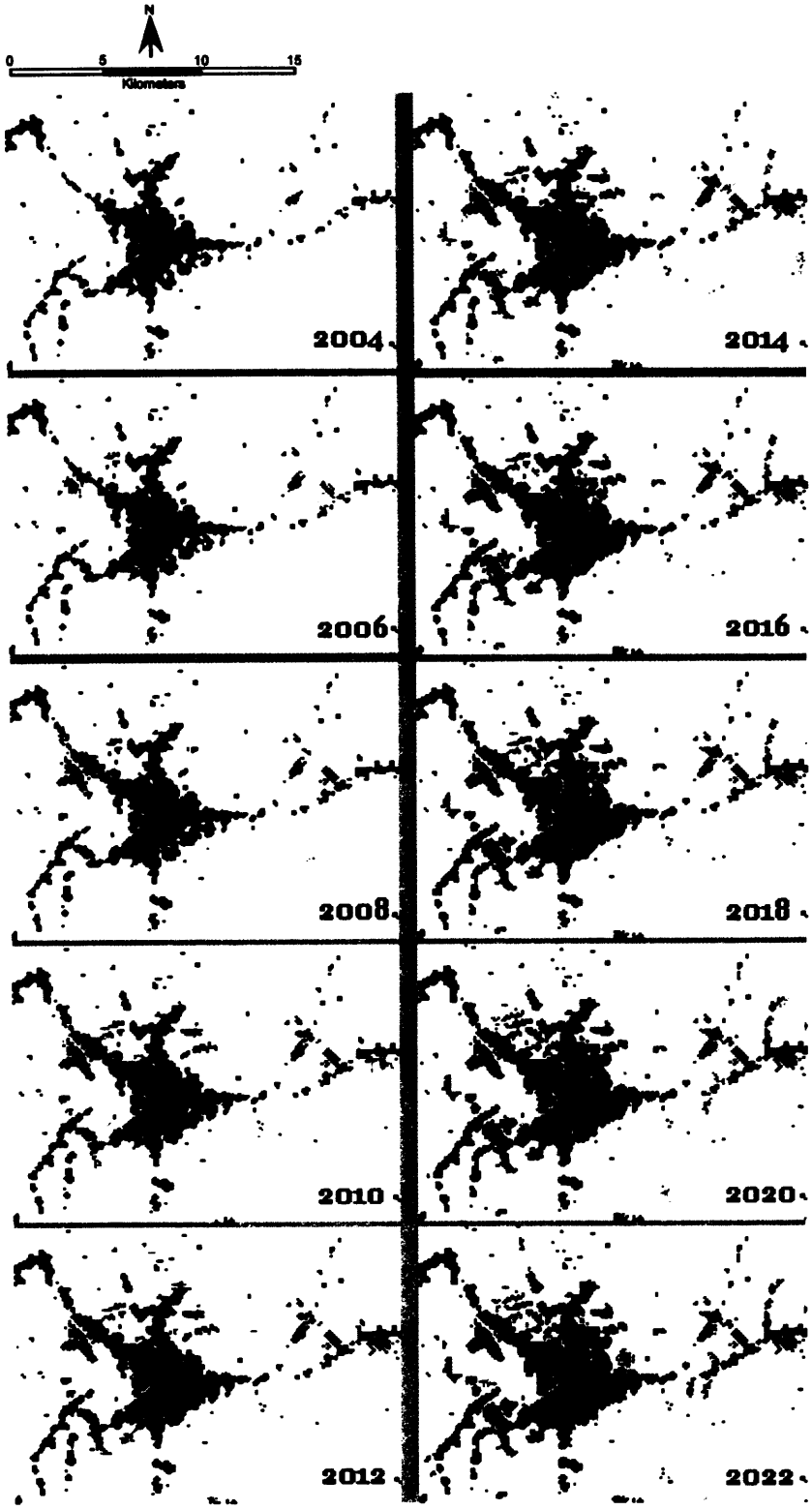


Figure 7.28 Urbanised Areas of Phitsanulok 2002-2022 with New Transportation Infrastructure and Continuing Urban Growth

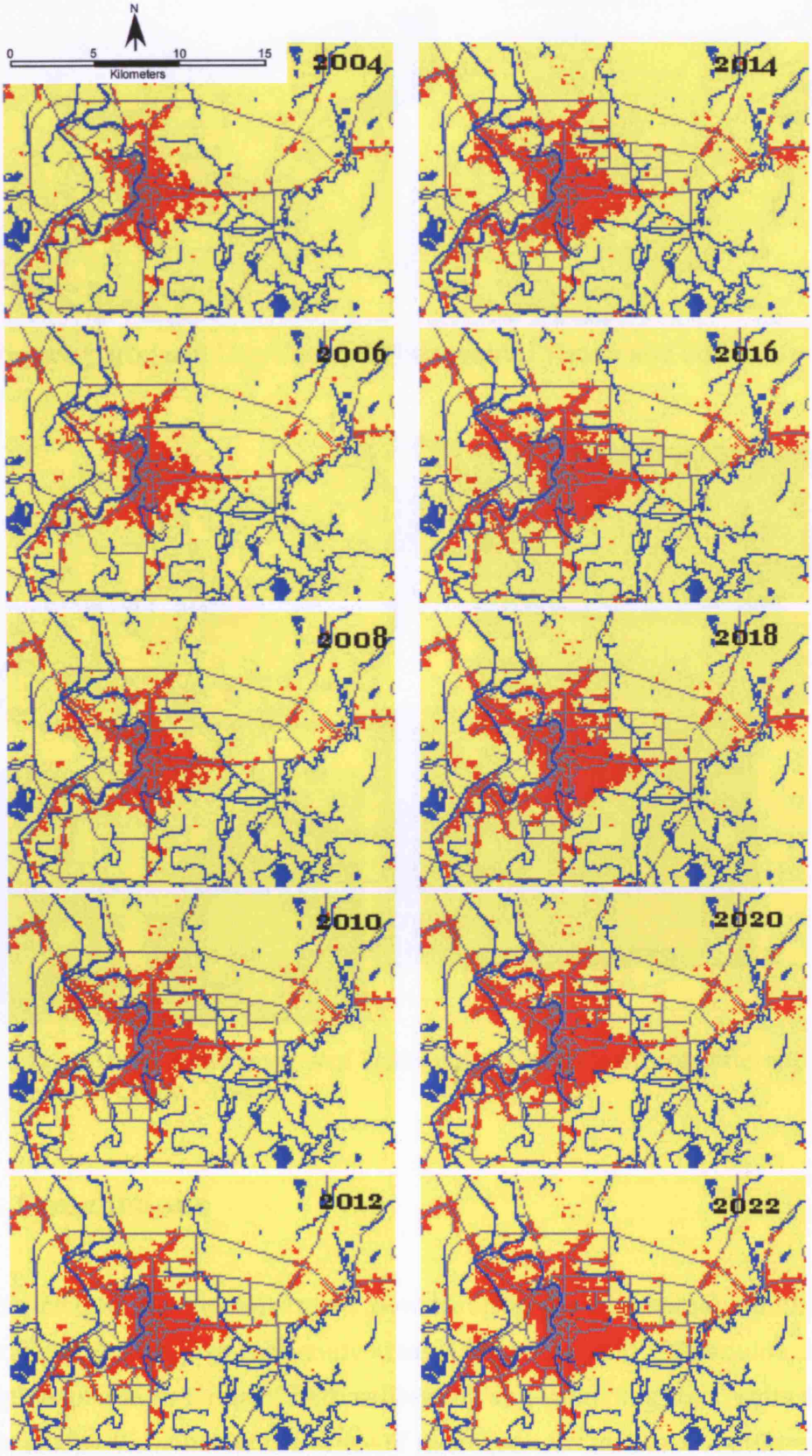


Figure 7.29 Land Use/Cover of Phitsanulok City from 2002-2022

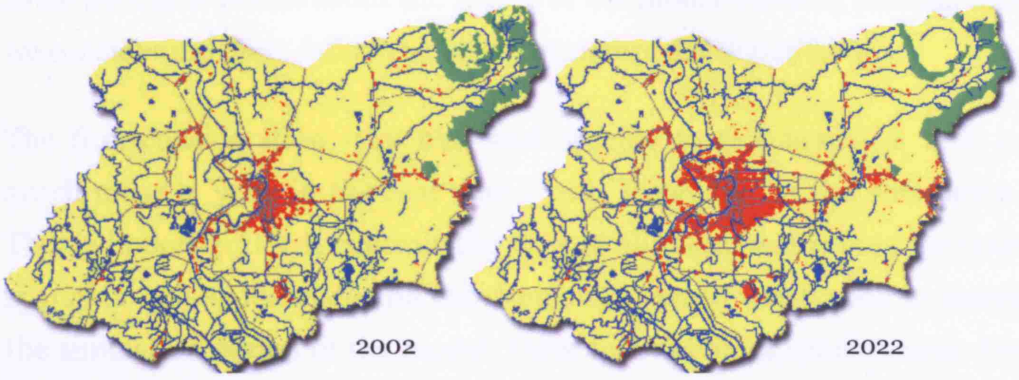


Figure 7.30 Land Use/Cover of Phitsanulok in 2002 and 2022

Figure 7.31 Some Images of Phitsanulok City in terms of Different Areas of Urban Growth

4. Conclusions

This chapter has demonstrated how to bring the conceptual simulation model to reality. Two case study areas, Chiang Mai and Phitsanulok, have been explored in terms of both calibrating and validating the results from the model. We consider the range of model options in terms of differences in model parameters and the case studies themselves to be sufficient to determine the model's suitability and in this closing section, we will make

some pertinent points about the future of the model-building process that we consider uniquely define the model structure developed here.

The first consideration very new and specific to this model is how to synchronise a 'year' in the real world with 'iterations' in the simulation. This is important for it controls the flow of all the dynamic data for both spatial or non-spatial data and the simulation results which need to match the simulation period of the model. Once time becomes synchronous, the second issue is to choose the initial parameters for running the simulation. Referring to Chapter 5, we chose parameters from the best results there and tweaked them a little for the actual runs here but we were unable to explore the entire range of combinations for this would have resulted in huge range of model parameter sets. It is an issue that dominates all modelling of this kind and it is something we flag up as being important to future research.

The model worked reasonably well and we have been successful in getting it to return visual outcomes as sequential sets of map layers. These we feel provide the model builder and user with a much richer set of visualizations than any that we have seen hitherto. In the experiments, Chiang Mai city had much more rapid growth than Phitsanulok city. The results from both were tested by comparing actual data collected from the 2002-satellite image through the confusion matrix from which it appears that the model performs less well as the rate of growth increases. In a sense, this suggests that when there is more to explain, the model finds it harder to do this although we cannot generalise from two cases. The confusion matrices provide overall accuracy between 80 to 90 percent with the Kappa coefficient around 0.6. This indicates that generally the model is able to yield fairly accurate results across all three land use classes: urban, agriculture and forest. Whereas considering just the urban class, the Phitsanulok case has yielded the outcomes more accurately than in the Chiang Mai case. This is possibly due to Chiang Mai having many more centres around the city as well as its division into many administrative units. This implies of course that urban growth in the large city of Chiang Mai is much more complex than in Phitsanulok.

We cannot expect a really accurate result from this model because of the way we choose so many of its elements using random structures. Many drivers in this model have been implemented using random methods and in sense, this means that the model produces a 'typical' urbanisation scenario. Rather than a model of Phitsanulok or Chiang Mai, it is a model of a generic Thai city. Hence, each time the simulation is performed, it offers quite different results. However, these results do trace an outline of generic urban patterns for each city as we demonstrated in the scenarios discussed earlier. Although we could not find out definitively which corresponding parameter sets can be matched to different city situations, the outcomes shown above do provide instructive examples for testing and evaluating the model.

The last section of this chapter implemented two forecasting examples which took advantage of the structure of the model. Both applications were involved in the prediction of future urban growth over the medium term of 20 years in both cities. For Chiang Mai, the simulation was performed to test whether or not the growth trend was similar up to the year 2023 as in the years 1989-2003. The results clearly indicate that the urbanised areas tend to expand to the east and the south of the city, also representing the increasing importance of all three ring roads in the future. In case of Phitsanulok, the new road network presented by the city planning office of Thailand was the main factor in implementing a 'what-if' scenario. The results show little change in the expansion of the urban form but they did point to some problems in over development in parts of the city. So far these experiments have ensured that it is possible to use the model for 'what-if' scenarios and adapt it to observing and learning about the city (and about the model). Finally, we have implied ways in which the model can be used as part of a decision support system for planning embracing both trend projections, 'what-if' scenarios, learning about the city, and using the model to visually communicate ideas about the present and future.

CHAPTER EIGHT
Conclusions and Future Developments

This final chapter provides an overall picture of the model developed in this study so that we might set this in context and suggest ways in which this research might be taken forward. We will concentrate on the major issues that have dominated this discussion, firstly the integration and creation of the dynamic model from two basic techniques based on cellular automata and the embodiment of data into the model through GIS modelling, and the development of the nodal regional system through techniques of spatial interaction and potential. We have spent over half this thesis on model development and the implementation of a hybrid simulation model – DSSM – and the rest of the thesis on its application to real world cities in Thailand, which have utilised the model to represent predictable urban and regional phenomena. Our emphasis has been on model development through theory, hypothetical experiment, and then through applications, all of which go towards making the model acceptable and relevant. This chapter will concentrate on these issues and will then move to future developments for DSSM based on further improvement and manipulations through exploration.

1. On the Model

The various concepts behind the model were discussed in Chapters 2 and 3. In essence, what we suggested was to develop the model by taking the traditional idea of cellular allocation using CA procedures and to embed spatial data and concepts into such allocation using the medium of raster based GIS modelling and the theory of the nodal regional system which would generate the relevant socio-economic data. As we argued in previous chapters, this structure was presented in previous chapters through at least five key issues encompassing techniques and visualisation derived

from general theories of urban development. These are described as follows.

Adding Temporal Variables into GIS Analysis and Modelling

Geographic phenomena, social, economic and physical, are dynamic by their nature. Such phenomena are never steady and does not occupy any equilibrium state permanently; stabilisation does occur but any equilibrium is dynamic and is dominated by change which can be abrupt and discontinuous, with phenomena transmuting from one state to another over rather different but nevertheless observable time periods. When we examine many geographical phenomena with respect to changing location, in this case at the cellular level, and consider the transforms from one state to another, it is easy to see how temporally dynamic the geographical phenomena can be. Space is strongly related to time as we described in Chapter 2. In describing locational phenomena, temporal variables should never be omitted as they represent an alternative dimension to geographical space. Even the simplest analyses of geographic phenomena require temporality in their representation so that they are able to describe space more precisely and more accurately. When constructing complex spatial models, there is a need for much more focus and emphasis on defining the interaction between space and time.

The GIS components in the DSSM model have been designed to define input and output as feedback systems through various causal loops based on the time states, instants, and periods over which the dynamics of the model develop. Although the GIS components in this study do not completely describe all the features in a spatio-temporal model of a city in that we need to approximate the way spatial and temporal information is represented, the DSSM model does provide a good example of the possible ways in which time might be introduced into GIS models. As we observed in all the outcomes shown in Chapters 5 and 7, changes in the locations and attributes of each factor which are driven by the CA model through time cause continual modifications to other factors at other, later points in time through the mechanisms of positive feedback. These transformations

provide the changes necessary to the entire mechanism of the system, in this case, the urban-regional system.

Moreover, building the model through the approach to mapping and GIS that we have adopted and which we consider to be an important innovation in this model allows us to visualise outcomes as thematic maps. When we add the temporal aspect to the model, the semi-continuous results that are generated act to produce time-stamped thematic maps of each factor that change over the predefined time intervals. The results from Chapter 5 clearly represent this point as we observe changes to the land uses and population densities over time. Unlike other models that create dynamic maps for only one theme or layer, the variety of thematic maps we are able to compute and visualise not only let us observe how the model actually works but also communicate its results in analysis and decision-making.

Embedding Spatio-Temporal Relations into Cellular Automata

As we noted at the beginning of this thesis, standard CA models are built around four key components: the cell space, cell states, transition rules and neighbourhoods as we discussed in Chapter 2. The complexity of the cellular transition processes is then provided by the type of cell states in their neighbourhood and how the transition rules enable actions to be made for changing the cell states based on the states of the cell's nearest neighbours. In this way, transformation of the states within the CA space occurs. In embedding this conception of how we model CA within some real application, we then have to focus on extending the model to embrace various constraining factors which ultimately lead to a multi-layered CA modelling approach. These constraining factors are mostly defined in static terms and thus they have less importance in the process of spatial transformation. Integrating this GIS modelling approach into a constrained CA model inevitably makes the model more complicated. But the processes in constrained CA do enable a wide set of variables represented as thematic maps to be combined with a standard set of GIS functions.

Including this spatio-temporal GIS approach within CA allows us to build parallel CA processes. Output from one CA space can be immediately used as input to another through the sharing of the same constraints. As in this model, while the urban space is affected by spatial information from the land use/cover map as a constraint on the CA process, cells in the land use map are updated from what is happening in the urban space. At the same time, cells are also being modified by other sets of transition rules that are applied to the land use map itself. Instead of returning the usual sorts of visual results and quantitative information that we associate with standard CA, our DSSM model fuses these various techniques by linking and enhancing various input data which is continually modified in terms of its outputs over the entire system as the various model feedback effects operate through time.

We have thus shown that the raster-based GIS and CA model are compatible in ways that go much beyond the ideas that have been introduced in previous models such as those introduced in IDRISI and Arc-Info, for example (Weiner 1997; Clarke et al. 1997). As both raster-based GIS and CA are cell-based structures, there is an obvious convenience in fusing them together. This study has demonstrated the clear point that it is possible and indeed necessary to create a model by taking advantages from both techniques which use the same cell-based structure. This suggests that it is then practicable to apply many raster GIS functions to the CA. DSSM utilises several combined GIS functions in creating the constrained thematic maps which drive various parts of the model. The map algebra approach and its cartographic model is one of main sets of functions used here, for example in constructing the population dasymetric maps, and in calculating the final probability fields for the overall constraining factors; focal operators are applied in part to compute these probabilities for some constraints, zonal operators are utilised to organise cells into zones, and interpolation operators are used to create potential surface maps based on incorporating spatial interaction effects into the model.

Giving Visual Significance to the Dynamics of the Nodal Regional System

Urban and regional development is typically complex, even if we try to generalise all its processes as being simple classifications of key attributes into observable types and even when we decrease the spatial scale of the region for analysis. It is hard, for example, to visualise how the nodal structure of the processes can be created without developing a simulation model at the cellular development level which aggregates up to the larger regional scale.

This model brings most of the components in the nodal region approach into its dynamic to incorporate and explore the mechanism of how a region evolves hierarchically. The results from Chapter 5 demonstrate how cells transform themselves and grow into city clusters, how they can evolve into realistic city patterns, constrained by factors which reflect the space they are growing within. When urban areas get bigger from increasing cell generations, the corresponding nodal system which reflects the hierarchy of centres over the time, also grows in parallel. At the same time, other features of the system show how spatial interactions emerge between the various nodes. These outcomes illustrate stages in the evolution of the nodal region through spatial interaction, networks, nodes, hierarchies, surfaces and diffusion. These all develop as simultaneous rather than chronological processes but they still reflect causes and effects that can change independently of one another or synchronise themselves through time.

Modelling Space and Time with Object-Oriented Programming

As mentioned earlier, there are both advantages and disadvantages to implementing CA on a GIS platform or to constructing the various CA and GIS components of the model in a mainstream programming language. This model has been built using JAVA that is one of the best-known object-oriented program (OOP) platforms. In addition to basic arithmetical and logical commands that can be used to control the simulation similar to

those in procedural and modular programming, OOP has several other properties that make it is suitable and practical for creating space-time models. For example, OOP can create prototypes (classes) that contain the blueprint of states (variables) and its behaviours (methods). Later the prototype can be used to create as many objects as are needed that come with their own changeable states and behaviours. The dynamic array property, vector, for example, can be used in order to collect many objects without predefined dimensions and associate the retrieval of each object within the program by its own individual index. These technical issues are important to the efficient construction and running of the model and as we noted at the outset of the thesis, readers should mount the program which is contained in the accompanying CD-ROM in the sleeve at the end of this thesis and explore its operation.

Moreover, as OOP like JAVA is an open and cross-platform programming language, it can be broadly used to build many different applications involving CA, GIS and spatial analysis models. OOP also has another useful property in that it can simply share classes from one application in building another. Recently if you try searching for JAVA libraries on the Internet, you will find many program libraries that can be directly used in your own application. Some of them open-source libraries that can be explicitly embedded into very specific code.

JAVA has its own graphic user interface (GUI) components that can be admirably used to build graphical applications like CA and GIS models. Applications created on JAVA can be interfaced as a standalone application that can run on many platforms: Windows, MAC OS-X, Linux or UNIX. It also can interface with JAVA Applets providing applications which can be distributed worldwide through the Internet (for details of using JAVA in spatial sciences, see Wood, 2002).

Representing Complex Behaviours

It is not possible to build models that are completely accurate portrayals of real world phenomena because these are complex by their nature and thus

in some sense, unpredictable and not entirely definable. Even in small and simple social situations where we think we are able to ensure control over the entire system, other factors we do not expect can intervene and add to the systems unpredictability. Therefore, when constructing a model to describe or predict, we can expect, at best, that it can provide an "*almost accurate*" answer. In short, this means that we would not expect a model like this to be able to be calibrated to ensure optimality with respect to the observed data.

Complicated social situations have a much wider array of factors that are unpredictable and unexpected; these then go beyond the region of "orderly" states to the territory of "chaos" which finally leads to a chaos that is too complicated to understand (for ideas about chaos and complexity, see Byrne 1998, Ward, 2002, and Waldrop 1992). In fact, complexity is a concept that many scholars have recognised as defining systems that exist on or at "the edge of chaos". As complexity models such as CA are in fact defined as ordered sets of rules, they do provide outcomes that are far more controllable while at the same time suggesting that chaos is never far away. A good example is in the first and perhaps simplest CA model based on the set of transition rules in Conway's Game of Life discussed in Chapter 2.

Experimenting with the DSSM model with the hypothetical test data in Chapter 5, represents a number of very clear points for organizing such model systems in a way which do as in fact provide results about complex spatial behaviour. Even though transition rules and constraining factors are predefined, running the model many times always offers different outcomes that characterise unpredictable processes and patterns. Including the probabilistic properties causes the model to get closer to the edge of chaos and although we cannot define such capricious behaviour, only hinted at in such models, the model developed here goes far beyond the generation of deterministic models fashioned for the simulation of urban systems a generation or more ago.

2. On Applications

Real World Urbanisation as Spatial Organisation

The results we have presented in many parts of this thesis lead to the important question as to how urbanisation in reality reflects the processes of spatial organisation. We have modelled these using the urban-regional concept of the nodal region, and tested the model with real datasets from two cities. As we observe in the results visualised in Chapter 7, the simulations do in fact provide adequate pictures of urbanisation in both cities compared to reality. Additionally, we evaluated the outcomes in a quantitative manner and our general conclusion is that these outcomes are sufficient to consider the model as being a good representation of the way we can simulate such virtual urbanisation. Hence, we would argue that real world urbanisation in regions such as those we have chosen are in fact characterised by the processes underlying the evolution of a nodal regional system.

That means that urbanisation is reflected in the major stages, which are described in the nodal region approach. Each city or town is represented as an individual centre or node within the variegated hierarchy of places, and each hierarchical node symbolises its level of importance in the regional space. Urbanisation reflects the synthesis of many spatial influences from every city and town that underlie the forces which determine the evolution of the urban-regional space; in other words, this reflects the fact that each city or town has different degrees of spatial interaction at various levels. Higher hierarchically positioned cities or towns have more intense interactions due to the operation of the forces of centralisation and decentralisation. These interactive forces effectively contribute to the spatial trend in the direction of urbanisation. This indeed is one of major causes in how cities appear in terms of their urban morphology. It is possible to find leapfrog and ribbon developments in the data and in our simulations (see Daniel and Hopkinson 1989, Holcombe et al. 1999 for instance), because the diffusion processes that work in urban systems and

in our model, scatter new urbanised areas away from the existing centres to many peripheral and even remote potential zones.

Considering the classified images for both cities in the years 1989 and 2002 (2003 for Chiang Mai city) illustrated in Chapter 6 for the data and in Chapter 7 for the simulation results, these clearly represent trends in urban development for both major cities, which diffuse and expand in directions that portray the importance of the corresponding nodes or centres and their spatial interactions.

The only issue that this model has not explicitly handled, and this may be possibly improved in the next version, is the simulation modelling of a dynamic transportation network. In other words, we have not modelled the two-way feedback between changing urban development and transportation infrastructures. The reasons are clear: firstly network systems intelligently created from algorithms in dynamic models do not match very well the evolution of real world transportation networks. In developing algorithms to construct road features based on existing road networks, there are clear difficulties in developing models that reflect the morphology of transport nets and the discontinuous, almost abrupt changes that occur in the real construction of such networks. The second reason is that although actual transportation networks and/or future planned road networks do shape the urbanised area, it is not possible to anticipate these without developing a much more complex decision-making model. Transport nets do not really grow organically in the same way that urban development takes place. In the case of improving the model by adding the dynamics of such networks, as we have little data on the staging of such networks, we find them hard to represent in the model other than in static fashion at the beginning or end of the simulation process. We have made little progress in this for the shape of real cities does depend on such networks but all we have been able to do is represent them statically.

Integrating real road networks and building a dynamic transportation model is a practical question of balance between dynamic artificial

networks and reality. In this context, one of the applications demonstrated in Chapter 7 has addressed this problem by assigning an attribute year to each segment of major road system to show in which year in the simulation time scale various segments are opened for public use. This is one simple way in which we can improve the model. Another potential approach will be described in the last section where we deal with future development of the model.

The Development of Scenarios

The last section in Chapter 7 demonstrates various potential scenarios for different model applications: the prediction of future development of the city —without any transformation of the transportation network in Chiang Mai, and with the chronological addition of future-planned road networks for Phitsanulok.

Consistent with observed growth, urban development in Chiang Mai is faster than in Phitsanulok city. This is probably explicable in the regional positioning of these cities since physically and socially, Chiang Mai is larger than Phitsanulok, and has a larger population with a greater variety of retail and service activities, being a more prominent centre within its territory. As indicated in the sequence of predicted images from 2003-2023 shown in Chapter 7, the urbanised area of Chiang Mai is likely to widely scatter all around the major city, broadening its sprawl along the existing transportation corridors, especially in the east and to the south of the city. The second and outer ring roads become more important as urbanised areas grow and unite separate settlements and clusters which are detached currently from the major city. In contrast, the urbanisation of Phitsanulok is much slower with the general growth trend in urban development steered toward the west and the south of the major city.

The basic potential of this kind of model is in the study of the processes of urbanisation based on constraining factors which we have embedded in a dynamic approach. We visualise such urbanisation in a sequence of images that are essentially visual extrapolations of the urbanisation of cities in

their hinterlands which we can represent in various ways from physical land use development to the emergence of the nodal regional structure. Basic scenarios can be implemented by adjusting some attributes or parameters of the model, e.g. spatial interaction, distance thresholds associated with centres, and so on. What is more, as in our second scenario, it is possible to set up other kinds of 'what-if' experiments that enable us to modify or increase various spatial features, by for example, adding a new centre in the hierarchy somewhere in the model space so that we can observe the changes in growth trends in the city, or in defining a new green space – a green belt for example – which would change the urban form in terms of densities and constraints on development.

Forecasting urbanisation according to these different scenarios provides useful information for those who are planning cities and making decisions about future change and growth. For example, urban planners attempting to design new road networks need to visually see how the city will look like in the future with respect to the implementation of their plan. They also need to explore the impact of different alternative plans and to engage in evaluating the most reasonable one and models such as DSSM are essential in enabling this to be done systematically. Property estate investors who are uncertain about investing in new projects on the edge of the city, may wish to know which side of the city centre is most attractive so that they can realise the highest benefit for the lowest costs. Facility managers may need similar information so as to prepare budgets for constructing sufficient infrastructure and services as the urbanised area continues to grow. There are many such uses that could be developed using DSSM.

3. Future Development of the Model

Developing a model such as this one is inevitably an unfinished project for a complete model is not achievable. In reality there are many other influential factors that have not been included and a lot more mechanisms

that need to be embedded into the system that are able to cause substantial changes in allocation and in urbanisation trends. To build real models, all we can expect is that we should be able to represent information that is as close as we can come to the real world. This model although built on several spatial factors and many applicable modules or algorithms, which finally produced quite complicated model mechanisms, did provide acceptable results in comparison to reality. But there are many improvements that suggest themselves and in this final section, we will identify the most important.

We will also critically assess the model in noting other potential developments in the future but first, let us sketch an outline for possible improvements which can be summarised in Figure 8.1. This illustrates the big picture for potential further expansions of DSSM. From this picture, we clearly see that DSSM needs to be merged as an element in a much larger structure of a spatial decision support system (SDSS) which forms a basis for an urban development framework. Details are shown below.

Potential Research and Development

To improve the model, let us consider in the broadest sense both the technical ideas and theoretical concepts behind the model. We can extend and thus improve the model in the following ways.

CA Enhancements: Changes to Multi-cell States at the Urban Level

Focussing only on the cellular automata module of the model, there are many ways in which this can improved. Enhancing the CA concerns modifications to its key components that are the cell space, cell states, the transition rules, and the neighbourhood patterns as well as getting to grips with the question of how deterministic and probabilistic issues are dealt with in terms of the model's methods for changing cell states.

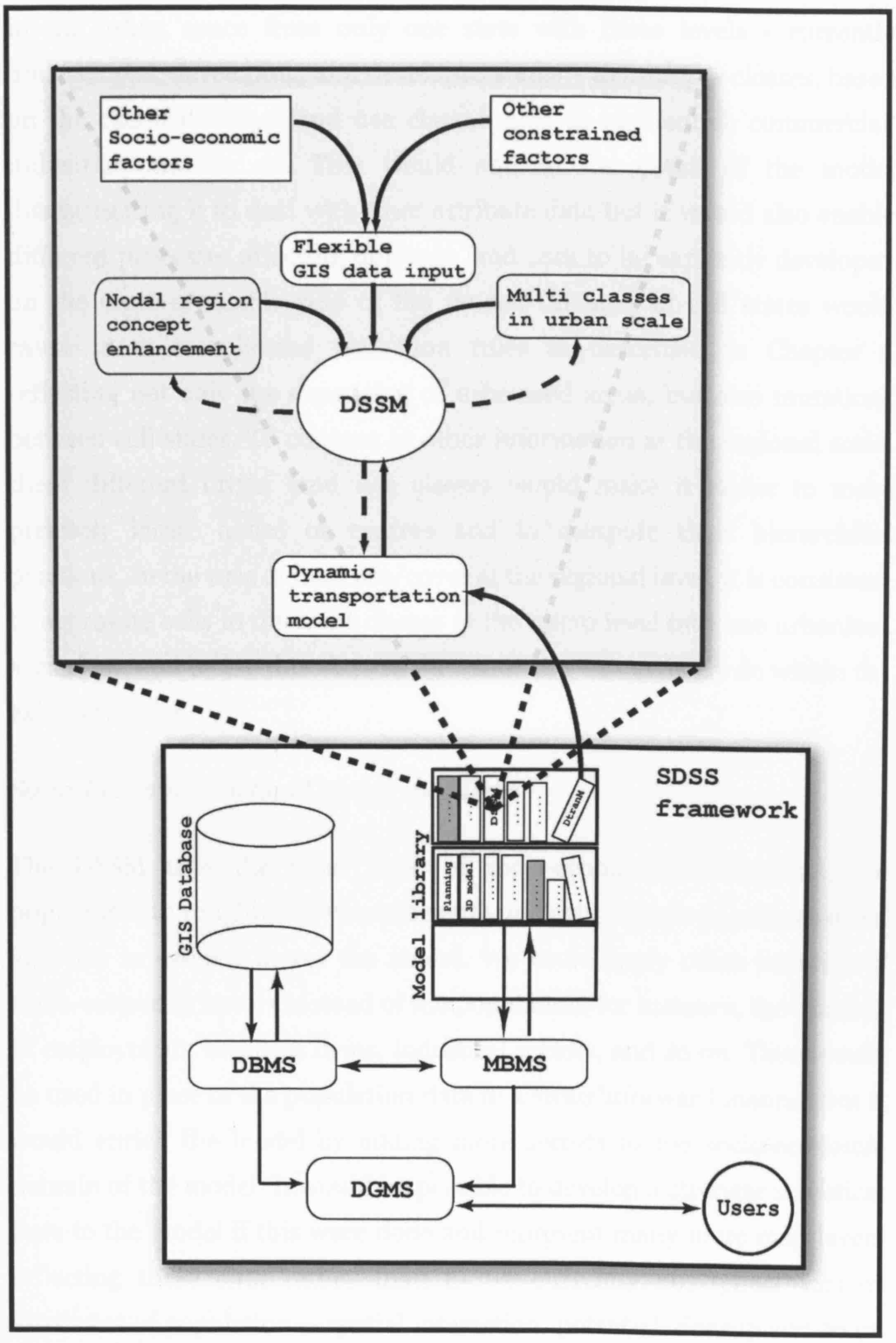


Figure 8.1 Potential Extendable Developments and an SDSS Framework

In this model, one possible alternative is to increase the number of states in the urban space from only one state with three levels - currently undeveloped, developing, and developed states – to *multiple classes*, based on the classical urban land use classes such as residential, commercial, industrial, and so on. This would add to the detail of the model disaggregating it to deal with finer attribute data but it would also enable different processes affecting different land uses to be explicitly developed on the socio-economic side of the model. Changes in cell states would cause more complicated transition rules as described in Chapter 2 reflecting not only the expansion of urbanised areas, but also mutations between cell states. To connect to other information at the regional scale, these different urban land use classes would make it easier to more precisely locate nodes or centres and to compute their hierarchical positions. In the case of land use/cover at the regional level, it is consistent to aggregate cells in different classes at the micro level into one urbanised area class, and to use this class in the same way we currently do within the existing model.

Socio-Economic Factors Driving the Model

The DSSM uses the most basic of socio-economic information, the population, to roughly but very efficiently indicate degrees of urbanisation, and this in essence drives the model. We could apply other varieties of socio-economic factors instead of the population, for instance, the number of employment, business firms, industrial sectors, and so on. These could be used in place of the population data in a straightforward manner but it would enrich the model by adding more sectors to the socio-economic domain of the model. It would be possible to develop a stronger statistical base to the model if this were done and represent many more map layers reflecting these data rather than as we currently do, reflect various attributes of population – spatial interaction, potential, density and so on using only one specific factor. It is also feasible, given the strongly modular basis of the program for this model, to extend the structure to automatically manage more than one driving factor.

Other Constraining Factors

In addition to the driving factors, one other important part of this model is in its use of constraints or as we have called them here, constraining factors. The DSSM can now support several static and dynamic constraining variables such as physical aspect (slope), transportation, land use/cover, and water bodies. In terms of the socio-economic domain, variables such as population density of each unit of land use, urban land demand, and the spatial interaction/potential through centrifugal and centripetal forces, are also handled as map layers which act as constraints. It is worth considering other factors that affect the urbanisation. For example, if attempting to limit the urban growth using ideas consistent with sustainable city concepts (Williams et al. 2000, Pinderhughes 2004), the liveable city (Girardet 2004, Dantzig and Saaty 1973), or the healthy city (Takano 2003, Werna et al. 1998), then it would be desirable to determine corresponding thematic map layers which imply new data for the model. Consequently, the model could build up visual information that demonstrates how an urban region might perform in terms of indices of sustainability, compactness, and quality of life if its urban growth is to be influenced by such normative and idealised approaches.

Enhancement to the Nodal Regional System

Each component in the nodal region concept and their relationship to one another needs to be more deeply researched so that we might characterise more intensive processes of change and improve the algorithms in the model. There are many possible developments that can extend the model in these terms. For instance, we can identify changes and extensions that might be made to: the concepts of centrality and hierarchy — applying alternative methods to locate centres and measure their position in the hierarchy; to spatial interaction—implementing supporting research to learn more about the associations between cities and towns, major urbanised areas, and their hinterlands; to diffusion—as research into the relationships between the diffusion process and various kinds of urban development such as ribbon, leapfrog, or agglomerated development; and

to diffusion again – in terms of the way this can lead to cascades of hierarchical diffusion that are also associated with the evolution of the hierarchical node distribution.

A Dynamic Transportation Model

We can also expand another element in nodal regional system which is the transportation network which we reflected on above. Predefining the year attribute of when each road segment in the model is timed in terms of its construction, and then using this data to simulate the growth and change in the network would make it possible to develop a set of algorithms to dynamically draw road cells or segments. These would then emerge as actual existing or planned road features based on other factors such as the trends in urban growth, land use/cover, density, and spatial interaction. To do so, it would be necessary to implement another extended application, the so-called dynamic transportation model, so that we might begin to show how other elements of the nodal region evolve over time.

The Future of the DSSM Model in Spatial-Decision Making and the Planning Support System Framework

In the long view, we need to consider the role of models such as DSSM not in a vacuum but within the wider context of decision support for urban and regional planning. Currently DSSM consists of three elements which include spatial data inputs, information flows within the model, and outputs; a set of algorithms based on theories of urban-regional development, which are the mechanisms used to drive the model; and a user interface that communicates with users through interactive interfaces for inputting and outputting required data and parameters. According to this categorisation, it is interesting to contemplate synthesising the DSSM within a more ambitious framework based on spatial decision support Systems (SDSS) (Densham 1991, Jankowski et al. 1997), or as it has come to be known in urban planning, the planning support system. A possible scheme is shown above in Figure 8.1.

The SDSS encompasses three major components including data base management systems (DBMS), model base management systems (MBMS), and the dialog generation management systems (DGMS) (see Malczewski 1999). For SDSS, it is necessary that the DBMS must be able to cope with spatial databases, which GIS technologies are quite adept at handling now. The MBMS are rather different systems that compose a variety of model components into a form that we can use to analyse spatial situations in such a way that we can generate decisive results that support planning and decision. The DGMS is a set of components constructed to manage the interface between the user/s and the system/s.

Figure 8.1 shows how the entire DSSM model system can be set into a model library that is organised using the MBMS. The model library contains many models from small models that require little spatial data to rather large models that have extensive data requirements and outputs, that is, from very simple to more complicated. The MBMS can retrieve any model in the library sequentially so that data can be analysed and then synthesised in order to predict spatial phenomena. In this way, we could assume that a dynamic transportation model be included in the model library. Accordingly, all the input data for the DSSM can be rearranged to be systematically stored in GIS databases and then managed using the DBMS component of the SDSS framework. Lastly, each model in the library can sustain its own user interface which should be efficiently designed to be able to connect to other models if necessary. For instance, the main interface may be organised into separate parts such as the predictive urban-regional growth model (e.g. the DSSM), an urban land use development model, routines for 3-D visualisation of model results (and of data) and so on.

Planning and implementing the SDSS for urban and regional development with such an extendable set of models provides us with a much greater set of opportunities for different urban applications over a wide range of scales of analysis along with more flexible use in sharing spatial data across all models in the system. Moreover, the outcomes returned from such models can be collected and passed in and out of other modules and

models; for example, results from DSSM can be represented visually in 3D by passing the outcomes to the 3D model. Urbanised areas predicted by DSSM can then be used in the urban land use model for example, and visualised in countless different ways. This is an interesting paradigm in which the system can support planners and decision makers in solving so-called "semi-structured decision or ill-structured decision problems" (Malczewski 1999, Crossland et al. 1995). But in the last analysis, we consider that this sort of framework would provide the model infrastructure for innovative expansions of theory and technique that would, in turn, stimulate new urban and regional models and new applications.

BIBLIOGRAPHY

Akhavein, J., Frame, W.S., and White, L.J. (2001), 'The diffusion of financial innovations: An examination of the adoption of small business credit scoring by large banking organizations'. Financial Institutions Center, The Wharton School, University of Pennsylvania.

Alhir, S.S. (2003), *Learning UML*. Cambridge: O'Reilly.

Almeida C.M., Batty, M., Nonteiro, A.M.V., Câmara, G., Soares-Filho, B.S., Cerqueira, C., and Pennachin, C.L. (2003), 'Stochastic cellular automata modeling of urban land use dynamics: empirical development and estimation'. *Computers, Environment and Urban Systems* **27**, 481-509.

Anas, A., Arnott, R., and Small, K.A. (1997), 'Urban spatial structure'. University of California at Berkeley, The University of California Transportation Center, Working Paper UCTC 357.

Armstrong, M. P. (1988), 'Temporality in spatial databases'. In *Proceedings: GIS/LIS'88* **2**, 880-889.

Barker, J. (2003), *Beginning Java Objects: From Concepts to Code*. USA: Apress.

Barnes, T.J. (2004), 'A paper related to everything but more related to local things', Forum, University of British Columbia, Department of Geography.

Barredo, J.L., Kasanko, M., McCormick, N., and Lavalle, C. (2003), 'Modelling dynamic spatial processes: simulation of urban future

scenarios through cellular automata'. *Landscape and Urban Planning* **64**, 145-160.

Batty, M., Couclelis, H., and Eichen, M. (1997), 'Urban systems as cellular automata'. *Environment and Planning B: Planning and Design*, **24**(2), 159-164.

Batty, M. (1998), 'Urban evolution on the desktop: simulation with the use of extended cellular automata'. *Environment and Planning A* **30**, 1943-1967.

Batty, M. (2003), 'Agents, cells and cities: new representational models for simulating multi-scale urban dynamics'. University College London, Centre for Advanced Spatial Analysis, CASA Working Paper 65.

Batty, M., and Xie, Y. (1997), 'Possible urban automata'. *Environment and Planning B: Planning and Design* **24**, 175-192.

Batty, M., and Xie, Y. (1994), 'From cells to cities', *Environment and Planning B: Planning and Design* **21**, 531-548.

Batty, M., and Xie, Y. (2005), 'Urban growth using cellular automata models'. *GIS, Spatial Analysis, and Modeling* edited by M.F. Goodchild, M. Batty, and D.J. Maguire, Redlands, 151-172, CA: ESRI Press.

Batty, M., Besussii, E., and Chin, N. (2003), 'Traffic, urban growth and suburban sprawl'. University College London, Centre for Advanced Spatial Analysis, CASA Working Paper 70.

Benenson, I., and Torrens P.M. (2004), *GeoSimulation Automata-based Modeling of Urban Phenomena*. West Sussex, Eng: John Wiley & Sons.

Berry, B. J. L. (1967), *Geography of Market Centers and Retail Distribution*. Foundations of Economic Geography Series, Englewood Cliffs, New Jersey: Prentice-Hall.

Bogue, D.J. (1950), 'The structure of the metropolitan community: a study of dominance and subdominance'. Ann Arbor: University of Michigan.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998), *The Unified Modeling Language User Guide*. Harrow, Eng: Addison-Wesley.

Brañas, P., Rodero, J., and Lorca, A.V. (1999) 'An empirical measure of the effect of externalities on location choice'. Loughborough University, UK.

Bryne, D. (1998), *Complexity Theory and the Social Sciences: An Introduction*. London: Routledge.

Burrough, P.A., and McDonnell, R.A. (1998), *Principles of Geographical Information Systems*. Oxford: Oxford University Press.

Bylund, E. (1960), 'Theoretical considerations regarding the distribution of settlement in inner north Sweden'. *Geografiska Annaler,* **42**, 225-231.

Castagneri, J. (1998). 'Temporal GIS explores new dimensions in time'. *GIS World* **11**(9), 48-51.

Chapin, F. S., and Weiss, S. F. (1968), 'A probabilistic model for residential growth'. *Transportation Research* **2**, 375-390.

Cheng, J., and Masser, I. (2002), 'Integrating spatially and temporally explicit decision-making process into cellular automata modelling'. In *Proceeding for 10th Annual Conference: GIS Research UK*, 3-5 April 2002.

Chrisman, N. (2002), *Exploring Geographic Information Systems*. New York: Wiley.

Christaller, W. (1966) *Central Places in Southern Germany*. Englewood Cliffs: Prentice-Hall.

Clarke, K.C., Hoppen, S., and Gaydos, L. (1997), 'A self-modifying cellular automaton model of historical urbanization in the San Francisco Bay area'. *Environment and Planning B: Planning and Design* **24**, 247-261.

Colby, C.C. (1933), 'Centrifugal and centripetal forces in urban geography'. *Annals of the Association of American Geographers* **23**, 1, 1-20.

Convery, K,M., Klopfer, S.D., and Roghair, L. (2003), 'A gap analysis of the UrBin pilot project watershed'. Virginia Tech, Conservation Management Institute, GIS & Remote Sensing Division, College of Natural Resources, CMI Publication No. CMI-GRS-03-02.

Couclelis, H. (1999). 'Space, time, geography', In *Geographical Information Systems. Volume 1 Principles and Technical Issues 2/e.* edited by P.A. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind, 29-38. New York: Wiley.

Craighead, A.C., Roberts, E.A., and Craighead, F.L. (2001), 'Bozeman pass wildlife linkage and highway safety study'. In *International Conference on Ecology and Transportation*, 24-28 September 2001.

Crossland, M.D. (1995), 'Spatial decision support systems: an overview of technology and a test of efficacy'. *Decision Support Systems* **14**, 219-235.

Dacey. M.F. (1964), 'Modified Poisson probability law for point pattern more regular than random'. *Annals of the Association of American Geographers* **54**, 559-565.

Daniel, P., and Hopkinson, M. (1989), *The Geography of Settlement.* Edinburgh: Oliver and Boyd.

Dantzig, G.B., and Saaty, T.L. (1973), *Compact City; a Plan for a Liveable Urban Environment.* San Francisco: Freeman.

Deadman, P., and Brown, R.D. (1993), 'Modelling rural resident settlement patterns with cellular automata'. *Journal of Environment Management* **37**, 147-160

DeMers, M. N. (2005), *Fundamentals of Geographic Information Systems*. 3rd ed., New York: Wiley.

Densham, P. J. (1991), 'Spatial decision support systems', In *Geographical Information Systems. Principles and Applications*, edited by D.J. Maguire, M.F. Goodchild, and D.W. Rhind, 403-412, Burnt Mill, UK: Longman.

Department of Provincial Administration, "Population information of Thailand' Available online at www.dopa.go.th.

Dijkstra, J., and Timmermans, H. (2002), 'Towards a multi-agent model for visualizing simulated user behavior to support the assessment of design performance'. *Automation in Construction* **11**, 135-145.

Egenhofer, M., and Golledge, R. (Eds) (1998), *Spatial and temporal reasoning in geographic information systems*. New York: Oxford University Press.

Fortheringham, A.S. (1983), 'A new set of spatial interaction models: the theory of competing destinations'. *Environment and Planning A* **15**, 15-56.

Fortner, A., Smith, D., and Rohrer, C. (2004), 'GIS as a prioritization and planning tool in abandoned mine reclamation'. Advanced Integration of Geospatial Technologies in mining and Reclamation, December 6-10, 2004, Atlanta, GA.

Francesca, S., and Giovanni, R. (2003), 'Land use dynamics: a cellular automata'. In *ERSA 2003 Congress*, The European Regional Science Association, University of Jyvaskyla.

Fu, S.C. (2002), 'Modelling epidemic spread using cellular automata'. *PhD Thesis*, University of Western Australia, Department of Computer Science and Software Engineering.

Gambini, R. (1966), 'A computer program for calculating lines of equilibrium between multiple centers of attraction'. Manuscript, Center of Regional Studies, University of Kansas, Lawrence.

Gardner. M. (1970), 'Mathematic games: the fantastic combinations of John Conway's new solitaire game 'Life''. *Scientific American* **223**(4), 120-123.

Gelan, A. (2003), 'Trade policy and urban-rural inequalities in LDCS: a simulation experiment with a new economic geography model'. In *International Conference on Globalisation and Development*, University of Strathclyde, Glasgow, Scotland, 10-12 September 2003.

Geertman S, and Stillwell J (Eds) (2003), *Planning Support Systems in Practice*. Heidelberg: Springer.

Girardet, H. (2004), *Cities People Planet: Livable Cities for a Sustainable World*. London: Wiley-Academy.

Hafstein, S.F., Chrobok, R., Pottmeier, A., Schreckenberg, M., and Mazur, F.C. (2004), 'A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system'. *Computer-Aided Civil and Infrastructure Engineering* **19**, 338-350.

Hägerstrand, T. (1953), *Innovation diffusion as a spatial process*, postscript and translation by Allan Pred (1967), Chicago : University of Chicago Press.

Haggett, P. (1965), *Locational Analysis in Human Geography*. London: Edward Arnold.

Haggett, P. (1975), *Geography: a Modern Synthesis*. 2nd ed., New York: Harper & Row.

Haggett, P., Cliff, A.D., and Frey, A. (1977), *Locational Models*. Bristol: Arrowsmith Ltd.

Haynes, K. E., and Fotheringham, A. S. (1984), *Gravity and Spatial Interaction Models*. Beverly Hills: Sage Publications.

Henderson, M., Messenger, D.M., and Skinner, G.W. (1999), 'A Hierarchical Regional Space Model for Contemporary China – Delineating Regional Systems and Core-Periphery Structures'. In *Proceeding for Geoinformatics'99 Conference*, University of Michigan, Ann Arbor, 20 June 1999.

Heuegger, M. (2002), 'The CA extension'. Available online at http://srf.tuwien.ac.at/lva/students/ca/index.html.

Holcombe, R.G., Pope, C., and Bast, J.L. (1999), 'Urban sprawl: pro and con'. *PERC Reports* **17**(1).

Horstmann, C.S., and Cornell, G. (2005a), *Core Java 2, Volume I: Fundamentals*. 7th ed., Englewood Cliffs: Prentice-Hall.

Horstmann, C.S., and Cornell, G. (2005b), *Core Java 2, Volume II: Advanced Features*. 7th Ed., Englewood Cliffs: Prentice-Hall.

Huff, D.L., and Jenks, G.F. (1968), 'A graphic interpretation of friction of distance in gravity models'. *Annals of the Association of American Geographers* **58**(4), 814-824.

Huff, D.L., and Lutz, J.M. (1979), 'Ireland's urban system'. *Economic Geography* **55**(3), 196-212.

Jensen, J.R. (1996), *Introductory Digital Image Processing: A Remote Sensing Perspective*. 2nd ed., Upper Saddle River, N.J.: Prentice-Hall.

Johansson, M. (2002), 'Polycentric urban structures in Sweden condition and prospects'. In *Facing ESPON* edited by C.Bengs, Nordregio Report 2002: 1. 99-118.

Jankowski, P. (1997), 'Spatial group choice: a SDSS tool for collaborative spatial decision making'. *International Journal of Geographical Information Science* **11**(6), 577-602.

Keane, R.E. (2000), 'Mapping vegetation and fuels for fire management on the Gila National Forest complex, New Mexico'. Rocky Mountain Research Station, USDA Forest Service, RMRS-GTR-46-CD.

Killough, B.D. (2003), 'A semi-empirical cellular automata model for wildfire monitoring from a geosynchronous space platform'. *Thesis*, College of William & Mary, Department of Applied Science.

King, L.J. (1962), *Statistic Analysis in Geography*. Englewood Cliffs: Prentice-Hall.

Krugman, P. A. (1998) 'The role of geography in development'. In *The Annual World Bank Conference on Development Economics*, Washington, D.C.

Langran, G., and Chrisman, N, R. (1988), 'A framework for temporal geographic information'. *Cartographica* **25**(3), 1-14.

Laura, L. (1999), *Transportation GIS*. Redlands: ESRI Press.

Li, X., and Yeh, A.G. (2000), 'Modelling sustainable urban development by integration of constrained cellular automata and GIS'. *International Journal of Geographical Information Science* **14**, 131-152.

Li, X., and Yeh, A.G. (2001), 'Calibration of cellular automata by using neural networks for the simulation of complex urban systems'. *Environment and Planning A* **33**, 1445-1462.

Liu, Y., and Phinn, S.R. (2002) 'Developing a cellular automaton model of urban growth incorporating fuzzy set approaches'. In *Proceedings of the 6th International Conference on GeoComputation*, University of Queensland, Australia, 24-26 September 2001.

Loibl, W., and Toetzer, T. (2003), 'Modeling growth and densification processes in suburban regions—simulation of landscape transition with spatial agents'. *Environment Modelling & Software* **18**, 553-563.

Longley, P.A., Goodchild, M.F., Maguire, D.J., and Rhind, D.W. (Eds)(1999), *Geographical Information Systems*. 2 Volume Set, New York: Wiley.

Malczewski, J. (1999), *GIS and Multicriteria Decision Analysis*. New York: Wiley.

Martin, D., and Wu, F. (1999), 'Empirical CA simulation from high resolution population surface', Geocomputation. Available online at http://www.geovista.psu.edu/geocomp/geocomp99/Gc99/gc_026.htm.

Masser, I., and Craglia, M. (1996) 'A comparative evaluation of GIS diffusion in local government in nine European countries'. In *GIS Diffusion: The Adoption and Use of Geographical Information Systems in Local Government in Europe*. Edited by. I. Masser, H. Campbell, and M. Craglia, Bristol, PA: Taylor & Francis, 49-66.

Möller, B. (2004), 'Geographical cost-supply analysis of forest biomass for distributed generation in Denmark'. *Paper presented at the 2nd World Biomass Conference,* May 2004, Rome.

Morrill, R.L. (1962), 'Simulation of central place patterns over time'. *Lund Studies in Geography, Series B, Human Geography*, **24**, 109-120.

Muttitanon, W., Kongthong, P. Kongkanon, C., Yoksan, S., Gonzalez, J.P., and Babazan, P. (2002), 'Spatial and temporal dynamics of dengue

hemorrhagic fever epidemics (Nakorn Pathom province, Thailand 1997-2001)'. Avaiable online at www.GISdevelopment.net.

O'Sullivan, D., and Torrens, P.M. (2000), 'Cellular models of urban systems'. In *Theoretical and Practical Issues on Cellular Automata.* S. Bandini and T. Worsh. London, Springer-Verlag. Available online at http://www.casa.ucl.ac.uk/working_paper.htm.

Omer, I. (1999), 'Demographic processes and ethnic residential segregation'. *Discrete Dynamics in Nature and Society* **3**, 171-184.

Palloni, A. (1998), 'Theories and models of diffusion in sociology'. University of Wisconsin-Madision, Centre for Demography and Ecology, Working paper 98-11.

Peuquet, D. J., and Duan, N. (1995), 'An event-based spatiotemporal data model (ESTDM) for temporal analysis of geographical data'. *International Journal of Geographical Information Systems* **9**(1), 7-24.

Pinderhughes, R. (2004), *Alternative Urban Futures: Planning for Sustainable Development in Cities throughout the World.* Lanham; Oxford: Rowman & Littlefield.

Preston, R.E. (1971), 'The structure of central place system'. *Economic Geography* **47**(2), 136-155

Raper, J., and Livingstone, D. (1995), 'Development of a geomorphological spatial model using object-oriented design'. *International Journal of Geographical Information Systems* **9**(4), 359-384.

Rayner, J.N., and Golledge, R.G. (1972), 'Spectral analysis of settlement patterns in diverse physical and economic environments'. *Environment and Planning* **4**, 347-371.

Rennard, J.P. (2000), 'Introduction to cellular automata'. Available online at http://www.rennard.org/alife/english/acgb.pdf.

Rumbaugh, J., Jacobson, I., and Booch, G. (1999), *The Unified Modeling Language Reference Manual.* Harlow: Addison-Wesley.

Schader, M., and Korthaus, A. (1998), *The Unified Modeling Language: Technical Aspects and Applications*, New York: Physical-Verlag.

Segev, A., and Shoshani, A. (1993), 'A temporal data model based on time sequences', *In Tansel et al.*, **31**, 248-270.

Siau, K., and Halpin, T. (2001), *Unified Modeling Language: Systems Analysis, Design and Development Issues.* Hershey: Idea Group Pub.

Skinner, G.W., and Henderson, M. (1999), 'A hierarchical regional space model for contemporary China – Analyzing the urban hierarchy'. In *Geoinformation '99 conference*, University of Michigan, Ann Arbor, 20 June 1999.

Takano, T. (2003), *Healthy Cities and Urban Policy Research*, London: Spon Press.

Tomlin, C.D., (1990), *Geographic Information Systems and Cartographic Modelling*. New Jersey: Prentice Hall.

Torrens, P.M. (2000), 'How cellular models of urban systems work (1. Theory)'. University College London, Centre for Advanced Spatial Analysis, CASA Working Paper 28.

Torrens, P.M. (2002). 'SprawlSim: modeling sprawling urban growth using automata-based models'. In *Agent-Based Models of Land-Use and Land-Cover Change*, edited by D. Parker, T.Berger, S. Manson, Belgium, LUCC International Project Office, 72-79.

Torrens, P.M., and O'Sullivan, D. (2000). 'Cities, cells, and complexity: developing a research agenda for urban geocomputation'. University College London, Centre for Advanced Spatial Analysis. Available online at http://www.casa.ucl.ac.uk/geosimulation/gc044.htm.

Torrens, P.M., and O'Sullivan, D. (2001), 'Cellular automata and urban simulation: where do we go from here?'. *Environment and Planning B: Planning and Design* **28**, 163-168.

Wagner, D.F. (1997), 'Cellular automata and geographic information systems'. *Environment and Planning B: Planning and Design* **24**, 219-234.

Waldrop, M.M.(1992), *Complexity the Emerging Science at the Edge of Order and Chaos*. London: Simon & Schuster.

Ward, D.P., Murray A.T., and Phin S.R., (2000), 'A stochastically constrained cellular model of urban growth'. *Computer, Environment and Urban Systems* **24**, 539-558.

Ward, M. (2002), *Beyond Chaos the Underlying Theory Behind Life, the Universe, and Everything*. New York: St. Martin's Press.

Webopedia.com (2004), www.webopedia.com.

Weimar, J.R.(2000), 'JCASim: Cellular automata simulation system'. Available online at http://www-public.tu-bs.de:8080/~weimar/jcasim/.

Werna, E., Harpham, T., Blue, I., and Goldstein, G. (1998), *Healthy City Projects in Developing Countries: an International Approach to Local Problems*. London: Earthscan.

White, R., Engelen, G., and Uijee, I. (1997), 'The use of constrained cellular automata for high-resolution modelling of urban land use-dynamics'. *Environment and Planning B: Planning and Design* **24**, 323-343.

White, R., and Engelen, G. (1997), 'Cellular automata as the basis of integrated dynamic regional modelling'. *Environment and Planning B: Planning and Design* **24**(2), 235-246.

White, R., and Engelen, G. (2000), 'High-resolution integrated modeling of the spatial dynamics of urban and regional systems'. *Computers, Environment and Urban Systems* **24**, 383-400.

Wiener, R., and Pinson, L.J. (2000). *Fundamentals of OOP and Data Structures in Java*. New York: Cambridge University Press.

Wilensky, U. (2003). 'NetLogo CA Stochastic model'. Available online at http://ccl.northwestern.edu/netlogo/models/CAStochastic, Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.

Williams, K., Burton, E., and Jenks, M. (2000), *Achieving Sustainable Urban Form*. London: E & FN Spon.

Wolfram, S. (1984), 'Cellular automata as models of complexity'. *Nature* **311**, 419-424.

Wolfram, S. (2002), *A New Kind of Science*. Champaign, IL: Wolfram Media, Inc.

Wood, J. 2002, *Java Programming for Spatial Sciences*. London: Taylor&Francis.

Worboys, M. F. (1992), 'A model for spatio-temporal information'. In *Proceedings: the 5th International Symposium on Spatial Data Handling* **2**, 602-611.

Wu, F. (1998), 'Simland: a prototype to simuate land conversion though the integrated GIS and CA with AHP-derived transition rules'. *International Journal of Geographical Information Science* **12**, 63-82.

Wu, F. (2002), 'Calibration of stochastic cellular automata: the application to rural-urban land conversions'. *International Journal of Geographic Information Science* **16**, 8, 795-818.

Wu, F., and Martin, D. (2002), 'Urban expansion simulation of Southeast England using population surface modelling and cellular automata'. *Environment and Planning A* **34**, 1855-1876.

Wu, F., and Webster, J.C. (2000), 'Simulating artificial cities in a GIS environment: urban growth under alternative regulation regimes'. *International Journal of Geographical Information Science* **14**, 625-648.

Wuu, G., and Dayal, U. (1992), 'A uniform model for temporal object-oriented databases'. In *Proceedings of the International Conference on Data Engineering (Tempe, AZ)*, 584-593.

Xie, Y. (1994), 'Analytical models and algorithms for cellular urban dynamics'. *PhD Thesis*, University of New York at Buffalo, The Graduate School, Buffalo, NY.

Yeh, A.G., and Li, X. (2001), 'A constrained CA model for the simulation and planning of sustainable urban forms by using GIS'. *Environment and Planning B: Planning and Design* **28**, 733-753.

Yuan, M. (1995), 'Temporal GIS and spatio-temporal modeling'. Available online at http://ncgia.ucsb.edu/conf/SANTA_FE_CD-ROM/sf_papers/yuan_may/may.html.

Zipf, G.K. (1949), *Human Behavior and the Principle of Least Effort: an Introduction to Human Ecology*. Cambridge: Addison-Wesley.

Appendix I
Interface of DSSM Version 1.0

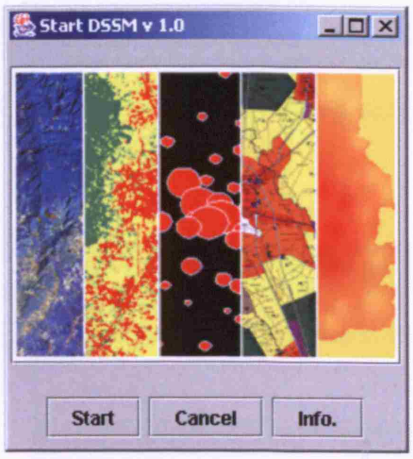
The following is a brief description of the interface of the first version of DSSM. Generally, there are four main windows including: the CA-DSM window, the dynamic spatial model window, parameters setting window and the hints&log window. The CA-DSM window separately represents CA- and node-space; the CA and node simulations are performed here. The dynamic spatial model window is used to display both static and dynamic map layers as described earlier in several chapters. The parameter window controls all processes in the model. This window allow users to easily change model parameters to suit their applications. The hints&log window give users information about the various steps for setting parameters, loading map layers and so on.

The DSSM v 1.0 and a set of example data can be download at:

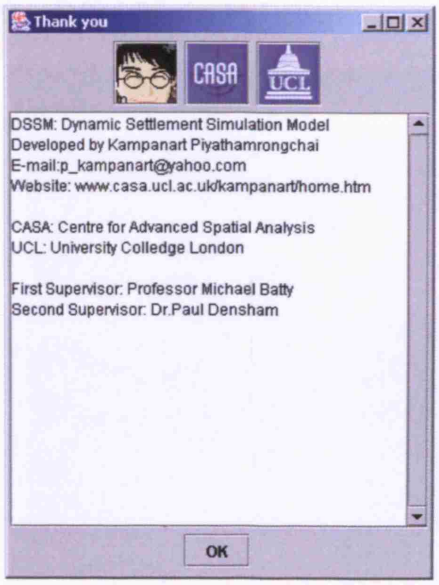
www.casa.ucl.ac.uk/kampanart/DSSM/resources.htm
File name: 'dssm.zip'

To unzip, you may need to install file compression software (Winzip or Winrar for example) on your computer. The zip file consist of two folders: 'bin' and 'data'. The bin folder contains a file, 'dssm.exe', which can be executed to begin the DSSM model. A set of example map data are kept in the data folder. Users can trial by playing around with this data.

1. Start Program

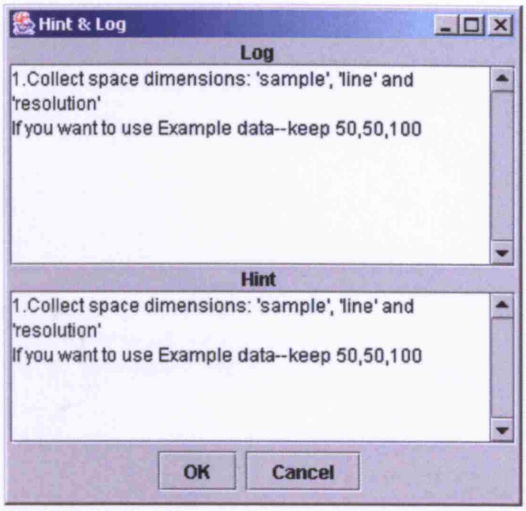


2. Software Information

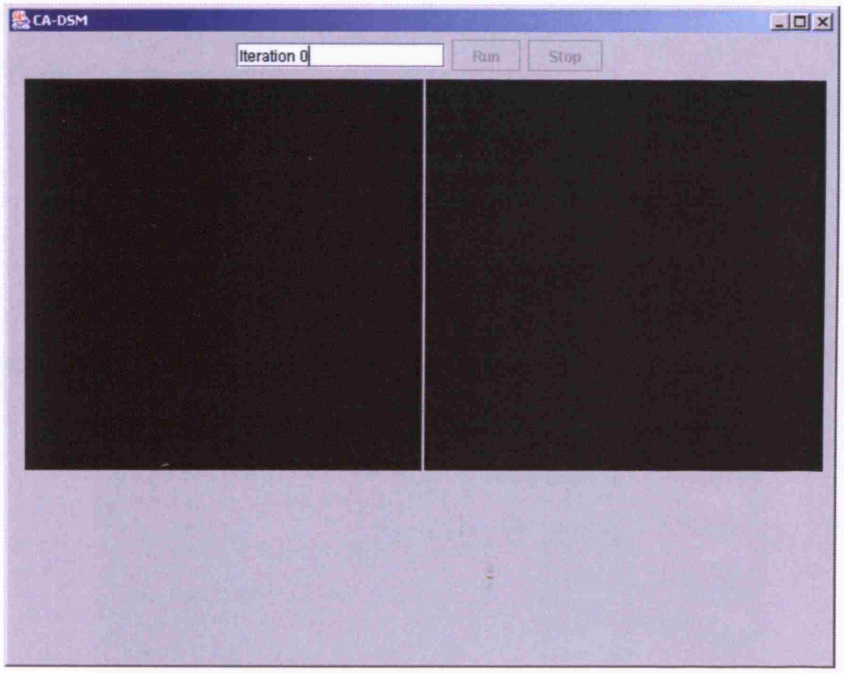


3. Dimension Setting Window

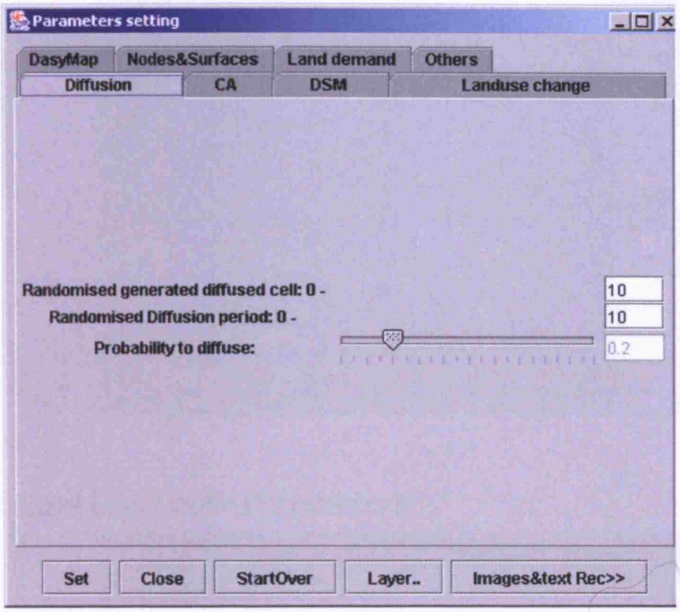
4. Hint& Log Window



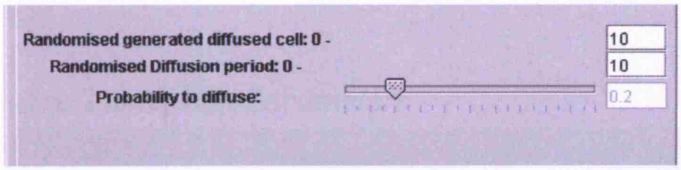
5. CA-DSM Main Window



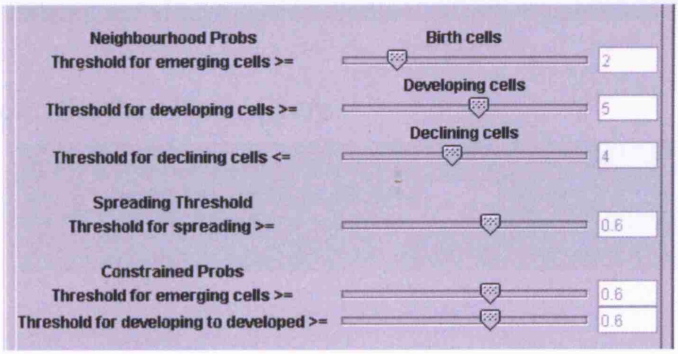
6. Parameter Setting



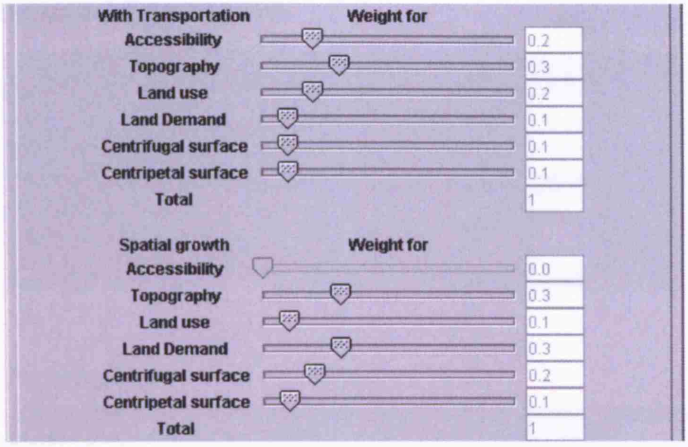
6.1 Diffusion Parameters



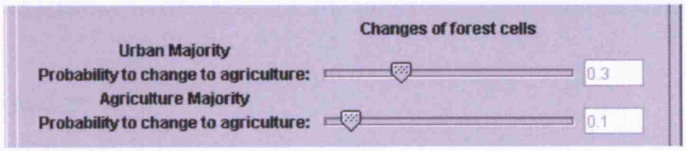
6.2 Cellular Automata Parameters



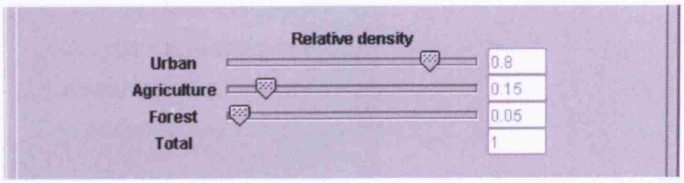
6.3 Dynamic Spatial Model Parameters



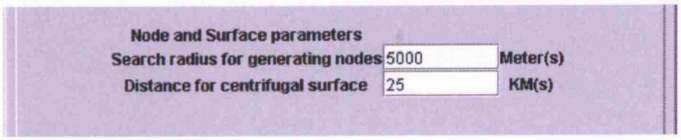
6.4 Land Use/Cover Parameters



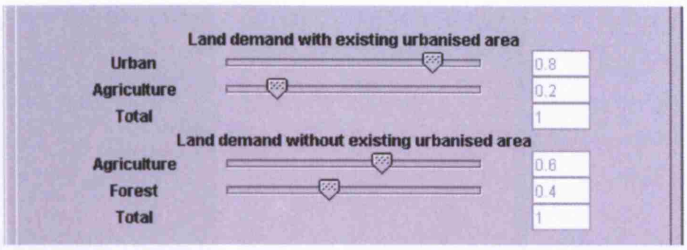
6.5 Dasymetric Population Parameters



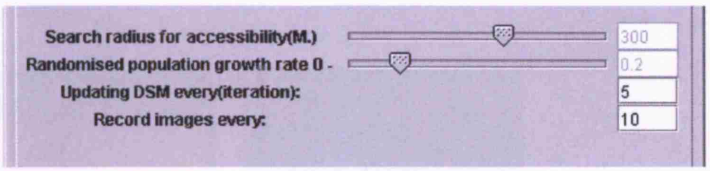
6.6 Node and Surface Parameters



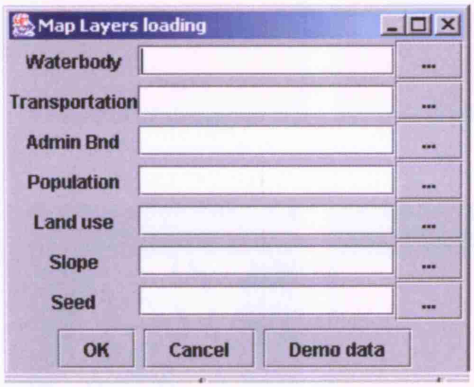
6.7 Land Demand Parameters



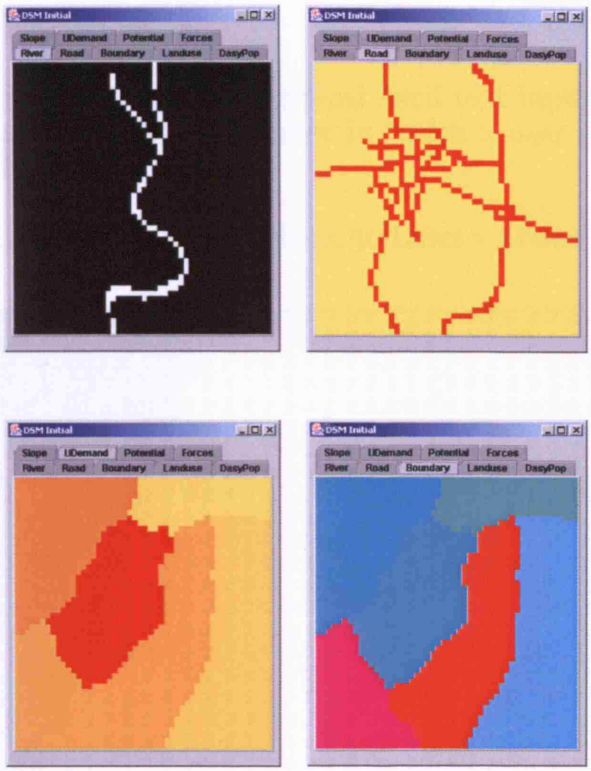
6.8 Other Parameters



7. Map Loader



8. Dynamic Spatial Model Window



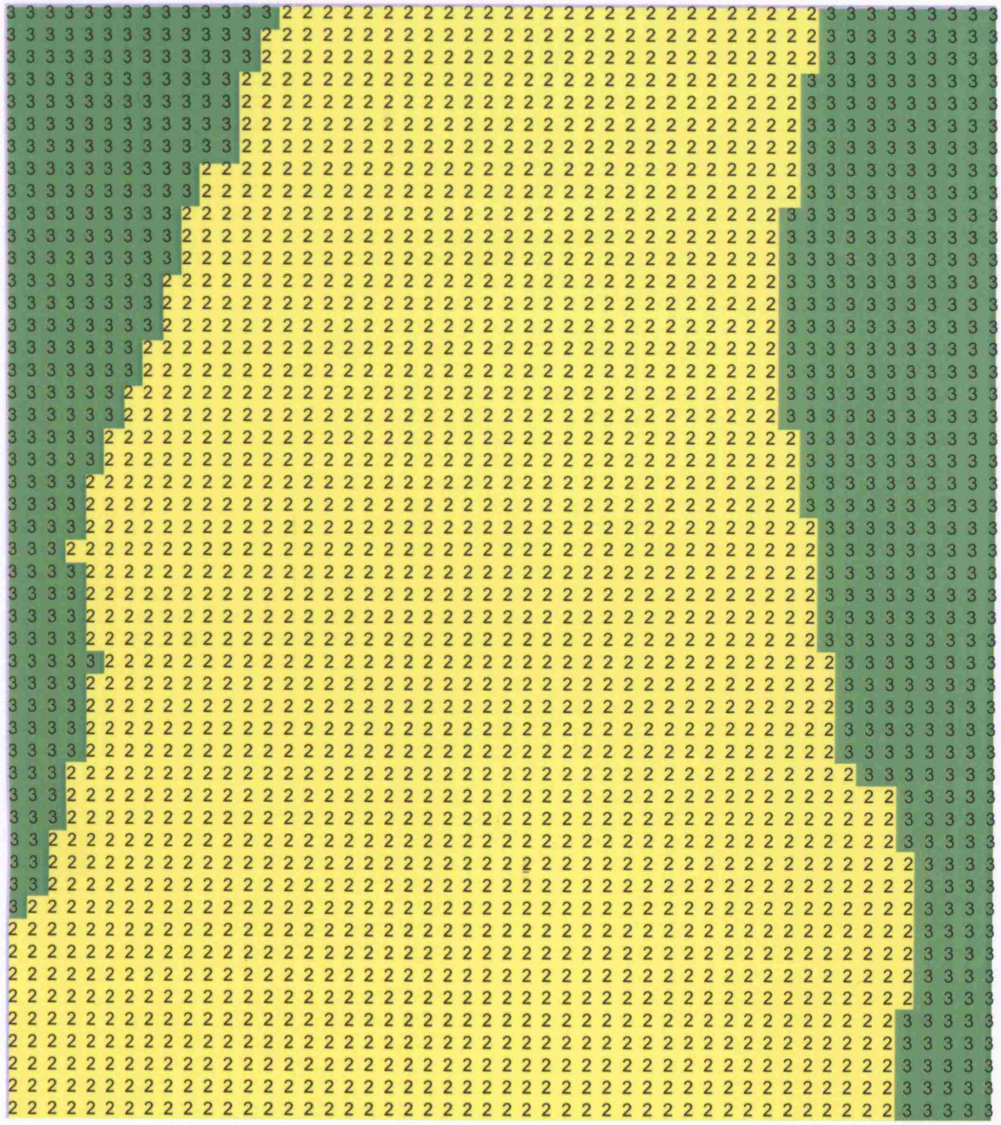
Appendix II
Ascii Text File Format

This is an example of a typical Ascii text input for a dynamic map which indicates the simple manner in which a user can define the raster-based data layer

;

; File dimension 50 samples x 50 Lines x 1 band

;



;