Computational Explorations of Life

Thesis by

Johan Chu

In Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy

California Institute of Technology
Pasadena, California

1999
(Submitted December 21, 1998)

Acknowledgments

First, I would like to thank my advisors, Chris Adami and Mike Cross. Without their patience and support, I would have been able to do nothing. They bore with me when I tarried, and were there to help me when I needed help most. I will always appreciate the guidance and teaching they have given me.

Indeed, I have been privileged to enjoy the attention and guidance of many teachers throughout my education. My first teachers in England, my teachers in Canada and the United States, and especially my teachers in Korea (who communicated with me and cared for me even though I had obvious and severe problems communicating in Korean) all helped me on my current path. I remember walking into my undergraduate advisor Dr. Hae-woong Lee's office in Korea and asking, "Should I go to the States to study?" The immediate reply was, "Of course, and to a good school." The standing joke is that he wanted to get rid of me quickly, but I know better. I grew immensely from being at Caltech, and I thank Professor Lee and Professor Byung-yoon Kim for their help getting me here.

These last six years would have been unbearable without my friends. The first few years at Caltech were remarkable for the many members of the Caltech Korean community who opened their hearts and lives to me. Among them, Dr. Namkyoo Park made sure I never lacked for a ride or for someone to have dinner with while he was here. I appreciate all the friendship I've received. Other friends from the Los Angeles area and visiting from Korea have kept me sane at Caltech and I thank them for it. Although, I rather suspect I might have graduated sooner if I had refused some fun! I am also thankful for the friendship of my friends in Korea, who I've too often distracted from qualifying and final exams, girlfriends, business and even the army to satisfy my selfish need for their company.

I've already thanked my teachers and friends, but the greatest of both to me have been my mother and father. I learnt English reading the Bible at my mother's knee, and my parents' love stays with me everywhere I go. I have received so much love from them and from all my extended family: my sister, aunts, uncles, grandparents and cousins. I carry around a huge burden of debt for their love. I know that with their love and faith, I am given great power and an even greater obligation. I thank them all with my whole being.

Finally, I thank God who created this world we strive to explore, and who created me. I am overwhelmed by His love. I know I will have lived well if I can reflect on others but a tiny portion of the love He has given me.

Dedication

This work is dedicated to my grandparents.

Abstract

Artificial Life, the creation and study of man-made systems that exhibit the characteristics of life, is a young and still emerging field. The goals of Artificial Life are two-fold; to gain knowledge about and from biology. The Artificial Life system sanda, which extends upon previous systems tierra and avida, was designed to help in investigations into the statistical nature of evolution. As such, it is a model of the simplest living, evolving organisms. Experiments involving tierra, avida, and sanda were the inspiration for investigations into the causes of apparently scale-free dynamics found in these systems. These investigations lead to identification of a branching process that explains the scale-free dynamics of not only these Artificial Life systems, but also those manifested in the taxon rank-frequency distributions of biology and in the size distributions of avalanches in "self-organized critical" sandpile models. This branching process can quantitatively predict—with no free parameters—the pattern of the observed distributions, including their divergence from a true power law. Further, the branching process gives insight into the universal mechanisms involved in the creation of, and divergence from, scale-free dynamics in these systems, including a definition of order and control parameters reminiscent of those from second-order phase transitions in statistical physics.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Why Artificial Life?

Ever since Darwin [16] proposed his grand theory of evolution, biologists and others have been asking, "What if?" What if the temperature on Earth had been a hundred degrees hotter or cooler throughout its history? What if the atmosphere had more nitrogen and less oxygen? What if no global changes had occurred to (perhaps) wipe out the dinosaurs? Was the creation of life on Earth an impossibly lucky accident? How about the creation of life as we know it? Could other forms of life—perhaps not based on carbon or on DNA—have arisen given different rolls of the "evolutionary dice"?

An even more fundamental question, asked since man first recognized the difference between living creatures and inanimate objects, is "What is Life?" What makes a thing alive and distinguishes it from things without life? A multitude of definitions have been proposed; physiological (centered on functions performed by organisms), metabolic (centered on the exchange of materials between the organism and its surroundings), biochemical (living organisms are characterized by storage of genetic information in nucleic acid molecules), genetic (characterized by evolution, not necessarily based on nucleic acids), thermodynamic (characterized by an ability to maintain low levels of entropy), etc. However, none of these definitions are completely satisfactory.

One of the major roadblocks to defining, and studying, the essence of life, is that we can not deconstruct it, separate it into its parts as we do entities in Physics or Chemistry. Once a living system is separated into its parts, the individual parts no longer have life. A study of the individual parts can only tell us so much about the whole. While we can learn much about the mechanisms of life from the structure of DNA, it is hard to believe that DNA, by itself, is alive. Life is a property of a living system, not of its parts.

Artificial Life, the creation and study of man-made systems which exhibit characteristics of life, offers an avenue of investigation into essential properties of life. Because these systems are man-made, we understand the workings of all the parts, and this offers us more hope of understanding what happens when we put all the parts together. By creating systems which mimic life, we hope to learn more about what life is.

Such systems also give us the fantastic ability to *experiment* with life, with evolution. We can set and reset the parameters of the system, and run it again and again, until we gain a true, fundamental, and quantitative understanding of the processes involved. To these ends, we need a system simple enough for us to understand and calibrate its workings, and fast enough to allow repeated evolutionary trials so as to gain a *statistical* picture of life and evolution, but still sophisticated enough to model the processes at work in natural evolution. The Artificial Life systems tierra [32, 33], avida [4], and my own system sanda are attempts at such a system. Personally, sanda has acted as a source of ideas, suggesting avenues of research by its tantalizing parallels to nature, while serving as a lab bench, giving me a system where I could run countless numbers of trials to gain statistical data to verify an idea.

1.2 Scale-free Dynamics

One striking statistical feature observed in tierra, avida, and sanda runs is the appearance of seemingly scale-free dynamics—manifested in near power law distributions—in genotype abundance distributions (analogous to species distributions in nature) [3, 6]. Various mechanisms were proposed to explain the appearance of scale-free dynamics, and also the variance from such dynamics in some trials. The power laws were compared to those observed in self-organized criticality (SOC), natural evolution, and random walks, while the deviance from power law was mostly attributed to finite-size effects.

Near power law distributions are found in the statistics of a wide variety of systems from diverse disciplines, including demography, taxonomy, geophysics, and nuclear physics. Per Bak [9] proposed that many of these power laws result from a single underlying process, which he termed self-organized criticality (SOC). The paradigm for SOC is the avalanche behaviour of sandpile models, which under certain conditions exhibit scale-free dynamics. Similarly, for taxonomic levels higher than species, the rank-abundance distributions of number of subtaxa per taxon approximate power laws. Yule [47] proposed a branching process model to explain these distributions at the generic level. He recognized that naturally observed distributions diverged from the power law predicted for equilibrium distributions by his theory and hypothesized that this deviation was caused by a finite-time effect.

In Chapter 6, I present a branching process model that explains the observed genotype abundance distributions in sanda and the near power law distributions in the other systems mentioned above. The model's formulation was inspired by observation of many sanda runs. The model was tested on various versions of sanda, some of which had simpler dynamics than others. The experiments on sanda versions with simpler dynamics allowed for better observation of the factors that determined the large-scale dynamics of the system, unobstructed by secondary perturbations. The branching process model predicts not only the power law distribution of genotypic abundances observed in some runs, but also predicts the divergences from power law observed in other runs. It gives a quantitative prediction of the shape of the distributions with *no free parameters.* Furthermore, the model also explains the seemingly scale-free dynamics observed in SOC's sandpile models, and in the rank-frequency distributions of taxons in nature. For these systems also, the branching process model allows quantitative prediction of distribution patterns with no free parameters. For the SOC sandpile models, I find that the "self-tuning" results from arbitrarily enforcing conditions on the order parameter, and show how relaxing these conditions change the observed dynamics. While for natural evolution, I find the deviance from power law is not due to a disequilibration effect (as Yule proposed), but rather results from fundamental properties of the observed taxa.

This research was inspired by observations of interesting statistical behaviour in artificial living systems. Its initial progress was two-pronged; finding an analytical model which could explain the dynamics leading to such distributions and simplifying the system until such dynamics could clearly be seen. The new understanding that these investigations yielded found broad application to such systems as seemingly disparate as the distribution of avalanche sizes in sandpiles and the distribution of taxon sizes in biology. Artificial life has served well as both muse and medium.

1.3 Outline of Thesis

In Chapter 2, I introduce the emerging field of Artificial Life and describe the Artificial Life system sanda. I present a test of sanda's validity in an application to propagation of information in self-replicating genetic systems in Chapter 3. Chapter 4 is a short chapter dealing with appropriate methods for binning histograms of events with exponential or power law probability distributions. Chapter 5 details an extension to the Bak-Tang-Wiesenfeld sandpile model that allows us to study characteristics of the model in previously ignored regimes. Finally, Chapter 6 discusses the scale-free dynamics of a branching process and its applications to statistics found in sanda, natural evolution, and sandpile models. Chapter 3 is from a paper presented at *Artificial Life V* held May 1996 in Nara, Japan [14]. Chapters 4-6 contain parts of papers to be submitted to Nature, Physical Review Letters, and Physical Review E.

Chapter 2 The Artificial Life System

Sanda

2.1 Artificial Life

Artificial Life is the creation and study of man-made systems which have characteristics of life. Characteristics modelled in current Artificial Life systems include genetic evolution and coevolution, flocking behaviour, locomotion, language acquisition, the spread of disease, and many others. Despite this diverse subject matter, almost all Artificial Life systems share an evolutionary approach to learning.

Artificial Life systems can be largely divided into two types; those that seek to gain a better understanding of natural life, and those that attempt to apply insights from biology to other fields of endeavour—engineering, recreation, etc. Artificial Life systems can also be classified by whether they emphasize the development and coevolution of populations, or the evolution of individuals. Yet another classification scheme involves the media used to implement the system—wetware (real biological components from natural systems such as RNA, DNA, proteins), hardware (robots, etc.), or software (simulations, computer code). Fig. 2.1 shows a sample of currently existing Artificial Life systems and their classifications. Refs. [4, 26] are recommended as more detailed introductions to Artificial Life systems.

Webb [45] and co-workers built a robotic cricket from Lego^{TM} parts to gain insight into the mechanisms used by real crickets to locate prospective mates by listening to their song. The robot was equipped with artificial ears and neurons, and actuators connecting its "brain"'s output to its left and right wheels. Differences in signal strength to each wheel caused the robot to move in a curved path. The artificial ears were built to mimic actual cricket ears and their sensitivity to phase differences between the sound waves arriving at each ear. Webb tested algorithms for phonolo-

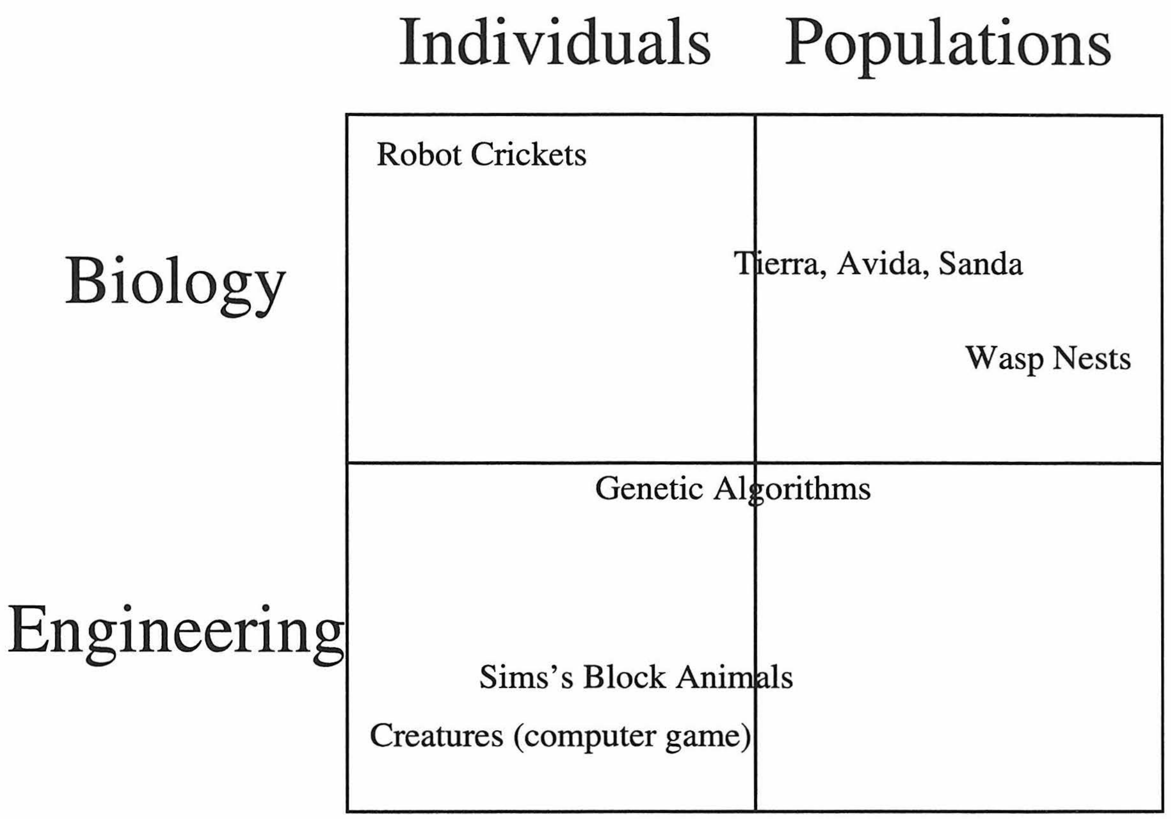


Figure 2.1: Various Artificial Life systems. Only a few of the many currently existing Artificial Life systems are shown. The characterizations are based purely on my own judgment. The systems shown are: robot crickets [45] tierra [32], avida [4], sanda [14], wasp nests [41], Creatures^{TM} [21], genetic algorithms [24], and Karl Sims's evolving animals [35].

cation. She found that a simple algorithm succeeded in recreating the movement patterns of crickets. Robots using the algorithm were able to distinguish between different songs when many were played simultaneously. Furthermore, this algorithm depended on breaks between chirps for phonolocation, suggesting an explanation for the distinctive chirping of crickets in nature.

Theraulaz and Bonabeau [41] developed a simple model of randomly moving artificial agents to test the possibility that complex structures such as wasp nests could be built without inter-agent interaction. Each agent in their model could sense only the local structure near it, and could only deposit elementary building blocks. With simple rules based purely on local conditions for depositing building, a swarm of these agents built structures with shapes strikingly similar to those of real wasp nests.

Creatures^{TM} [21] is a computer game where players raise, teach, and breed computer "animals." The animals' behaviours are determined by neural nets, and can be modified by user-supplied stimuli.

Genetic algorithms [24] have been used extensively in optimization problems. The problem to be solved is encoded into "chromosomes"—bit strings representing candidate solutions to the problem. A population of these chromosomes is simulated. With successive selection (choosing the chromosomes which currently best approach a solution to the problem) and mutation (random bit flips in chromosomes, mixing of segments of two chromosomes to create a new chromosome), the system tends to evolve chromosomes which approach solutions of the problem. In addition to pure engineering uses, genetic algorithms have also been used as models of biological and social systems.

Karl Sims created systems where both the shape and behaviour of block creatures were allowed to evolve, and creatures were selected for being able to perform physical feats, such as swimming, crawling, or object manipulation. In these systems, creatures had "genotypes" (much like a genetic algorithm's chromosomes) containing information on both their morphology and behaviour. His work differs from classical genetic algorithms in that he introduced direct competition between different members of the population as a means of selecting the fittest. For example, in one of

Sims's systems, creatures attempt to gain control of a cube placed in the middle of their simulated world. The creatures are paired off and duel for the cube, and the results determine their fitness. The fitness of a creature is determined not only by its own morphology and behaviour, but that of the other creatures existing with it.

Although the proliferation of Artificial Life systems, and the corresponding recognition of Artificial Life as a distinct field of study, is a recent (since ~1988) development, theoretical explorations of Artificial Life character have a longer history dating back to Von Neumann [43] and his thinking on self-replicating cellular automata. NASA even studied self-replicating robots as a means of mining the moon [25]. However, it has only been recently that we have gained sufficient computing power and a good enough understanding of the base-level mechanisms of evolution to make non-trivial Artificial Life a possibility. The need for solutions to complex problems (some of which have already been solved by nature) has also stimulated recent Artificial Life research.

Artificial intelligence, to which Artificial Life is often compared, was widely touted in its early days as a quick and easy road to making "intelligent" computers, as conscious as—and smarter than!— ourselves. This was, of course, too much to ask. I believe Artificial Life will have many applications in optimizing characteristics and behaviours of complex systems. However, my greater hope is that Artificial Life systems will become a bona fide tool in a biologist's toolbox, one that allows the correct degree of abstraction for any particular problem; enough to make the problem tractable, but not enough to change its character.

Modern biology, much like modern physics or chemistry, has become centered on deconstruction of the whole into its components. Molecular biology has pushed the likes of anthropology, botany, and zoology from the front of the stage. Advances in deconstructive biology have led us much closer to an understanding of the biochemical mechanisms that life uses in the lifeforms present on Earth. However, with the recognition that such mechanisms can not explain all of life's characteristics, more holistic disciplines (e.g., ecology, large-scale theories of evolution and extinction, systematic neurobiology, complexity) which seek to understand how the individual parts and mechanisms work together to create the dynamics of the system as a whole—what we call life—are gaining an increasing share of attention. Artificial Life is a discipline at the crossroads of biology, physics, computer science, and all the engineering fields concerned with the artificial media man has created. It ultimately seeks to explore the essence of life.

2.2 Overview of Sanda

Sanda, from the Korean for 'to live', is a software system designed to emulate and study the evolution of populations of self-replicating code. In the classification of Artificial Life systems proposed in the previous chapter, sanda, like avida and tierra, is more a biological tool than a biology-based application. Although, with improvements in the software and even faster computers, the system may eventually become powerful enough at creating new strategies and algorithms through evolution that it will become an algorithm-creating application of biological principles. Sanda can also be used to study the evolution of characteristics in individual creatures, but its greatest strength lies in investigating large-scale population effects of evolution; broad statistical laws which hold true in all replicating, competing, mutating, *evolving* systems.

Sanda is the third generation in the tierra family line. Tierra, in which strings of self-replicating assembly-like code proved *robust under mutation* was the first of its kind. Avida added a spatial structure to tierra, creating a physically more realistic system in which the dynamics of diffusion and information propagation could be observed. Sanda expands avida's boundaries by allowing simulations with unprecedented size, or the possibility of easily running multiple, related simulations. A larger system size yields better statistics and easier observation of spatial effects, and—perhaps most importantly—allows the possibility of observing a system evolving always away from equilibrium.

Sanda was written first in C, and then completely rewritten in C++ to allow easy extension to the base system. It runs on a wide range of machines, but was designed and optimized primarily for use on the Intel Paragon, a massively-parallel MIMD architecture supercomputer.

Like avida, sanda works with a population of strings of code residing on an $M \times N$ lattice with periodic boundary conditions. Each lattice point can hold at most one string. Each string consists of a sequence of instructions from a user-defined set. These instructions, which resemble modern assembly code and can be executed on a virtual CPU, are designed to allow self-replication. The set of instructions used is capable of universal computation.

Each string has its own CPU which executes its instructions in order. A string self-replicates by executing instructions which cause it to allocate memory for its child, copy its own instructions one by one into this new space, and then divide the child from itself and place it in an adjacent grid spot. The child then is provided with its own virtual CPU to execute its instructions. When a string replicates, it places its child in one of the sites in its 9-site neighbourhood (Fig. 2.2), replacing any string which may have been there. How the site to be replaced is chosen can be defined by the user. See the section on replication and selection below for more information.

It should be noted that this birth process, and indeed all interactions between strings, are local processes in which only strings adjacent to each other on the grid may affect each other directly. This is important as it both supplies the structure needed for studies of spatial characteristics of populations of self-replicating strings of code, and allows longer relaxation times – making possible studies of the equilibration processes of such systems and their nonequilibrium behavior.

This process of self-replication is subject to mutations or errors which may lead to offspring different from the original string and in most cases non-viable (i.e., not capable of self-replication). Of the many possible ways to implement mutations, we have mainly used copy errors. That is, every time a string copies an instruction there is a finite chance that instead of faithfully copying the instruction, it will write a randomly chosen one. This chance of mutation is implemented as a per-instruction mutation rate γ—the probability of copy-error per instruction copied. A mutation rate γ for a string of length ℓ will therefore lead to a fidelity (probability of the copied

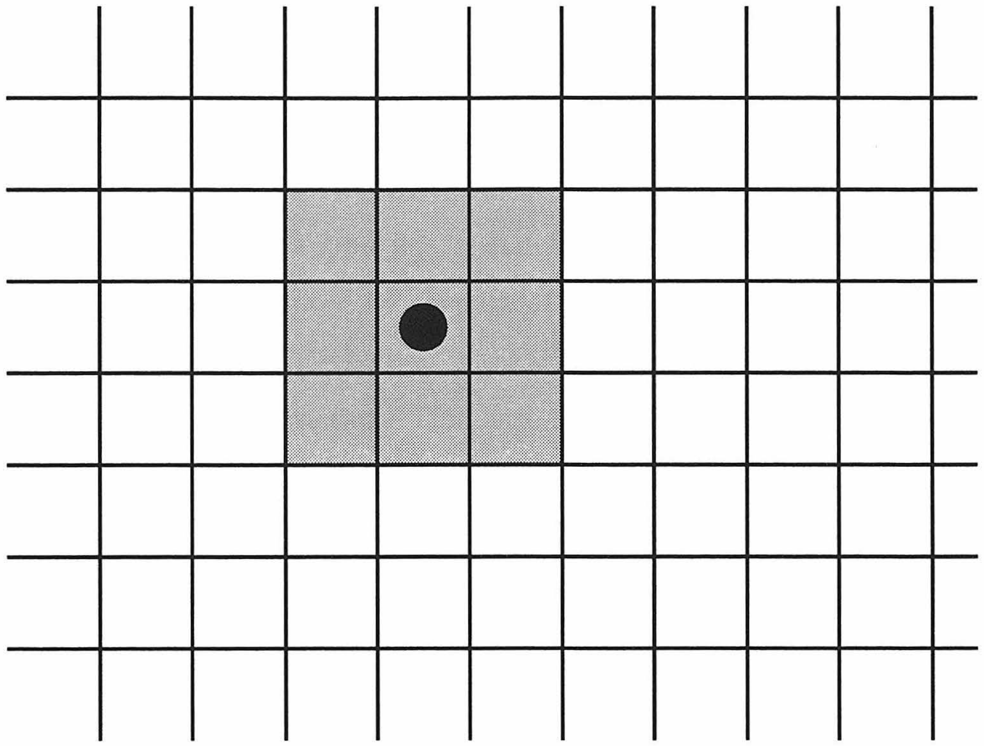


Figure 2.2: Sanda grid. The organisms live on an Euclidean grid, one organism to a site. When an organism replicates, its daughter replaces one of the organisms in its 9-site neighbourhood. (If the organism marked by a black dot replicates, its daughter replaces one of the organisms at a gray site.) Which criteria are used in choosing the neighbour to be replaced affects the dynamics of the system.

string being identical to the original) $F = (1-\gamma)^{\ell}$. Mutations allow us to evolve a very heterogeneous population from an initially homogeneous one. The resulting evolution, coevolution, speciation, etc. have been and continue to be studied [33, 2, 3, 1, 5, 6].

The factors which decide whether one particular sequence of instructions (or genotype) will increase or decrease in number are the rate at which it replicates and the rate that it is replaced at. In this model, the latter is genotype independent. Accordingly, we define the former (i.e., its average replication rate) as the genotype's fitness. In other words, fitness is equal to the inverse of the time required to reproduce (gestation or replication time).

To consistently define a replication rate, it is necessary to define a unit of time. Previously, in tierra and avida, time has been defined in terms of instructions executed for the whole population (scaled by the size of the population in the case of avida). In sanda, we define a physical time by stipulating that it takes a certain finite time for a cell to execute an instruction. This base execution time may vary for different instructions—certain instructions can be arbitrarily made more time-consuming and "expensive" for creatures to execute. The *actual* time a cell takes to execute a certain instruction is then increased or decreased by changing its *demerit.* Initially, each cell is assigned an demerit near unity, $e = (1 \pm \eta)$, where η represents a small stochastic component. In summary, the time it takes a cell to execute a series of instructions depends on the number of instructions, the particular instructions executed, and the cell's demerit.

Self-replication consists of the execution of a certain series of instructions by the cell. Thus, the fitness of the cell (and its respective genotype) is just the rate at which this is accomplished and depends explicitly on the cell's demerit. We can assign better (or worse) demerit values to cells which contain certain instructions or which manage to carry out certain operations on their CPU register values. This allows us to influence the system's evolution so as to evolve strings which carry out allocated tasks. A cell that manages a user-defined task can be assigned a better demerit for accomplishing it. Such cells, by virtue of their higher replication rate, would then have an evolutionary advantage over other cells and force them into extinction. At the same time, the discovery that led to the better demerit is propagated throughout the population and effectively frozen into the genome.

In addition to the introduction of a real time, sanda differs from its predecessors in its parallel emulation algorithm. Instead of using a block time-slicing algorithm to simulate multiple virtual CPUs, sanda uses a localized queuing system which allows perfect simulation of parallelism.

Finally, sanda was written to run on both parallel processors and single processor machines. Therefore, it is possible, using parallel computers, to have very large populations of strings coevolving. This permits studies of extended spatial properties of these systems of self-replicating strings and holds promise of allowing us to study them away from equilibrium.

The following sections contain more technical information about sanda. The reader is advised to read Ref. [4] for a more extensive treatment of the closely-related avida system.

2.3 The Grid

The grid is a $N \times M$ Euclidean lattice with periodic boundary conditions. Each lattice site (*cell*) may have at most one CPU attached to it. Each cell has its own time value. This time corresponds to the system's physical time (see section below on parallel emulation).

2.4 CPU Structure

A CPU (or creature) is attached to a cell (its grid location), and has components as shown in Table 2.1. When a CPU replicates, its daughter CPU (which contains initialized values of all structure members) replaces one of its neighbours. Which neighbour is replaced is explained in detail below.

Type	Component	Explanation
Status	age	current age of CPU (how long since CPU was created)
	last divide time	when did CPU last replicate
	genotype	what is this CPU's genotype name (changed if the CPU code undergoes mutation)
	demerit	How fast (relatively) does this CPU execute instructions? Initially set to $1 \pm \eta$, where η is a small, positive random number. Executing desired tasks (see below) will give a creature lower demerit, and thus a faster replication rate.
	facing	Used in certain instructions which allow interaction with neighbouring CPUs.
Physical Structure	stacks	One or more stacks.
	stack pointers	Pointers for the stacks.
	stack number	Number of stack we are currently using.
	registers	Three or more registers.
	input buffer	Buffer for receiving input from the user or other CPUs.
	output buffer	Buffer for output to the user or other CPUs.
	input pointer	Pointer to current location in input buffer.
	output pointer	Pointer to current location in output buffer.
	code	The CPU's string of instructions.

Table 2.1: CPU structure.

2.5 Genotypes and the Instruction Set

A CPU's code (or *genome*) largely determines the replication rate of the CPU. The code is a string of instructions from a user-defined instruction set. If two CPUs have the same genome, they belong to the same *genotype*. The instruction set consists of assembly language-like instructions designed to be computationally both powerful and simple, and robust under mutations. Both the function of each instruction and the composition of the instruction set can be easily modified by the user.

A sample instruction set is shown in Tables 2.2-2.6. A major difference between the instruction sets commonly used in sanda and computer assembly language is the use of nops as arguments to other instructions. For example,

```
inc
nop-A
```

would cause an increment of the AX register value, while

```
inc
nop-B
```

would cause an increment of the BX register value. This kind of addressing obviously only works for as many registers as we have labelled nops. In the list of instructions below, whenever a register name is surrounded by question marks (e.g., ?bx?), the indicated register is the default register, used when there are no arguments to the instruction. If a label nop (nop-A, nop-B, etc.) follows the instruction in the creature's code, the indicated register is used instead of the default register. nops can also be used as labels for a search or jump. For example, the following code snippet,

`search-f`
`nop-B`
`nop-A`

would search forward in the genome for the complementary label

`nop-C`
`nop-B`

(assuming we had at least three `nops` in the current instruction set). If desired, this behaviour can be modified so that the search is done for a copy of the label instead of its complement. The complements for individual `nops` are as follows (assuming exactly three labelled `nops`):

Nop	Complement
`nop-A`	`nop-B`
`nop-B`	`nop-C`
`nop-C`	`nop-A`

The extension to different numbers of labelled `nops` is straightforward.

A sample self-replicating creature using instructions from Tables 2.2-2.6 is shown in Fig. 2.3. The string shown replicates by: searching forward (`instruction 1`) for the complement of the template `nop-A nop-A` (`2-3`), which is `nop-B nop-B` (`21-22`), manipulating this value in an internal register to find the genome length (`4-5`), allocating enough memory to store code of the genome length (`6`), setting registers to prepare for copying (`7-11`), copying the instructions one at a time (`12-19`) until all instructions have been copied (`15-16`), and replicating (`20`)—placing the daughter in its own grid site. Execution restarts at the beginning of the genome when the end of the genome is reached, and continues until the organism is replaced by the newly replicated daughter of another organism (or its own daughter). The `copy` command

Type	Name	Explanation
Null operations	nop-A	Labelled nops. Do nothing when executed. These also act as letters in labels, and arguments to certain instructions.
	nop-B	
	nop-C	
	nop-X	A pure no-operation instruction. Does not act as an argument to any commands.
Flow control operations	if-not-0	If the value of the ?bx? register is non-zero, execute the next instruction, otherwise skip it.
	if-n-equ	If the value of the ?bx? register does not equal the value of its complementary register, execute the next instruction, otherwise skip it. For example, a nop-A following this command causes the values of ax and bx to be compared.
	if-bit-1	Execute the next instruction if the last bit of ?bx? is 1.
	jump-b	If a label follows, search for its complement in the part of the genome before the current instruction, and if a match is found, change the instruction pointer to point at the last instruction of the complementary label. If there is a label, but its complement is not found, do nothing. If there is no label following, decrement the instruction pointer bx instructions. If the instruction pointer becomes negative, reset it to a positive value such that the new value is less than the size of the genome and the old and new values share the same remainder modulo the genome size.
	jump-f	If a label follows, search for its complement in the part of the genome after the current instruction, and if a match is found, change the instruction pointer to point at the last instruction of the complementary label. If there is a label, but its complement is not found, do nothing. If there is no label following, increment the instruction pointer bx instructions. If the instruction pointer becomes larger than the size of the genome, reset it to a positive value such that the new value is less than the size of the genome and the old and new values share the same remainder modulo the genome size.

Table 2.2: Sanda instructions (part 1/5).

Type	Name	Explanation
Flow control operations (cont'd)	`jump-p`	Jump into the genome of the CPU that the executing CPU is facing. If a label follows, search for its complement from the beginning of the target genome, and if a match is found, change the instruction pointer to point at the last instruction of the complementary label. If there is a label, but its complement is not found, do nothing. If there is no label, jump to instruction `bx` in the target genome. If the instruction pointer becomes larger than the size of the target genome, reset it to a positive value such that the new value is less than the size of the genome and the old and new values share the same remainder modulo the genome size. A CPU's instruction pointer may only point at an instruction in its own genome or in the genome of the CPU it is facing.
	`call`	Push the location of the next instruction on the stack, and jump forward to the complement of the label which follows. If there is no label, jump `bx` instructions. See `jump-f` for further details.
	`call-p`	Push the location of the next instruction on the stack, and jump to the complement of the label which follows in the genome of the CPU currently being faced. If there is no label, jump to instruction `bx` of the target genome. See `jump-p` for further details.
	`return`	Pop the top value from the stack, and move the instruction pointer to that location in the creature's genome. If the instruction pointer no longer points at a valid genome site, reset the instruction pointer as in `jump-f` or `jump-b`.
Single argument math operations	`shift-r`	Rotate the bits of the `?bx?` register right.
	`shift-l`	Rotate the bits of the `?bx?` register left.
	`bit-1`	Set the last bit of `?bx?` to `1`.
	`inc`	Increment `?bx?`.
	`dec`	Decrement `?bx?`.
	`zero`	Set `?bx?` to zero.
	`push`	Push `?bx?` onto the stack.
	`pop`	Pop the first value in the stack into `?bx?`.

Table 2.3: Sanda instructions (part 2/5).

Type	Name	Explanation
Single argument math operations (cont'd)	`set-num`	Set `bx` to the ternary equivalent of the label which follows, defining `nop-A` as `0`, `nop-B` as `1` and `nop-C` as 2. For example, `nop-C nop-A nop-B` is `2 0 1` in ternary, or $2 \times 3^2 + 0 \times 3 + 1 \times 1 = 19$ in decimal. If there is no label, set `bx` to zero.
Double argument math operations	`add`	Set `?bx?` equal to the sum of the values of the `bx` and `cx` registers (`?bx?` = `bx`+ `cx`).
	`sub`	`?bx?` = `bx - cx`.
	`nand`	`?bx?`= `bx` NAND `cx` (bitwise NAND).
	`nor`	`?bx?` = `bx` NOR `cx` (bitwise NOR).
	`order`	Swap the values of `bx` and `cx`, if needed, so that `cx > bx`.
"Biological" operations	`allocate`	Allocate memory for `bx` instructions at the end of the current genome for this CPU and return the start location of this memory in `ax`. This instruction does nothing if there has not been a successful `divide` since the last `allocate`. The total size of the genome after allocation is forced to be less than a user defined maximum value (default 128).
	`divide`	Split the genome at `?ax?`, placing the instructions beyond the dividing point into a neighbouring cell. This instruction has no effect if either the mother or the daughter genome would be less than a minimum number (default 10) of instructions long.
	`c-alloc`	Allocate memory equal to the size of the current genome at the end of the genome and return the location of the start of this memory in `ax`. This instruction does nothing if there has not been a successful `c-divide` since the last `c-alloc`. This instruction and the next are used instead of `allocate` and `divide` when we want to experiment with creatures with constant genome sizes.
	`c-divide`	Split the genome of the creature in half, placing the instructions beyond the division point into a neighbouring cell.

Table 2.4: Sanda instructions (part 3/5).

Type	Name	Explanation
"Biological" operations (cont'd)	`copy`	Copy the instruction from the genome location pointed to by the `bx` register to the memory location pointed to by `ax` + `bx`, i.e., copy the instruction at location `bx` into a location offset by `ax`. If either of the locations is not a valid genome location, this command uses a modified value like the one used for the instruction pointer in `jump-f` or `jump-b`.
	`read`	Copy the instruction at location `bx` in the genome into the `cx` register. Again, if `bx` is out of range, an appropriate "modulo" value is used.
	`write`	Copy the value of the `cx` register as an instruction into the memory location at `ax` + `bx`.
	`if-n-cpy`	Only execute the next line if the contents of memory locations `bx` and `ax + bx` are *identical*; otherwise skip it. This command has an error rate equal to the copy mutation rate. (It can be used for copy error checking).
I/O and "sensory" operations	`get`	Read the value pointed to by the input pointer from the input buffer and place it in the `?cx?` register.
	`put`	Write the value of the `?bx?` register into the output buffer, and then set the register to zero.
	`search-f`	If a label follows, search forward for the complementary label and place the distance (in instructions) to it in the `bx` register and the size of the label in `cx`. If a complementary label is not found, a distance of 0 is returned in `bx`. If no label follows, `bx` is unchanged and `cx` is set to 0.
	`search-b`	If a label follows, search backward for the complementary label and place the distance (in instructions) to it in the `bx` register and the size of the label in `cx`. If a complementary label is not found, a distance of 0 is returned in `bx`. If no label follows, `bx` is unchanged and `cx` is set to 0.
Additional Instructions	`switch_stack`	Switch the active stack.
	`rotate-l`	Rotate the current facing of the CPU counterclockwise.
	`rotate-r`	Rotate the current facing of the CPU clockwise.

Table 2.5: Sanda instructions (part 4/5).

Type	Name	Explanation
Additional Instructions (cont'd)	`inject`	This instruction acts somewhat like `divide`, but instead of killing another creature and replacing it with the executing CPU's daughter, the daughter code is instead injected into the middle of a running CPU's memory. The CPU currently being faced is injected, and the injection position is chosen by matching complementary labels. If a complementary label can not be found in the genome of the CPU faced, or there is no label following this instruction, the instruction fails.
	`set-cmut`	This instruction allows a CPU to set its own copy mutation rate. The value in `?bx?` becomes the CPU's new copy mutation rate ($\times 10^{-4}$).
	`mod-cmut`	This instruction modifies the copy mutation rate of a CPU. When executed, the copy mutation rate of the CPU has `?bx?` $\times 10^{-4}$ added to it.

Table 2.6: Sanda instructions (part 5/5).

(`14` in this particular genotype) fails and writes a random instruction with probability γ (the copy mutation rate).

A snippet of code which takes two numbers from the input buffer, adds them, and outputs the result to the output buffer is shown below.

```
get
nop-B
get
add
put
```

A value from the input buffer is placed in `bx`, then the next value from the input buffer in `cx`. The values of `bx` and `cx` are added and the result placed in `bx`. Finally, the value in `bx` is output to the output buffer. The fragment above is obviously written by a human—rarely do sanda creatures evolve any code so simple.

We can select for code which accomplishes certain tasks by rewarding CPUs which

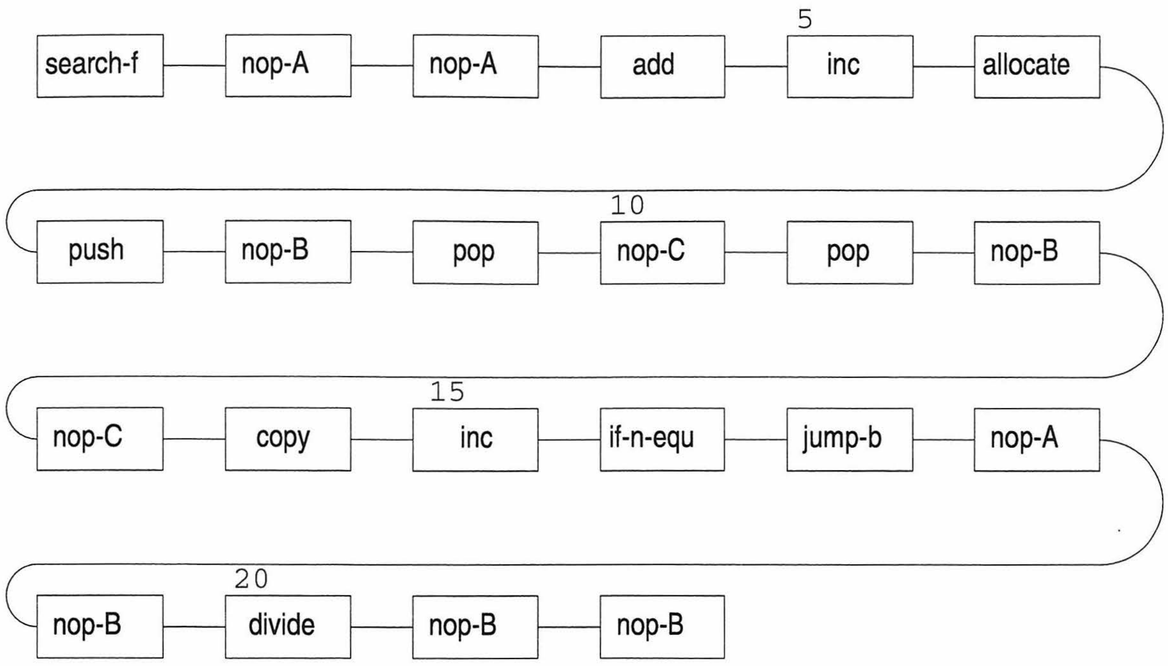


Figure 2.3: Example sanda genome. Sanda organisms have genomes which are strings of sanda instructions.

accomplish this result—or parts of this result, such as getting numbers from the input buffer, executing the add instruction, or outputting to the output buffer—with lower demerit values. Lower demerit values lead to a higher replication rate for selected CPUs, and a growth in the number of CPUs of this genotype.

2.6 Mutation Methods

Mutations are random changes in the code of a CPU or its daughter. Without mutations, the system would settle into a non-interesting steady state where all the creatures would have the same genotype. There are several ways of introducing mutations into the system (Table 2.7). Any combination of these methods can also be used.

In addition to these explicit mutation mechanisms, incomplete or faulty copy algorithms in creature's genomes cause implicit mutations, as do certain exotic instructions (insert for example). These implicit mutations will tend to increase the effective mutation rate.

Mutation Type	Explanation
Per-instruction Copy	Each time the `copy` instruction is executed, there is a finite chance that an instruction chosen at random from the instruction set will be written instead of the intended instruction.
On-replication Copy	Each time a creature replicates, there is a chance of a single, randomly chosen instruction in the daughter being mutated into another. This may be used instead of per-instruction copy mutations to allow genomes of greater length and information content to be viable.
Point	A finite probability per time of a randomly chosen instruction in a creature's genome (including memory allocated for its daughter) being mutated.

Table 2.7: Mutation methods.

2.7 Replication and Death

Genotypes which allow faster replication, by a more efficient copy algorithm, a decrease in genome size (less instructions to be copied), or by accomplishing user-defined tasks, will *ceteris paribus* have a growing number of CPUs. The speed of this growth is greatly affected by the selection scheme used when choosing CPUs to be replaced by newly replicated daughters. Fig. 2.4 shows some selection schemes that can be used in sanda. In a selection scheme which includes *matricide*, the daughter can replace the parent; without matricide, the daughter can only replace one of the parent's eight neighbours. In an age-biased selection scheme, the oldest creature in the neighbourhood is replaced, whereas in a non-biased scheme, a creature is chosen at random. Additional schemes (e.g., replacing the creature with the highest error rate) can be implemented easily by modifying a single procedure in the program.

In addition to the replacement of a CPU by a newly created creature, death in sanda can be implemented explicitly—CPUs past a certain age or CPUs which make too many errors may be killed. This can add another factor to selection schemes; dead cells are replicated into before living cells are replaced.

	Non-biased	Age-biased
Matricide	High rates of copy mutation lead to a soup with no viable genotypes. Low diffusion rate.	High rates of copy mutation lead to a soup with no viable genotypes. High diffusion rate.
No Matricide	A small number of viable genotypes even with very high mutation rates. Low diffusion rate.	A small number of viable genotypes even with very high mutation rates. High diffusion rate.

Figure 2.4: Replacement selection schemes. Various combinations are possible and lead to different dynamics.

2.8 Parallel Emulation Algorithm

To properly emulate the independent metabolism of each creature, we need an emulation algorithm to execute the instructions in each creature's genome in turn. In avida and tierra several of a creature's instructions were executed sequentially, and then several of the next creature's instructions, and so on (*block time-slicing*). Creatures which performed user-defined tasks were rewarded with larger blocks of execution time.

In sanda, such approximations are avoided by defining a physical time for the system. Each cell in the system has its own time. In a simple algorithm, one instruction is executed for the CPU of the cell with the lowest time in the system and the time needed for the CPU to execute the instruction is added to the cell's time (and CPU's age) (Fig. 2.5). This is repeated *ad nauseum.* Each instruction takes a certain base amount of time to execute (the length of this base execution time can be set to different values for each instruction by the user). This base execution time is modified by the demerit of the CPU executing the instruction; a CPU which has performed user-defined tasks and has thus lowered its demerit value will take less time to perform the same instruction than a CPU with a higher demerit value.

The method outlined above works well for small grid sizes on a single computer. However, for larger population sizes, or sanda runs across many processors, the computational load of maintaining a sorted list of cells and their times is prohibitive. We can solve this problem by recognizing that all interactions in the system are local. Cells further than two grid sites apart can not interact with one another via execution of a single instruction (Fig. 2.6). This allows us to relax the condition for update of a cell from the single cell having the lowest time value in the whole system, to those having the lowest time value in their 25-site neighbourhoods (Fig. 2.7). If we choose to ignore the second-order effects leading to interactions between cells two sites apart, we can get even higher emulation speeds by updating cells with the lowest time-values in 9-site neighbourhoods (Fig. 2.8)

With these localized time-slicing methods, the computational time needed per

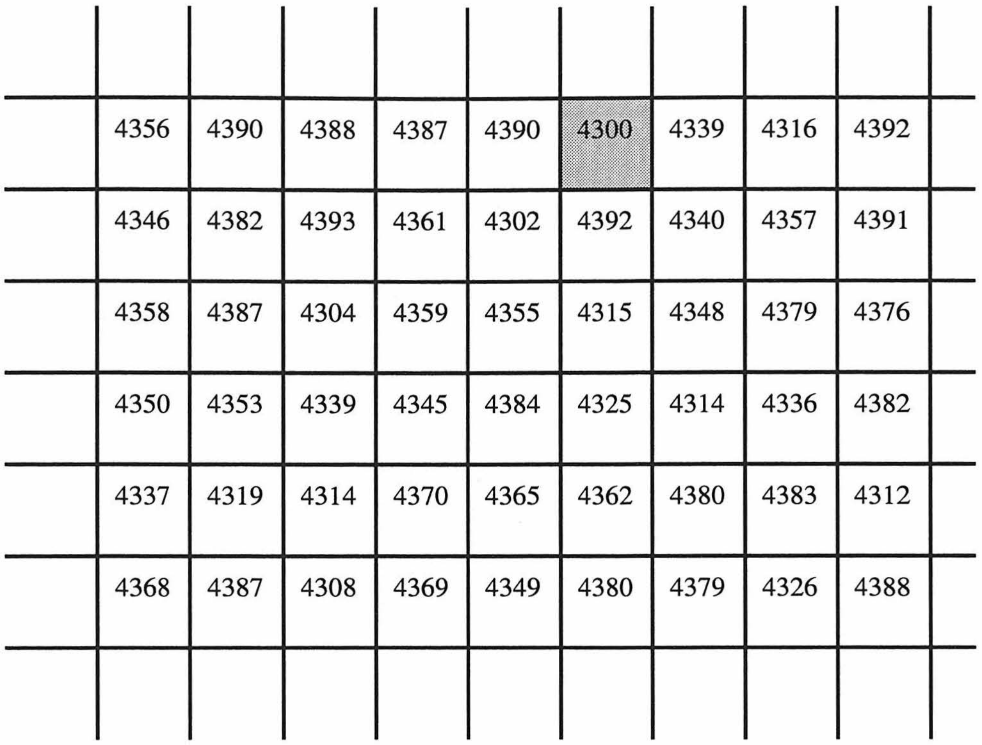


Figure 2.5: A non-local time-slicing algorithm. Cells are shown with their time values. Only the cell with the smallest time value (shown shaded) executes an instruction. The new time value for the executed cell is then sorted into the global list of time values. The computation time needed for this sorting increases with the size of the grid. Also, for a grid spread across many processors, time values must be communicated between processors for each executed instruction, and only one processor can execute an instruction at a time.

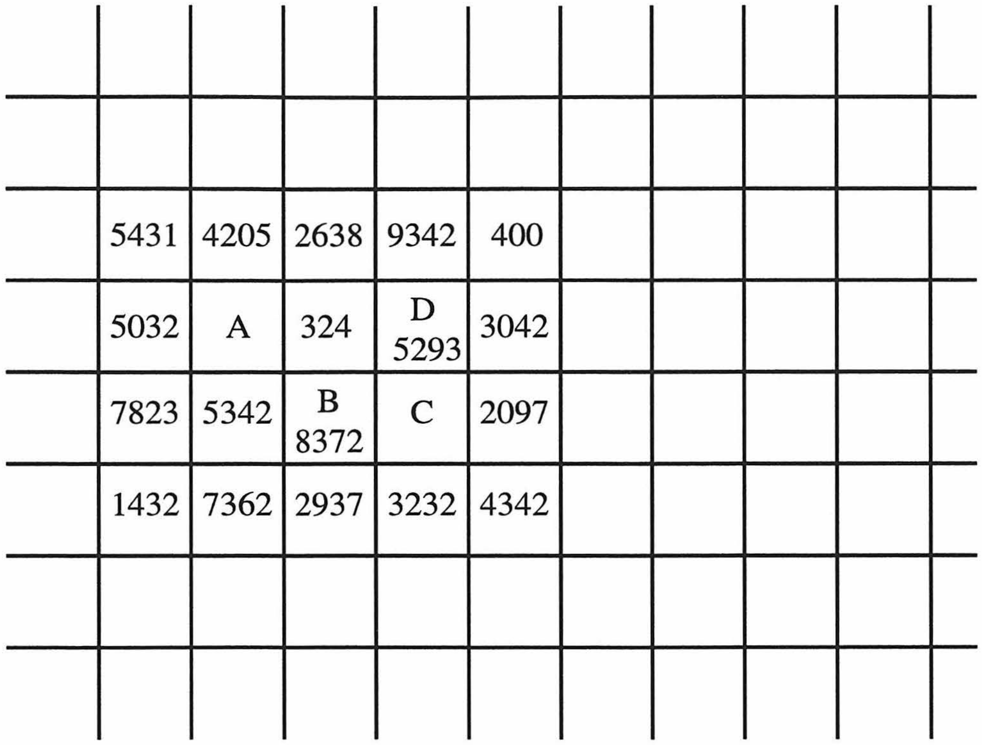


Figure 2.6: Inter-cell interactions. Each square in the diagram is a cell, and the numbers are CPU ages. Assuming an age-biased selection scheme, if **A** divides, it will replace **B**—a one-site distance interaction. If **C** divides next, then **A** will have indirectly affected **C** and **D** as well, since **C** will now replicate into **D** instead of **B**—two-site distance interactions. Longer range (greater than 2 sites) interactions cannot result from the execution of a single instruction.

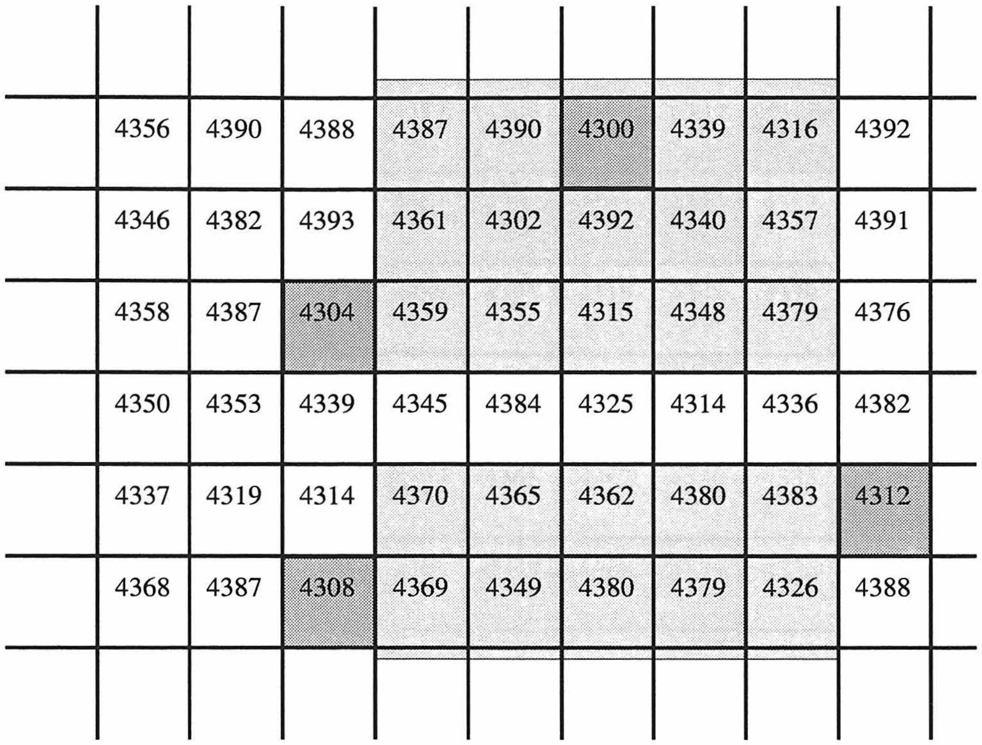
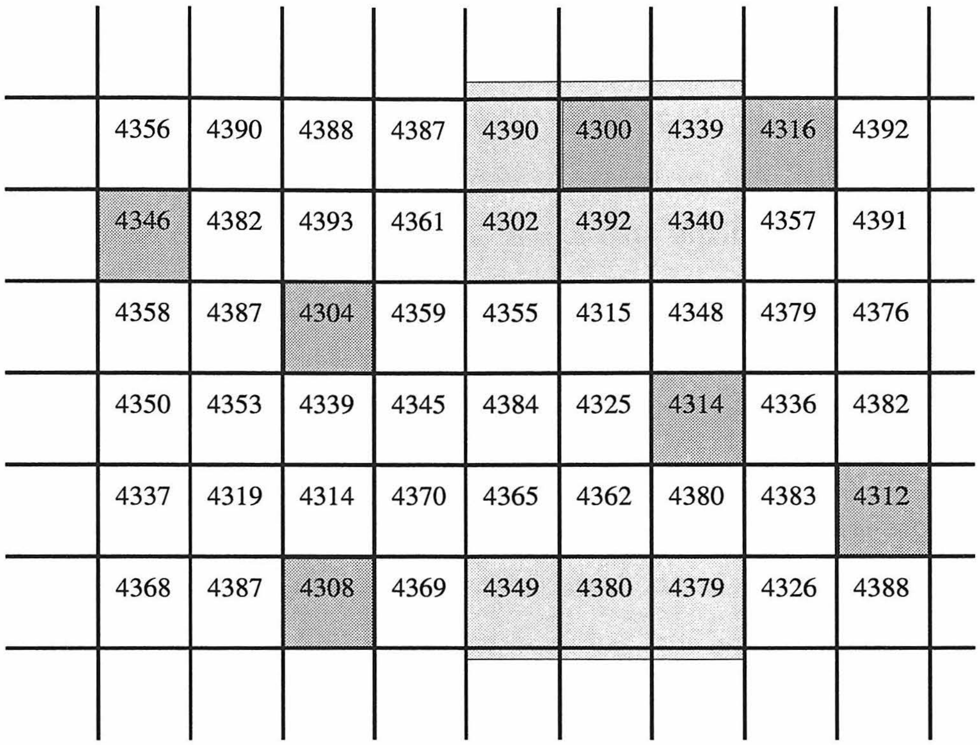


Figure 2.7: Local time-slicing algorithm with 25 neighbours. The cells which will execute instructions next are in dark grey. The 25-site neighbourhood of the cell with time 4300 is shown in light grey. Each instruction execution is followed by 24 integer comparisons.



4356	4390	4388	4387	4390	4300	4339	4316	4392
4346	4382	4393	4361	4302	4392	4340	4357	4391
4358	4387	4304	4359	4355	4315	4348	4379	4376
4350	4353	4339	4345	4384	4325	4314	4336	4382
4337	4319	4314	4370	4365	4362	4380	4383	4312
4368	4387	4308	4369	4349	4380	4379	4326	4388

Figure 2.8: Local time-slicing algorithm with 9 neighbours. The cells which will execute instructions next are in dark grey. The 9-site neighbourhood of the cell with time 4300 is shown in light grey. Each instruction execution is followed by only 8 integer comparisons, allowing for very speedy emulation of independent, parallel creature execution.

instruction execution no longer depends on the size of the soup. Further, the size and frequency of interprocessor communications in multiprocessor runs are greatly reduced, as only information about cells lying on the border between two processors need be communicated and it is no longer necessary to communicate the time values of all the cells in a processor's grid to other processors.

2.9 Sanda on Parallel Computers

Sanda was explicitly written for use on multiprocessor machines, specifically the Intel Paragons. The main bottlenecks for emulation speed on the Paragons are in the latency of inter-processor communications, and in data input and output. Several of the design decisions for sanda reflect these conditions.

The grid is divided between processors as shown in Fig. 2.9. Each processor has an $n \times m$ portion of the grid and must communicate with the eight processors surrounding it. In practice, a processor communicates directly only with the four processors sharing borders with it. Message-passing to processors at the corners is accomplished indirectly by relaying through processors with which the message-sending processor directly communicates (Fig. 2.9). Each processor has four input message buffers and four output message buffers, one each for each processor it borders. For any two neighbouring processors, one will be in *send mode* and the other in *receive mode* with regard to each other (Fig. 2.10). A processor can only send data to a neighbouring processor if it is in send mode in regard to that processor. Once a processor sends data to a neighbour it switches to receive mode with regard to that neighbour. These mechanisms are in place to avoid corruption of the receive buffer by newly transmitted data before the previously received data has been properly processed.

The data to be sent between processors includes records of new CPUs, new genotypes, and time updates for cells on the borders between the processors. Depending on the instruction set (whether or not a CPU can examine or change its neighbour's genome) and the time-slicing algorithm chosen (9-neighbour or 25-neighbour time-slicing), data for cells one site removed from the border may or may not need to be

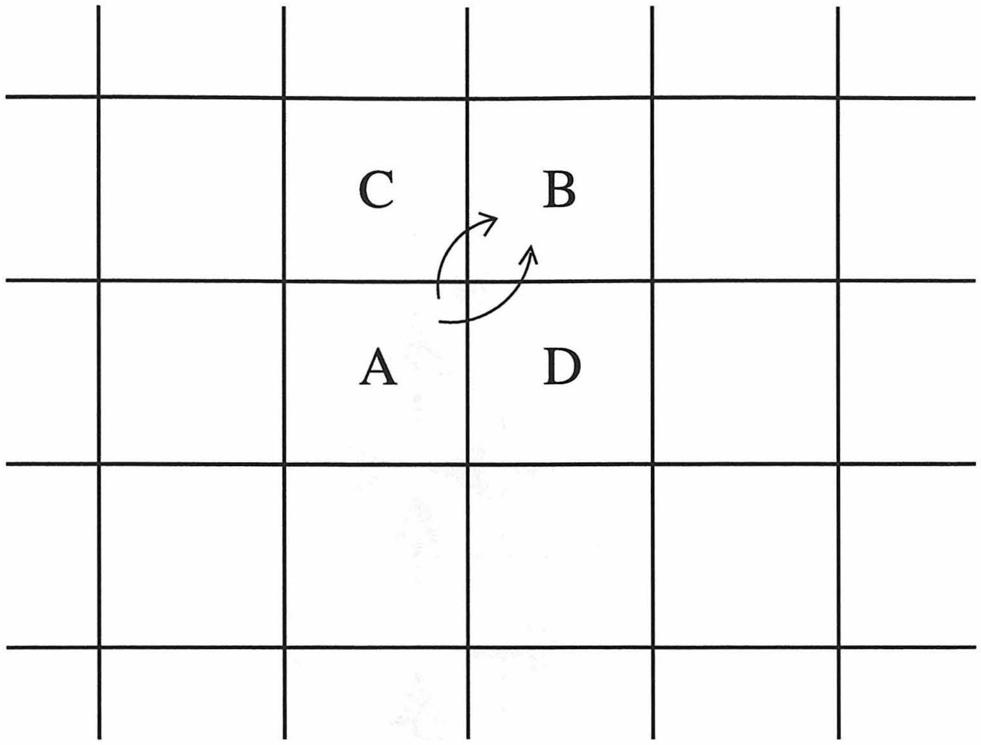


Figure 2.9: Grid allocation and interprocessor communication relaying. Each processor has part of the grid in classic "patchwork" fashion. Information about cells on the borders is communicated to directly neighbouring processors and is relayed to those processors sharing only a corner with the originating processor. In the diagram, information from processor A needed by processor B is relayed through processor C or D.

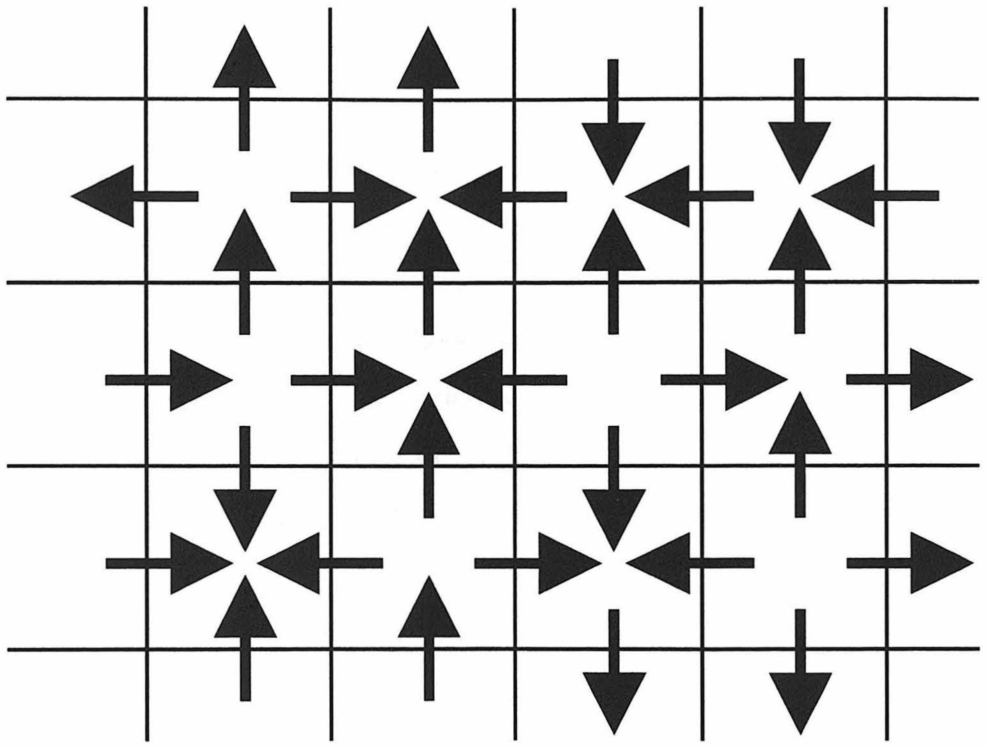


Figure 2.10: Processors in send and receive modes. For each pair of neighbours one is in send mode and another in receive mode. In the diagram, the arrow heads point to the processor in receive mode.

sent.

2.10 User Files

Sanda is designed to be run in batch mode with little interaction with the user. To alleviate the bottlenecks caused during disk I/O, sanda fully utilizes the pfs filesystem of the Intel Paragons, which has multiple hard drives and hard drive controllers. Sanda's output is widely configurable, and due to the vast amount of data generated from large population sizes, it is recommended that the user consider the output data needed for his or her purposes and modify the output classes to output only needed data, or that the user choose a low sampling rate of several generations (a generation is defined as the time it takes the system to have a number of births equal to the population or grid size) between outputs. The default output format is two binary files, one containing genotype data (genotype name, total number, fitness, etc.) and one containing update data (time, existing genotype populations, etc.). A 10 hour sanda run on one processor generates about 5 megabytes of data with output once per generation using the default outputs. Obviously, with up to 512 processors, this is not a recommended setting.

Most adjustable system parameters can be changed by editing the default_params.san file and recompiling. The main impetus for reprogramming sanda in C++ was the ease of modification and extension that it provides. Most classes should be easily modifiable to allow specialized versions of sanda.

2.11 Extensions

Sanda can be easily extended to higher-dimensional grids, to different selection schemes, and to include different interactions between CPUs. The easiest changes to sanda involve changing the instruction set (see below for an example). For instance, if you were interested in evolving genomes which could sort their inputs, you might want to implement pointer operations on buffers, such as a `swap` instruction or a better compare instruction.

A very simple alternative instruction set is the *neutral model* instruction set. It has only two instructions:

Instruction Name	Explanation
nop-A	Do nothing.
NM-divide	Divide. The daughter is of the same genotype as the parent with probability F, of a new genotype but still viable with probability $F \times N$, and non-viable with probability $1 - F - F \times N$.

Genomes in this variation of sanda are all one instruction long. Genotypes are redefined; merely having the same genome does not mean two creatures in the neutral model share the same genotype. This model is explored further in Chapter 6.

Chapter 3 Propagation of Information

Sanda models populations of self-replicating strings residing in an environment with spatial structure. In this section, I test the sanda system by comparing the propagation of information in sanda to theoretical predictions and to propagation in biological systems. I observe the propagation of information in sanda as a function of the fitness and mutation rate of carrier strings. Comparison with theoretical predictions based on the reaction-diffusion equation shows that the response of the artificial system to fluctuations (e.g., velocity of the information wave as a function of relative fitness) closely follows that of natural systems. I find that the relaxation time of the system depends on the speed of propagation of information and the size of the system. This analysis offers the possibility of determining the minimal system size for observation of non-equilibrium effects at fixed mutation rate.

3.1 Introduction

Thermodynamic equilibrium systems respond to perturbations with waves that re-establish equilibrium. This is a general feature of statistical systems, but it can also be observed in natural populations, where the disturbance of interest is a new species with either negligible or positive fitness advantage. The new species spreads through the population at a rate dependent on its relative fitness and some basic properties of the medium which can be summarized by the diffusion coefficient. This problem has been addressed theoretically [18] and experimentally (see, for example, Ref. [17] and references therein) since early this century. The application of the appropriate machinery (diffusion equations) to the spatial propagation of *information* rather than species, is much more recent, and has been successful in the description of experiments with *in vitro* evolving RNA [11, 28].

Systems of self-replicating information (cf. the replicating RNA system mentioned above) are often thought to represent the simplest living system. They offer the chance to isolate the mechanisms involved in information transfer (from environment into the genome) and propagation (throughout the population), and study them in detail.

It has long been suspected that living systems operate, in a thermodynamical sense, far away from the equilibrium state. On the molecular scale, many of the chemical reactions occurring in a cell's metabolism require non-equilibrium conditions. On a larger scale, it appears that only a system far away from equilibrium can produce the required diversity (in genome) for evolution to proceed effectively (I will comment on this below).

In the systems that we are interested in—systems of self-replicating information in a noisy and information-rich environment—the processes that work for and against equilibration of information are clearly mutation and replication. In the absence of mutation, replication leads to a uniform non-evolving state where every member of the population is identical. Mutation in the absence of replication, on the other hand, leads to maximal diversity of the population but no evolution either, as selection is absent. Thus, effective adaptation and evolution depend on a balance of these driving forces (see, e.g., [2, 1]). The relaxation time of such a system, however, just as in thermodynamical systems, is mainly dictated by the mutation rate which plays the role of "temperature" in these systems [1]. As such, it represents a crucial parameter which determines how close the system is to "thermodynamical" equilibrium. Clearly, a relaxation time larger than the average time between (advantageous) mutations will result in a non-equilibrium system, while a smaller relaxation time leads to fast equilibration. The relaxation time may be defined as the time it takes information to spread throughout the entire system (i.e., travel an average distance of half the "diameter" of the population). A non-equilibrium population therefore can always be obtained (at fixed mutation rate) by increasing the size of the system. At the same time, such a large system segments into areas that effectively can not communicate with each other, but are close to equilibrium themselves. This may be the key to genomic diversity, and possibly to speciation in the absence of niches and explicit barriers.

The advent of artificial living systems such as tierra [32, 2] and avida [5, 6] have opened up the possibility of checking these ideas explicitly, as the evolutionary pace in systems both close and far away from equilibrium can be investigated directly. As a foundation for such experiments, here I investigate the dynamics of information propagation in the artificial life system sanda. This is a necessary capability for investigating arbitrarily large populations of strings of code. The purpose of my experiments is two-fold. On the one hand, I would like to "validate" the Artificial Life system by comparing experimental results to theoretical predictions known to describe natural systems, such as waves of RNA strings replicating in $Q\beta$-replicase [11, 28]. On the other hand, this benchmark allows determination of the diffusion coefficient and the velocity of information propagation from relative fitness and mutation rate. Finally, I arrive at an estimate of the minimum system size which guarantees that the population will not, on average, equilibrate.

In the next section I briefly describe the sanda configuration used for these experiments. The third section introduces the reaction-diffusion equation for a discrete system and analytical results for the wavefront velocity as a function of relative fitness and mutation rate. I describe results in the subsequent section and close with some comments and conclusions.

3.2 The System

For these experiments, when a string replicates, it places its child in one of the *eight* (not including itself) adjacent grid spots, replacing any string which may have been there. Grid sites are allowed to be empty—have no string. If there is an adjacent site which contains no string, the daughter is placed there rather than replacing a string. In these experiments, when there are no adjacent empty sites, the string to be replaced is chosen in either of two ways; *random selection* where an adjacent site is selected purely at random, or *age-biased selection* where the oldest string among the neighbours is replaced. As we shall see, the selection mechanism has a significant effect on the spread of information.

It should be noted that this birth process, and indeed all interactions between strings, are local processes in which only strings adjacent to each other on the grid may affect each other directly. This is important as it both supplies the structure needed for studies of the spatial characteristics of populations of self-replicating strings of code, and results in longer relaxation times—making possible studies of the equilibration processes of such systems and their nonequilibrium behaviour.

I have studied the relaxation of the system both with and without mutations allowed and for varying relative fitness differences between originally existing wild-type or background genotypes and new, fitter mutated genotypes.

3.3 Diffusion and Waves

Information in sanda is transported mainly by self-replication. When a string divides into an adjacent grid site, it is also transferring the information contained in its code (genome) to this site. I have looked at the mode and speed of this transfer in relation to the fitness of the genotype carrying the information, the fitness of the other genotypes near this carrier, and the mutation rate.

Consider what happens when one string of a new genotype appears in an area previously populated by other genotypes. I will make the assumption that the fitness of the other viable (self-replicating) genotypes near the carrier are approximately the same. This holds for cases where the carrier is moving into areas which are in local equilibrium. I will use f_c for the fitness of the newly introduced (carrier) genotype and f_b for the fitness of the background genotypes. If $f_c < f_b$, obviously the new genotype will not survive nor spread. I have studied three different cases: diffusion, wave propagation without mutations, and wave propagation with mutations.

The diffusion case represents the limit where the fitness of both genotypes are the same. It turns out that this can be modelled as a classical random walk. On average, if the carrier string replicates it will be replaced before it can replicate again. This is effectively the same as the carrier string *moving* one lattice spacing in a random direction chosen from the eight available to it (Fig. 3.1). The random

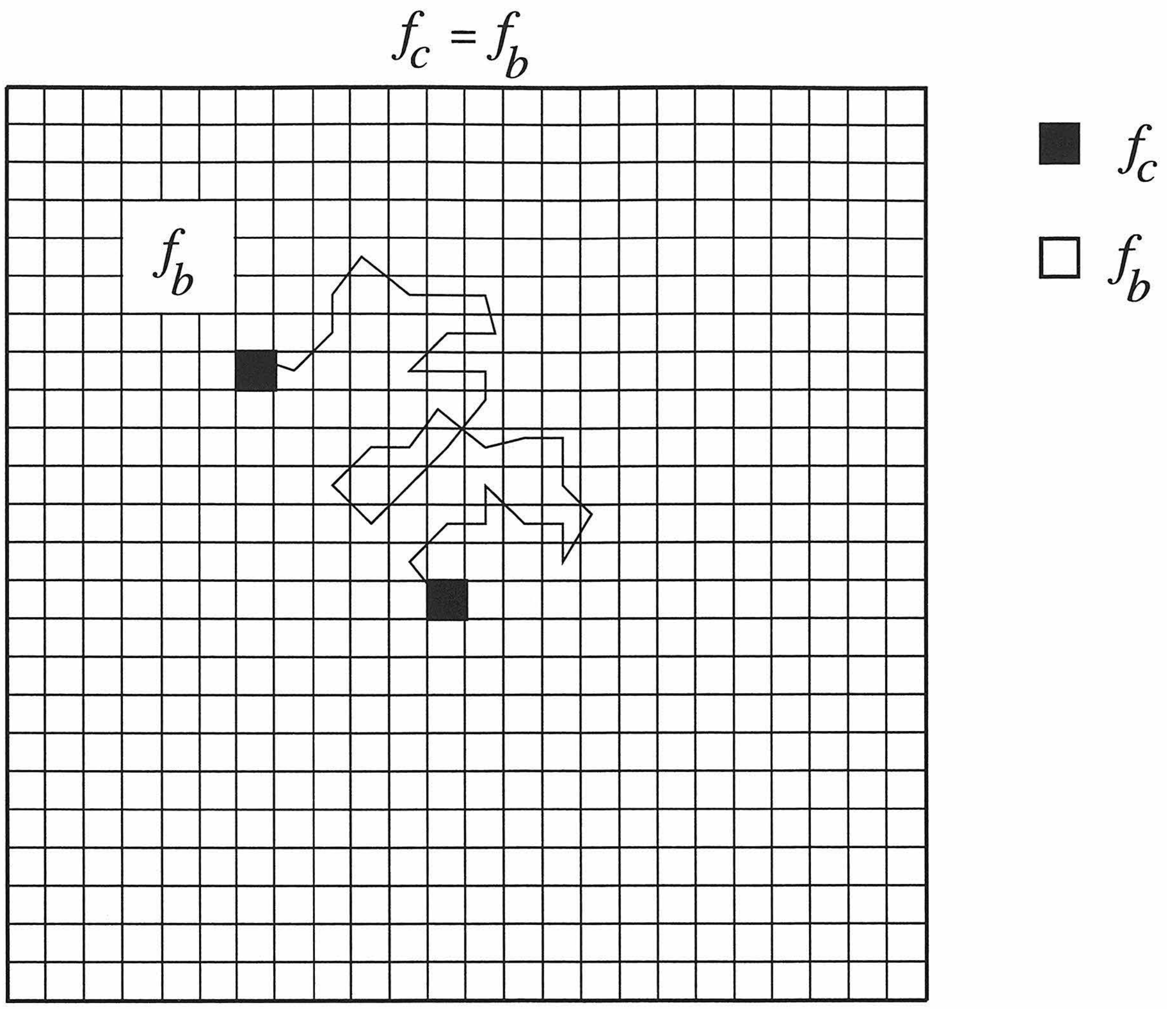


Figure 3.1: Random walk of a carrier genotype with fitness $f_c = f_b$. On average, the carrier genotype is replaced as often as it replaces another cell; the carrier genotype looks like it is stepping from site to site—a random walk.

walk is characterized by the disappearance of the mean displacement and the linear dependence on time of the mean squared displacement:

$$\langle r \rangle(t) = 0, \tag{3.1}$$

$$\langle r^2 \rangle(t) = 4Dt, \tag{3.2}$$

where D is defined as the diffusion coefficient.

For this particular choice of grid and replication rules, this find for the diffusion

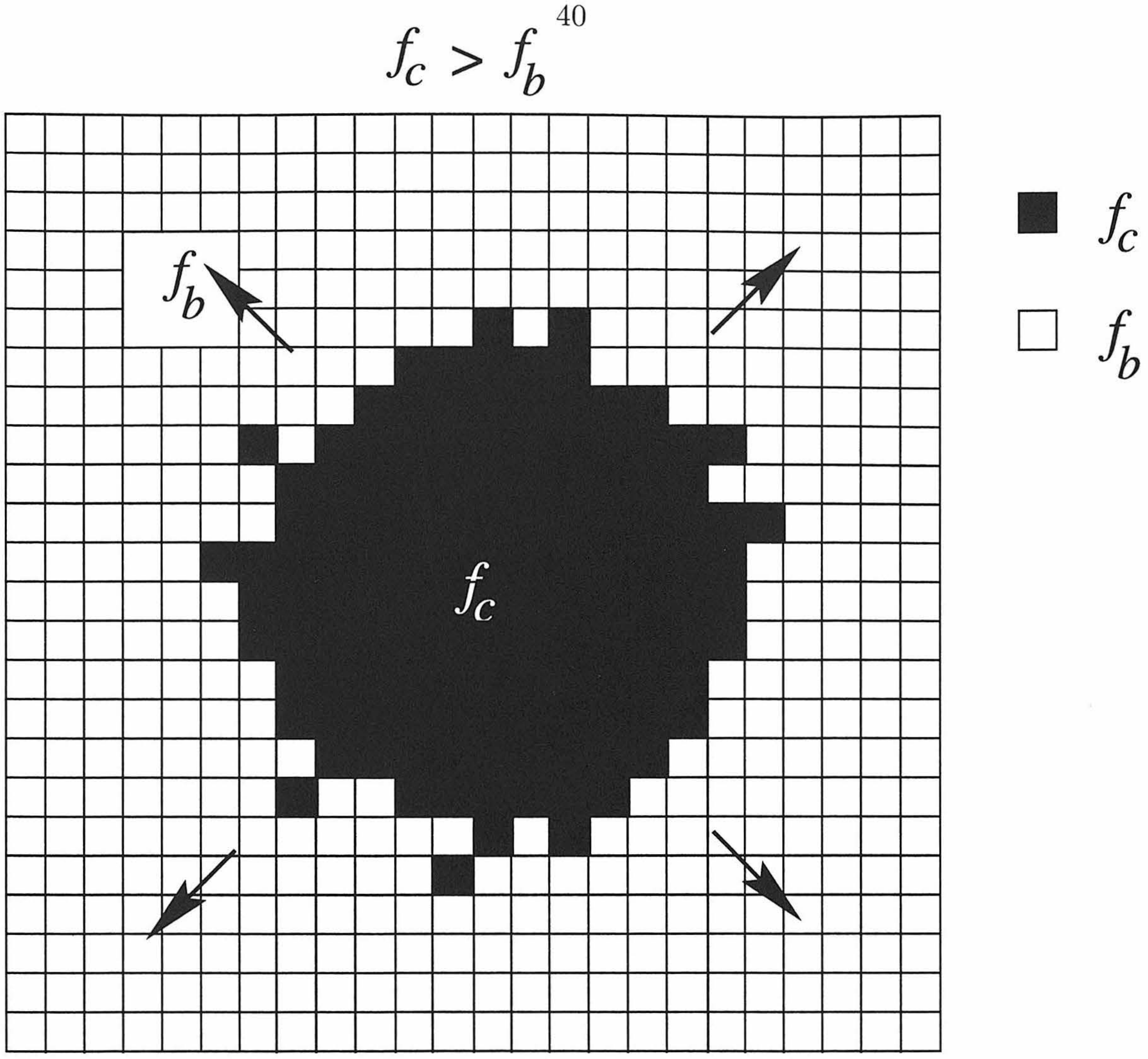


Figure 3.2: Spread of a carrier genotype with better fitness than the background genotype. The boundary between the two genotypes moves with a speed v outwards.

coefficient of a genotype with fitness f,

$$D^{(b)} = \frac{3}{8}a^2 f, \tag{3.3}$$

where a is the lattice spacing. This holds for the age-biased selection scheme where the oldest cell in the neighbourhood is replaced.

If $f_c > f_b$ then we find that instead of diffusion we obtain a roughly circular population wave of the new genotype spreading outward (Fig. 3.2). We are interested in the speed of this wavefront.

Let us first treat the case without mutation. If the radius of this wavefront is not too small, we can treat the distance from the center of the circle r as a linear coordinate. I define $\rho(r,t)$ as the mean normalized population density of strings of the new genotype at a distance r from the center at a time t measured from initial seeding with the new genotype. We assume that the ages of cells near each other have roughly the same distribution and that this distribution is genotype independent, ensuring that the selection of cells to be replaced does not depend on genotype either.

Then, we can write a flux equation (the reaction-diffusion equation) which determines the change in the population density $\rho(r,t)$ as a function of time

$$\begin{aligned}\frac{\partial\rho(r,t)}{\partial t} &= \left[\frac{3}{8}\rho(r-a,t)+\frac{1}{4}\rho(r,t)+\frac{3}{8}\rho(r+a,t)\right] f_c\,(1-\rho(r,t)) \\ &- \left[\frac{3}{8}(1-\rho(r-a,t))+\frac{1}{4}(1-\rho(r,t))+\frac{3}{8}(1-\rho(r+a,t))\right] f_b\rho(r,t)\ . \quad (3.4)\end{aligned}$$

Since we are interested in the speed of the very front of the wave, we can assume ρ to be small. Also, from physical considerations we assume ρ is reasonably smooth. Then, we can use a Taylor expansion

$$\rho(r\pm a,t)=\rho(r,t)\pm a\frac{\partial\rho(r,t)}{\partial r}+a^2\frac{\partial^2\rho(r,t)}{\partial r^2} \quad (3.5)$$

and keep the lowest order terms to obtain

$$\frac{\partial\rho(r,t)}{\partial t}=\frac{3}{8}a^2\,f_c\,\frac{\partial^2\rho(r,t)}{\partial r^2}+(f_c-f_b)\rho(r,t)\ . \quad (3.6)$$

This can be solved for the linear wavefront speed $v^{(b)}$ yielding [15]

$$\begin{aligned} v^{(b)} &= a\sqrt{\frac{3}{2}}\sqrt{f_c(f_c-f_b)} & (3.7)\\ &= 2\sqrt{D_c^{(b)}(f_c-f_b)} & (3.8)\end{aligned}$$

where $D_c^{(b)}$ is the diffusion coefficient of the carrier genotype when using a biased (by age) selection scheme.

To study the case of wave-propagation with mutation, we make the assumption that all mutations are fatal—the daughter is dead. We write equilibrium equations for this selection scheme:

$$\rho F - \rho(1-\sigma_D)^8 = 0, \tag{3.9}$$

where σ_D is the number of sites with dead strings. We can then calculate the steady state density of dead cells,

$$\sigma_D = 1 - F^{1/8} \tag{3.10}$$

where the fidelity F is the probability that a child will have the same genotype as its parent (i.e., not be mutated). As mentioned earlier, the fidelity is related to the mutation rate γ by

$$F = (1-\gamma)^{\ell} \tag{3.11}$$

where ℓ is the length of the particular string. Modifying the previous flux equation to take into account these new factors and repeating the previous analysis gives us

$$v^{(b)} = 2\,F\sqrt{D_c^{(b)}(f_c - F^{1/8} f_b)}\ . \tag{3.12}$$

Let us now consider the effects of different selection schemes for choosing cells to be replaced. The relations above hold true for the case in which we replace the oldest cell in the 8-cell neighbourhood when replicating ("age-based" selection). Another method of choosing a cell for replacement is to choose a random neighbouring cell regardless of age. This scheme, which we term random selection as opposed to the age-biased selection treated above, effectively halves the replication rate of all cells. It follows that the diffusion coefficient is also halved,

$$\begin{aligned} D^{(r)} &= \frac{3}{16}a^2 f && (3.13)\\ &= \frac{1}{2}D^{(b)} && (3.14)\end{aligned}$$

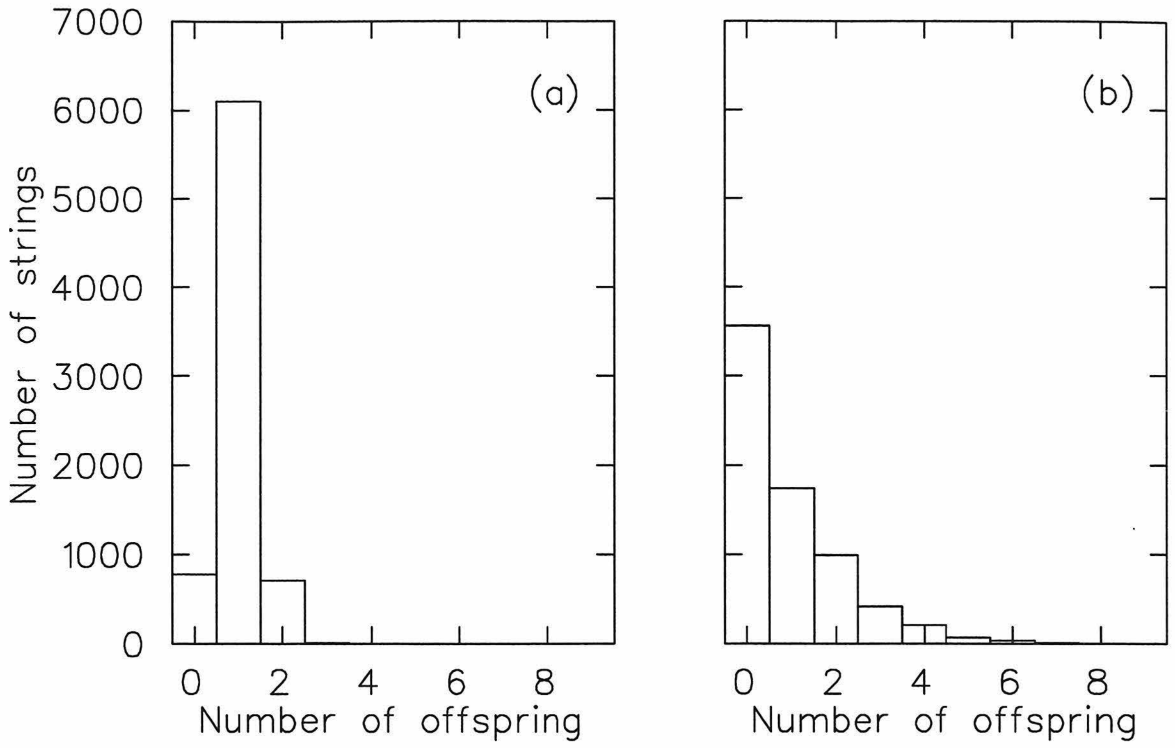


Figure 3.3: Distribution of number of strings generating different numbers of offspring, for the biased selection case [panel (a)] and the random selection scenario [panel (b)].

and for the velocity of the wavefront (with no mutation) we find

$$v^{(r)} = 2\sqrt{D_c^{(r)}\frac{(f_c - f_b)}{2}} \; . \tag{3.15}$$

In Fig. 3.3, we show a histogram of the number of offspring that a cell obtains before being replaced by a neighbour's offspring, for the biased selection case (left panel) and the random case (right panel). As expected from general arguments, half of the cells in the random selection scenario are replaced before having had a chance to produce their first offspring (resulting in a reduced diffusion coefficient), while biased selection ensures that most cells have exactly one child.

Experiments are carried out by first populating the grid with a single (background) genotype with fitness f_b. Then, a single string of the carrier genotype with fitness f_c is placed onto a point of the grid at time $t = 0$. We then observe the position and speed of the wavefronts formed, the mean squared displacement of the population of carrier genotypes, and various other parameters as a function of time.

With f_b kept constant[1], I varied f_b/f_c from 0.1 to 1.0 in increments of 0.1. Also, the mutation rate γ was varied from 0 to 14×10^{-3} mutations per instruction, in increments of 1×10^{-3}.

A comparison of the theoretical vs. measured mean square displacement as a function of time for a genotype with no fitness advantage compared to its neighbours ($f_b/f_c = 1$) is shown in Fig. 3.4. The data were obtained from approximately 1500 runs. The solid lines represent the (smoothed) averages of the measurements (for biased and random selection schemes), while the dashed lines are the theoretical predictions obtained from the diffusion coefficients (3.3) and (3.13) respectively. The slopes of the measured and predicted lines agree very well, confirming the validity of the random walk model and the diffusion coefficient predicted by it (without any free parameters). The slight discrepancy between the experimental curves and the predicted ones at small times is due to a finite-size effect that can be traced back to the coarseness of the grid.

Fig. 3.5 shows the measured values of the wavefront speed for cases where $f_c > f_b$ and without mutation, with the corresponding predictions. Again, the higher curve is for age-biased selection and the lower for random selection. Note that the wavefront speed gain from an increase in fitness ratio is much better than linear. Note also that all predictions are again free of *any* adjustable parameters.

The dependence of this curve on the mutation rate is shown in Fig. 3.6. Increasing the mutation rate tends to push the speed of the wave down. It should be noted, however, that because we have only used copy mutations there is no absolute cutoff point or error threshold F_c where all genotypes cease to be viable, with $F_c > 0$. Rather, genotypes can spread until F is very close to the limit $F_c = 0$.

Finally, the dependence of the wavefront speed on the mutation rate for a fixed value of the fitness ratio ($f_b/f_c = 0.6$) is shown in Fig. 3.7. Data were obtained from an average of four runs per point in the biased selection scheme. Again, the prediction

[1]The gestation time was approximately 330,000, where the base execution time for each instruction was (arbitrarily) set to 1000: $f_b = \frac{1}{330000}$

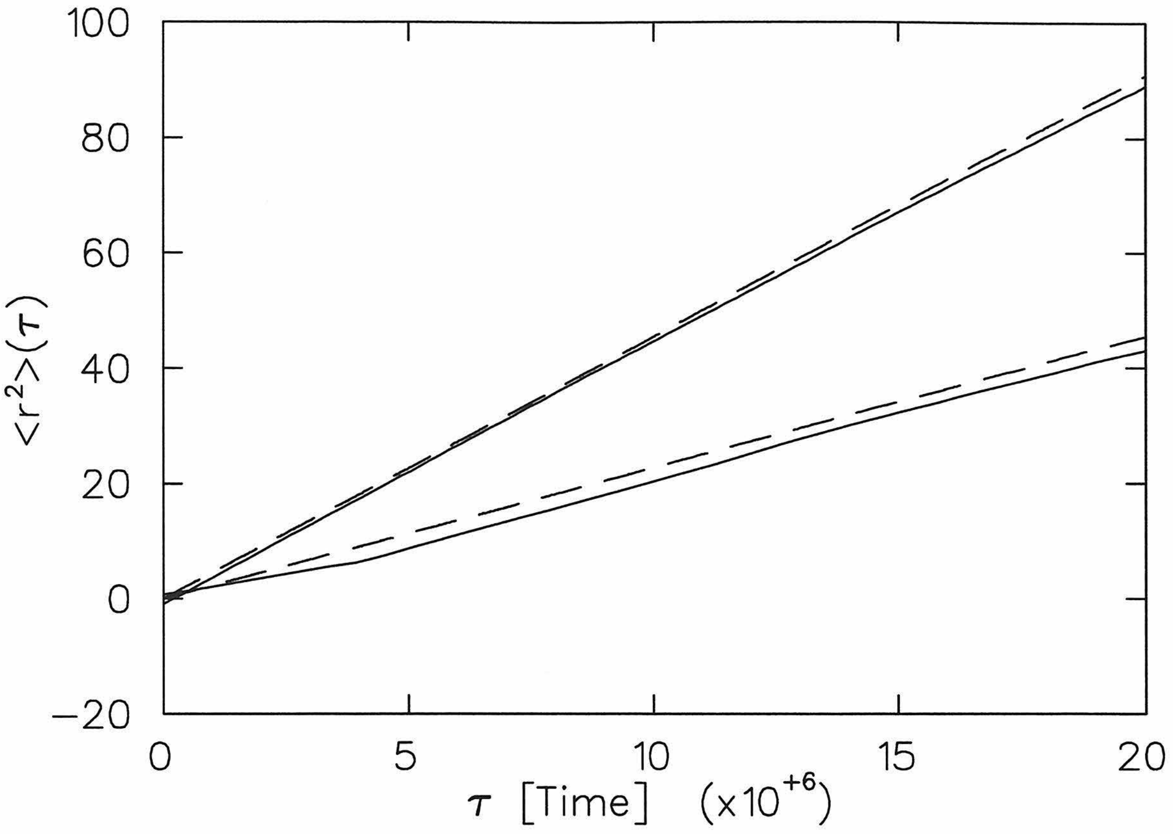


Figure 3.4: Mean squared displacement of genome as a function of time due to diffusion. Solid lines represent experimental results obtained from 1500 independent runs. Dashed lines are theoretical predictions. The upper curves are obtained with the biased selection scheme while the lower curves result from the random selection scenario.

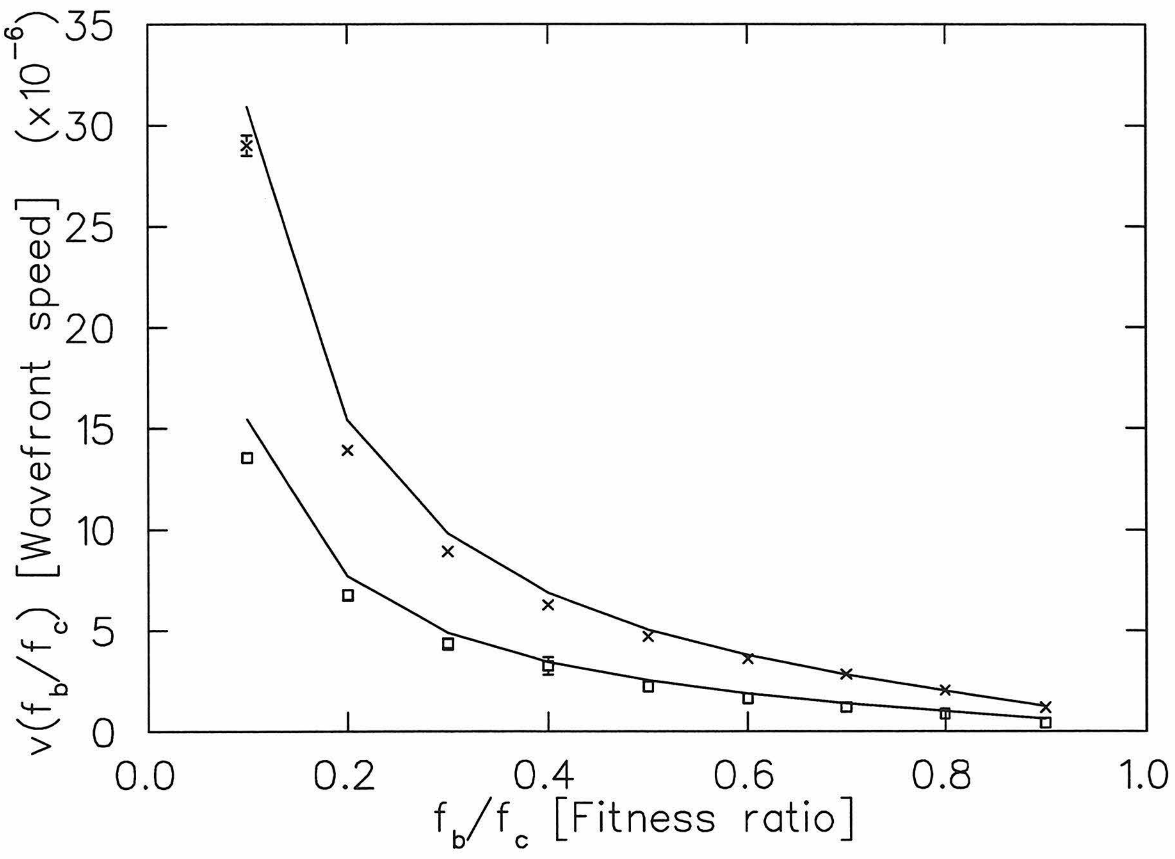


Figure 3.5: Wavefront speed of a genotype with fitness f_c propagating through a background of genotypes with fitness f_b, averaged over four runs for each data point. Upper curve: biased selection, lower curve: random selection. Solid lines are predictions of Eqs. (3.8) and (3.15).

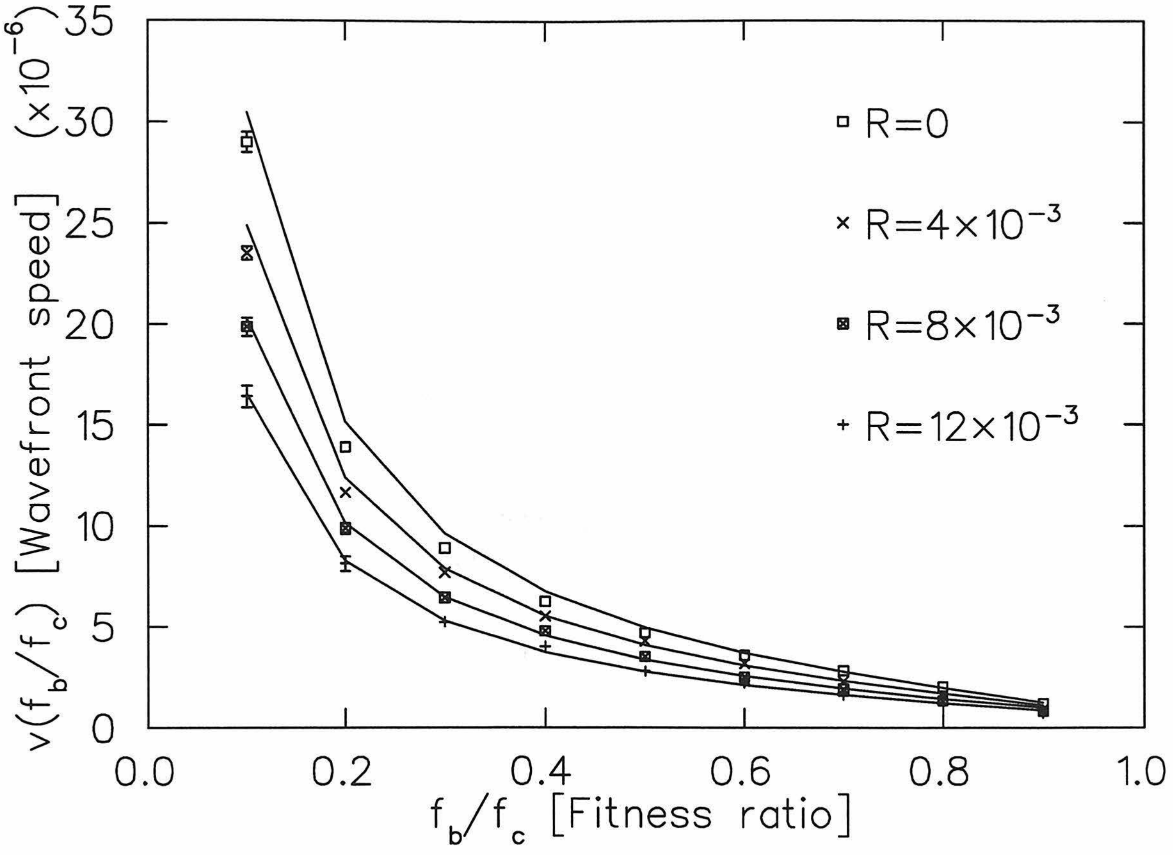


Figure 3.6: Measured wavefront speeds versus fitness ratio for selected mutation rates γ (symbols) are plotted with the theoretical predictions from Eq. (3.12) (for the biased selection scheme only).

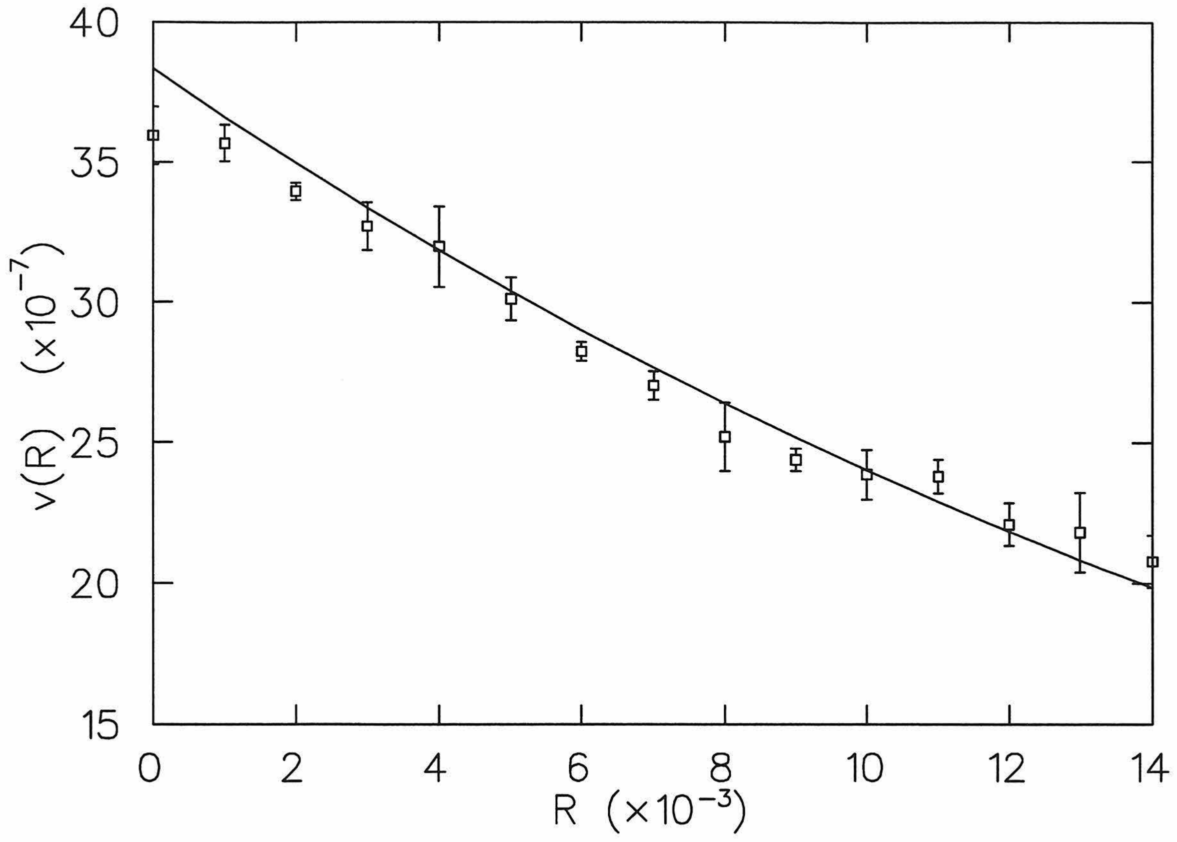


Figure 3.7: Wavefront speed of a genotype (biased selection) with relative fitness f_b/f_c as a function of mutation rate (symbols). Solid line is prediction of Eq. (3.12).

based on the reaction-diffusion equation with mutation agrees well (within error bars) with measurements.

3.4 Discussion

Information propagation via replication into physically adjacent sites can be succinctly described by a reaction-diffusion equation. Such a description has been used in the description of *in vitro* evolution of RNA replicating in Qβ-replicase [11, 28], as well as the replication of viruses in a host environment [46]. The same equation is used to describe the wave behaviour of different strains of *E. Coli* bacteria propagating in a petri dish [7], even though the means of propagation in this case is motility rather than replication.

Sanda allows the investigation of large populations of self-replicating strings of code, and the observation of non-equilibrium effects. The propagation of information was observed for a broad spectrum of relative fitness, ranging from the diffusion regime where the fitnesses are the same through regimes where the difference in fitness led to sharply defined wavefronts propagating at constant speed. The dynamics of information propagation led to the determination of a crucial time scale of the system which represents the average time for the system to return to an equilibrium state after a perturbation. This relaxation time depends primarily on the size of the system, and the speed of information propagation within it. Equilibration can only be achieved if the mean time between (non-lethal) mutations is larger than the mean relaxation time. Thus, a sufficiently large system will never be in equilibrium. Rather, it is inexorably driven far from equilibrium by persistent mutation pressure.

For artificial living systems such as the one investigated here, it is possible to formulate an approximate condition which ensures that it will (on average) never equilibrate, but rather consist of regions of local equilibrium that never come into informational contact. From the timescales mentioned above, we determine that the number of cells N in such a system must exceed a critical value:

$$N > \left(\frac{2\,v(f)}{R_\star\,a}\right)^{2/3} , \tag{3.16}$$

where $R_\star$ is the rate of *non-lethal mutations*, $v(f)$ the velocity of information waves (Fisher velocity), and a the lattice spacing (assuming a mean time between non-lethal mutations $t_\star \approx (N\,R_\star)^{-1}$).

Beyond the obvious advantages of a non-equilibrium regime for genomic diversity and the origin of species, such circumstances offer the fascinating opportunity to investigate the possibility of nonequilibrium pattern formation in (artificial) living systems. However, the most interesting avenue of investigation opened up by such artificial systems is that of the study of the fundamental characteristics of life itself. Since it is widely believed that many of the processes that define life, including evolution, occur in a state which is far from equilibrium, to study such processes it is necessary to have systems which exhibit the properties of life we are interested in and that can be quantitatively studied in a rigorous manner in this regime. The availability of artificial living systems as experimental testbeds that can be scaled up to arbitrary population sizes on massively parallel computers is a step in this direction.

Chapter 4 Binning

When dealing with event distributions best plotted on single log or double log scales (such as exponential and power law distributions), care must be taken in the proper binning of the experimental data. Say we are interested in the probability distribution $P(n)$ of an event distribution over positive integer values of n. We conduct N trials, resulting in a data set $Q(n)$ of number of events observed for every n value. For ranges of n where the expected or observed number of events for each n is much higher than 1, normally no binning is required. However, for ranges of n where $Q(n)$ or $P(n)$ is small, binning is necessary to produce both statistically significant data points, and intuitively correct graphical representations. A constant bin size has several drawbacks: One must guess and choose an intermediate bin size to serve across a broad range of parameter space, and the shape and slopes of the curve (even in a double log plot) are distorted [4].

These disadvantages can be overcome by using a variable bin size. However, choosing bin sizes for variable binning can be time-consuming and arbitrary—different choices will lead to different conclusions. I propose two related methods of systematically determining appropriate variable bin sizes. Both methods lead to binned data which help in correctly visualizing the underlying distribution (slopes and shapes are conserved). First, I introduce the *Data Threshold Method*, which requires no *a priori* knowledge about the distribution, and is a good predictor of the underlying distribution. However, when there are few data points, the *Template Threshold Method*, explained in Section 2 is often more reliable. For both methods, a range of the threshold value should be tried and the best threshold value (neither over- or under-binning) chosen.

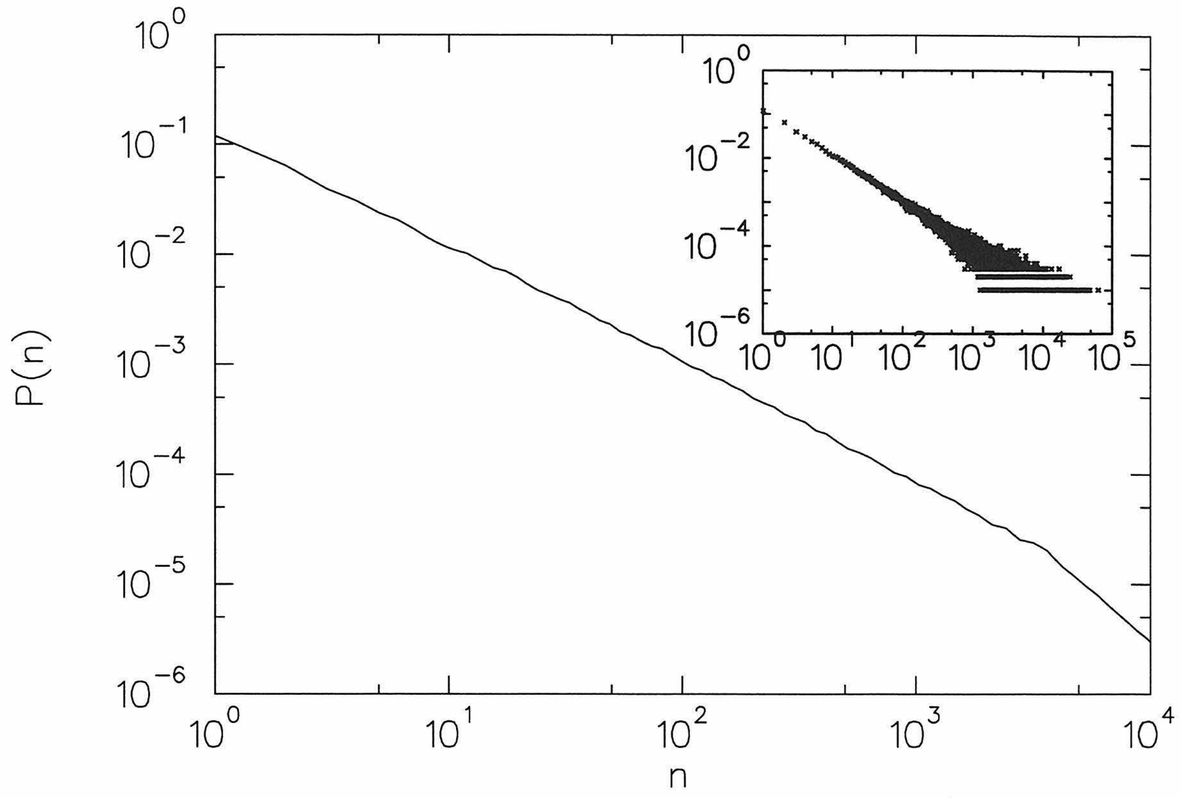


Figure 4.1: Binned avalanche size distribution for the BTW sandpile ($h \to 0$). The inset shows avalanche size distribution data after 100,000 avalanches. The main panel shows the same data binned using the data threshold method with $T = 1000$. Overlaying this figure over Fig. 4.2 (which is the same data for 16 million avalanches) shows no discernible differences between the predictions made by binning and the conclusions given by more data. The shape of the distribution through $n \sim 10^4$, especially the kink at $n \sim 5000$, is clearly shown by the binned distribution.

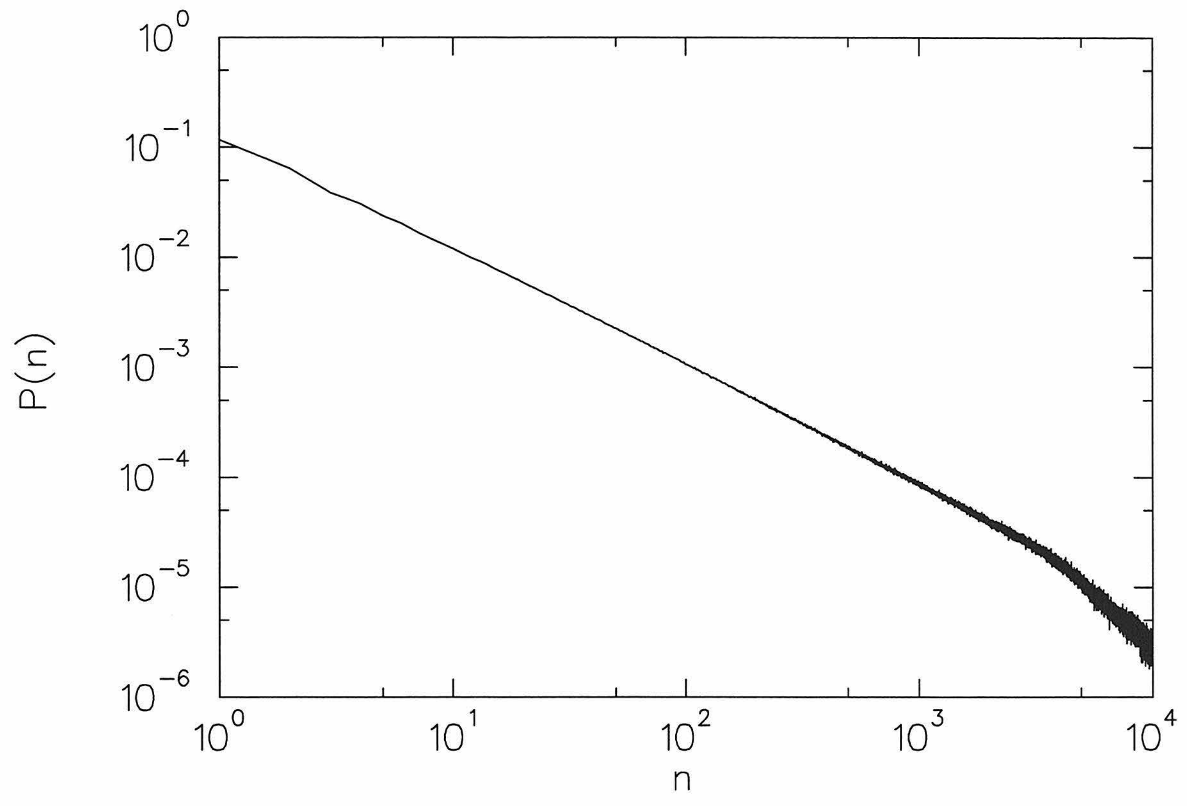


Figure 4.2: Avalanche size distribution in the 2-d BTW sandpile model with driving rate $h \to 0$. The lattice size for these simulations was 100×100 (note the cutoff in the distribution at $n \sim 5000$ due to system size effects). The data is unbinned and involves $\sim$16 million avalanches.

4.1 Data Threshold Method

For the data threshold method, start by selecting a threshold value T. Starting from $n = 1$ and proceeding to higher values, no binning is done until a value of n is found for which $Q(n) < T$. When such a value n_s is found, subsequent $Q(n)$ values are added to this amount until the sum of these values is greater than the threshold value,

$$\sum_{n=n_s}^{n_l} Q(n) > T. \tag{4.1}$$

We then have a bin size $(n_l - n_s + 1)$, with value $\sum_{n=n_s}^{n_l} Q(n)$. When plotting, it is convenient to plot this as a single point at the midpoint of $[n_s, n_l]$, with an averaged value,

$$\left(\frac{n_s + n_l}{2}, \frac{\sum_{n=n_s}^{n_l} Q(n)}{n_l - n_s + 1} \right). \tag{4.2}$$

This yields a graphical representation with little distortion and good predictive power (Figs. 4.1,4.2). This binning procedure is continued until no more data remains to be binned.

Example 1. Table 4.1 shows results from sampling a probability distribution obeying a power law with exponent $\beta = -1$. 20 trials were made. The binned data is obviously a better visual representation than the unbinned data (Fig. 4.3).

4.2 Template Threshold Method

Unlike the data threshold method, the template threshold method uses a predicted probability distribution $P(n)$, or a reasonable surrogate. Again, we define a threshold value for fitting T. However, in this case, the bin sizes are determined by comparing values of the *expected distribution*

$$E(n) = P(n) \times N \tag{4.3}$$

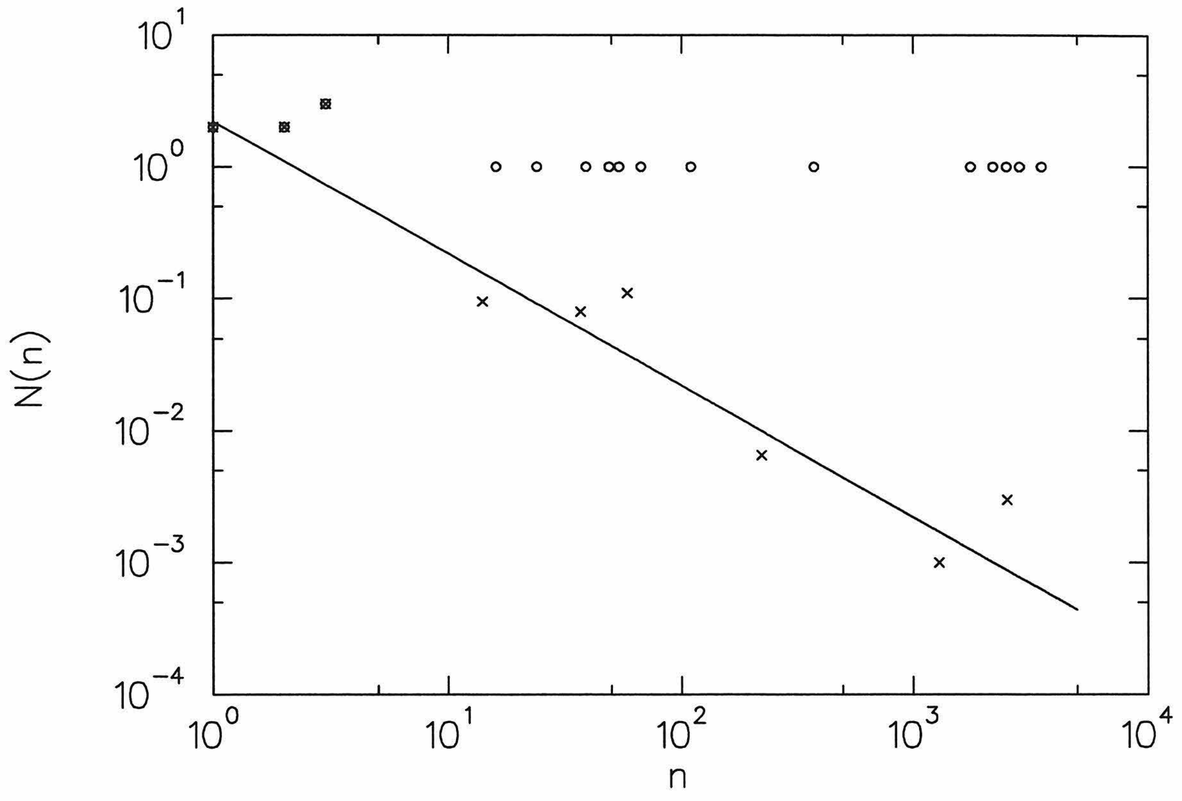


Figure 4.3: An example of data threshold binning. The circles are unbinned data, the crosses are data binned using data threshold binning with $T = 2$, and the solid line is the probability distribution the data was drawn from.

n	Number	Bin Sum	Bin Size	Bin Average	Bin Midpoint
1	2	2	1	2	1
2	2	2	1	2	2
3	3	3	1	3	3
16	1	2	21	0.095	14
24	1				
39	1	2	25	0.08	37
49	1				
54	1	2	18	0.11	58.5
67	1				
110	1	2	308	0.0065	221.5
375	1				
1758	1	2	1821	0.001	1286
2196	1				
2503	1	2	652	0.003	2522.5
2848	1				
3518	1				

Table 4.1: An example of data threshold binning. Note that the last data point is not used in the binned data.

to T. Starting from $n = 1$ and proceeding to higher values, no binning is done until a value of n is found for which $E(n) < T$. When such a value n_s is found, subsequent $E(n)$ values are added to this amount until the sum of these values is greater than the threshold value,

$$\sum_{n=n_s}^{n_l} E(n) > T. \tag{4.4}$$

We then have a bin of $[n_s, n_l]$ with corresponding size $(n_l - n_s + 1)$. The average value associated with this bin is

$$\frac{\sum_{n=n_s}^{n_l} Q(n)}{n_l - n_s + 1}. \tag{4.5}$$

This procedure is repeated until the data is exhausted. For this method, the data may be graphically represented either as a single point per bin (as in the data threshold method above), or as a point (showing the associated average value) for each measured

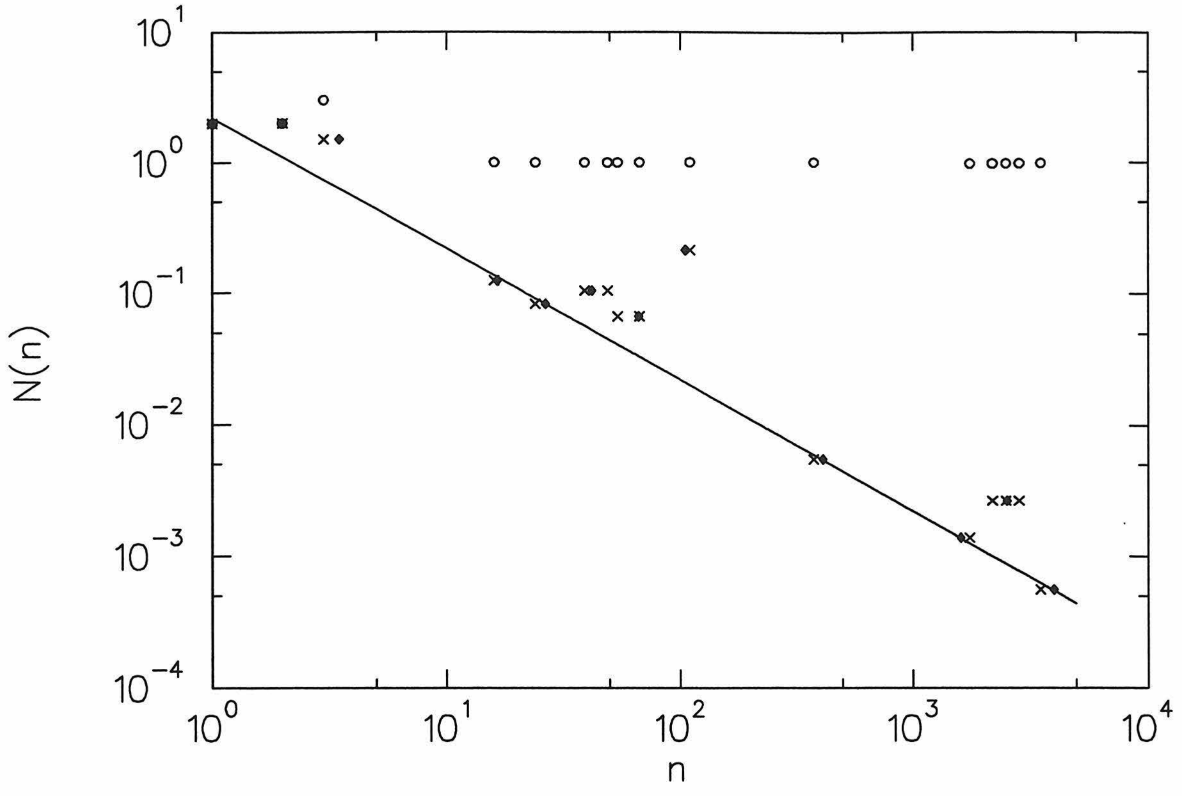


Figure 4.4: An example of template threshold binning. The circles are unbinned data, the crosses are data binned using data threshold binning with $T = 1.0$ and plotted with one point per unbinned data point, the diamonds are binned data plotted at bin midpoints, and the solid line is the probability distribution the data was drawn from.

(non-zero) data point $Q(n)$.

Example 2. The same data as in Example 1 are shown binned using the template threshold method in Table 4.2, Table 4.3, and Fig. 4.4. The template function used was the actual underlying distribution. However, a decent guess at the underlying distribution would have served just as well.

Bin Start	Bin End	Bin Size	Bin Midpoint	Bin Sum	Bin Average
1	1	1	1	2	2
2	2	1	2	2	2
3	4	2	3.5	3	1.5
5	7	3	6	0	0
8	12	5	10	0	0
13	20	8	16.5	1	0.125
21	32	12	26.5	1	0.0833
33	51	19	42	2	0.105
52	81	30	66.5	2	0.0667
82	128	47	105	1	0.213
129	202	74	165.5	0	0
203	319	117	261	0	0
320	503	184	411.5	1	0.00543
504	793	290	648.5	0	0
794	1250	457	1022	0	0
1251	1970	720	1610.5	1	0.00139
1971	3105	1135	2538	3	0.00264
3106	4893	1788	3999.5	1	0.00056

Table 4.2: Bin values in an example of template threshold binning. The bins were chosen with the aid of a template function with $T = 1.0$.

n	Number	Averaged Value	Bin Midpoint
1	2	2	1
2	2	2	2
3	3	1.5	3.5
16	1	0.125	16.5
24	1	0.0833	26.5
39	1	0.105	42
49	1		
54	1	0.0667	66.5
67	1		
110	1	0.213	105
375	1	0.00543	411.5
1758	1	0.00139	1610.5
2196	1		
2503	1	0.00264	2538
2848	1		
3518	1	0.00056	3999.5

Table 4.3: An example of template threshold binning. Note that all data points are utilized.

Chapter 5 An Extension to the Bak-Tang-Wiesenfeld Sandpile Model

Sandpile models with finite driving rates are limited by the restriction of one tumble per site per update. In this chapter, I describe a natural extension to the Bak, Tang, and Wiesenfeld sandpile model which removes this restriction and allows investigation of the dynamics of the model at high driving rates.

5.1 Introduction

The Bak, Tang, and Wiesenfeld (BTW) sandpile model [10] is defined on a d-dimensional lattice. Each site on the lattice has an *energy* z_i associated with it. A "grain" of energy of size 1 is dropped on a random site i and if the resultant energy of that site is greater than a critical energy ($z_i > z_c = 2d - 1$), the site transfers energy to its neighbours;

$$\begin{aligned} z_i &\rightarrow z_i - z_c, && (5.1) \\ z_n &\rightarrow z_n + \frac{z_c}{2d}. && (5.2) \end{aligned}$$

If the energy of a neighbour becomes supercritical through this process, the neighbour in turn *tumbles*. A series of tumbles (an *avalanche*) can result from the dropping of a single grain, ending only when all sites are again just critical or subcritical (Fig. 5.1). These tumbles are carried out in lockstep, each tumble takes exactly the same amount of time (an *update*), and the transport time of energy between sites is ignored. The only dissipation comes at the edges of the lattice where grains may "fall off." If this process is carried on long enough and on a large enough lattice, the system reaches a stationary state where the distribution of sizes of avalanches (total number of tumbles resulting from the dropping of one grain) and several other statistical properties of the system obey power law distributions. It was originally suggested that this self-organization was an inherent property of the system, while it now seems established that the system is actually tuned by waiting until avalanches are over before dropping new grains—this is equivalent to allowing non-local interactions [37, 20].

So far, we have ignored dissipation and assumed a vanishing driving rate—grains are added only after all sites have finished tumbling and the current avalanche is over. This is obviously not a physically realistic system. The effects of finite dissipation and a non-vanishing driving rate force the sandpile from its critical state—the avalanche size distribution is no longer a power law (this behaviour is explored in detail in Chapter 6). When the dissipation rate ϵ (number of grains falling off per grain travelling between sites) is larger than the driving rate h (probability of a site having a grain dropped on it per tumble update), the sandpile is in a stationary state where the avalanche size distribution starts to diverge from power law and the number of tumbles per site per update is less than one. Previously, the regime where $h \gtrsim \epsilon$ (Region B in Fig. 5.2) was considered trivial and uninteresting [42]. However, by refinement and extension of the BTW sandpile model to allow for multiple overlapping avalanches, we show that this is not the case and that the dynamics in Region A are continued into Region B.

5.2 Overlapping Avalanches

Let's first clarify the propagation of avalanches when two avalanches add grains to the same site during the same update. If more than one grain arrives at a site i at the same update, we order them randomly and the grain which by its addition causes

$$z_i = z_c + 1 = 2d \tag{5.3}$$

(the addition of this grain causes the site to become just supercritical) is defined as the grain which triggered the tumble (Fig. 5.3). The $2d$ grains at this site which

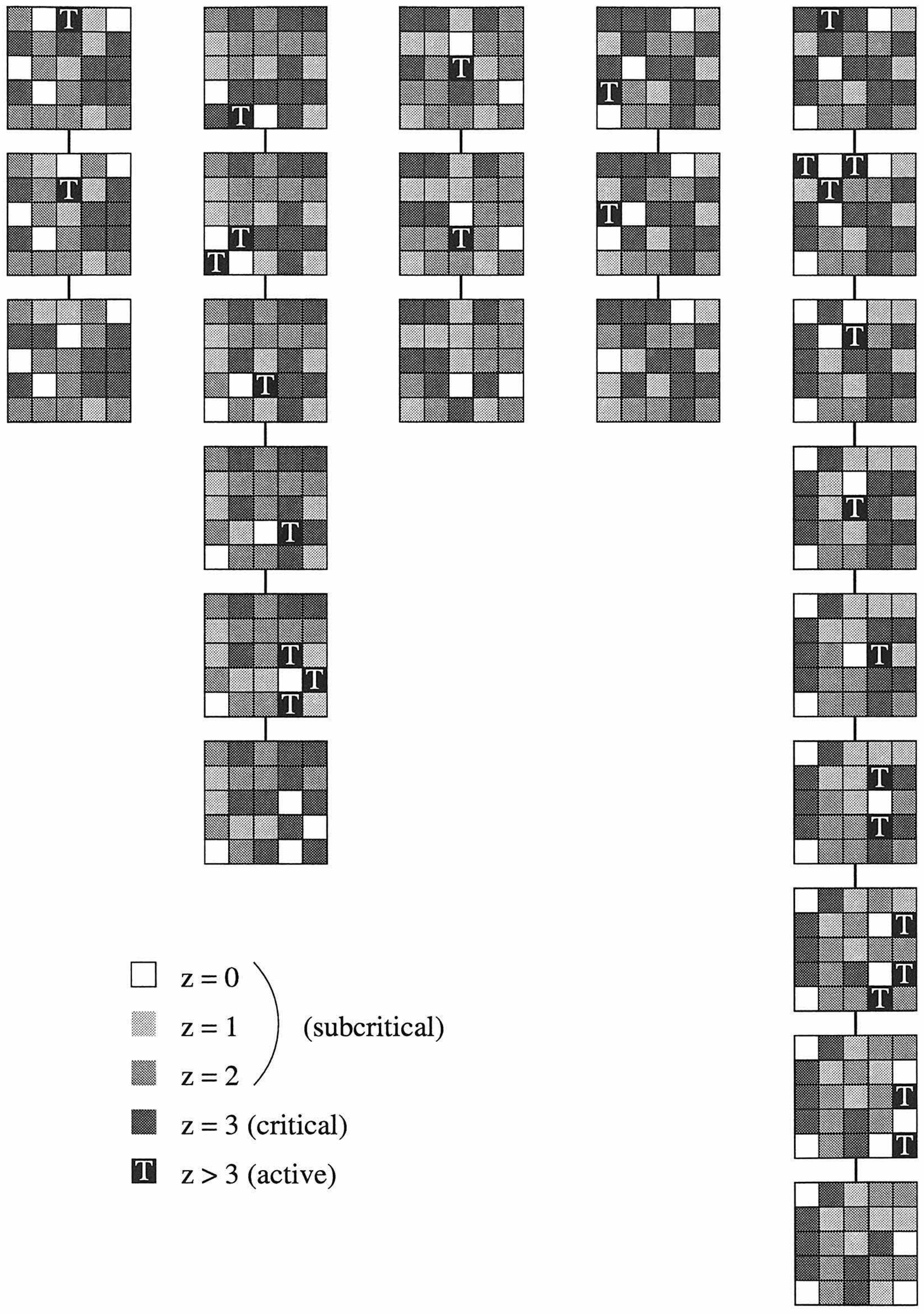


Figure 5.1: Avalanches in the 2-d BTW sandpile model. The total number of tumbles is (from left to right) n = 2, 8, 2, 2, 14. The second avalanche has 2 tumbles in its second update, 1 in its third update, and so on. Compare with the branching process trees of Fig. 6.1.

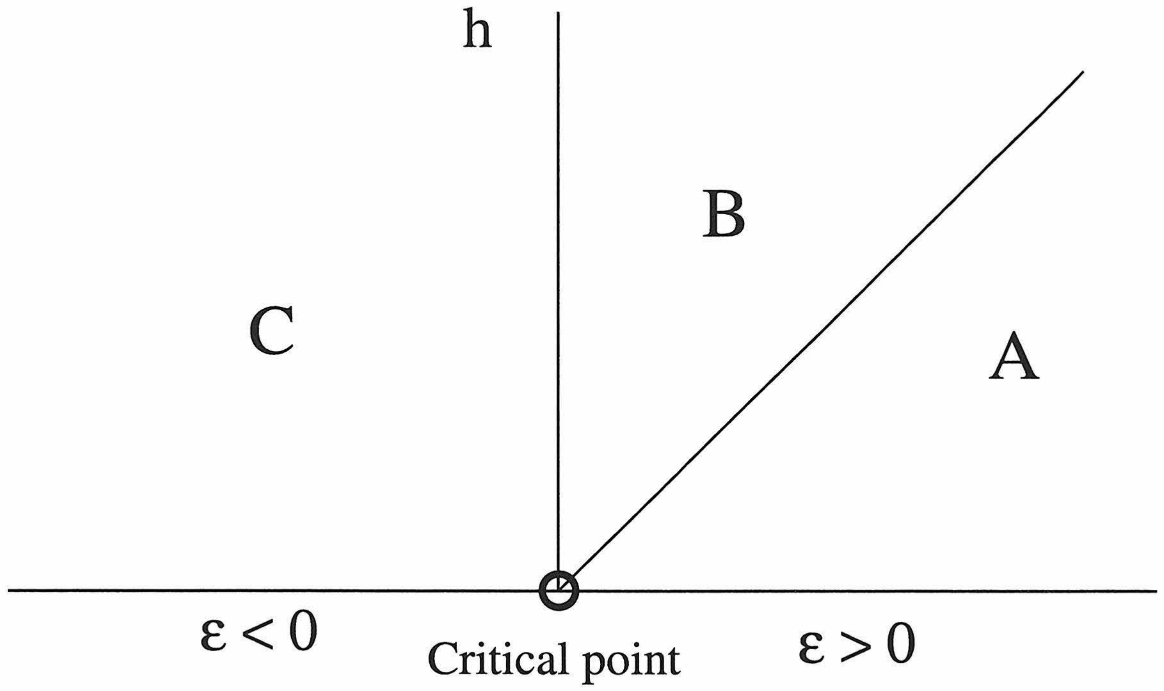


Figure 5.2: Sandpile model regimes. The critical point is for $h \rightarrow 0$ and $\epsilon \rightarrow 0$. Region A has $h \lesssim \epsilon$, while Region B has $h \gtrsim \epsilon$. Previously, Region A was considered the only region with interesting dynamics. However, extensions to the BTW model show that the dynamics of sandpiles is nontrivial and continuous throughout Regions A and B.

tumble as a result of this added grain we term *effect grains* of this grain, and the added grain the *cause grain* of the newly transported grains. This allows us to follow a chain of cause and effect for any transported grain back to a single grain dropped on the lattice. The size of an avalanche is defined as the number of tumbles caused by a single dropped grain (the *ancestor grain*), its effect grains, the effect grains' effect grains, and so on.

We now refine the model to allow more than one tumble per site per update. This is done by changing the condition for a tumble to

$$z_i = z_c \times n = 2d \times n \qquad (n = 0, 1, 2, \ldots). \tag{5.4}$$

Every $2d$-th grain (cause grain) causes a tumble in which $2d$ grains (effect grains) are transferred to neighbouring sites at the next update (Fig. 5.3). In this way, during one update a site can have multiple tumbles, of the same avalanche or of different avalanches.

5.3 Discussion

These two refinements permit meaningful discussion of avalanche dynamics in sandpiles driven with finite driving rate $h \gtrsim \epsilon$. As can be seen in Fig. 5.4, the dynamics of the system are continuous from the critical point to Region A to Region B: The extended model is a natural extension to the BTW model and to the BTW model with finite driving rates.

The extended model introduced here may be thought of as corresponding to a more physical situation than the original BTW sandpile model. The extended model incorporates finite driving rates, and finite and stochastic transport times of grains between sites. The model is still not completely realistic as it imposes an arbitrary periodicity on update (tumble) times. Whether the same dynamics would be observed in an even more physically realistic continuous-time model where this periodicity is not enforced is an interesting question.

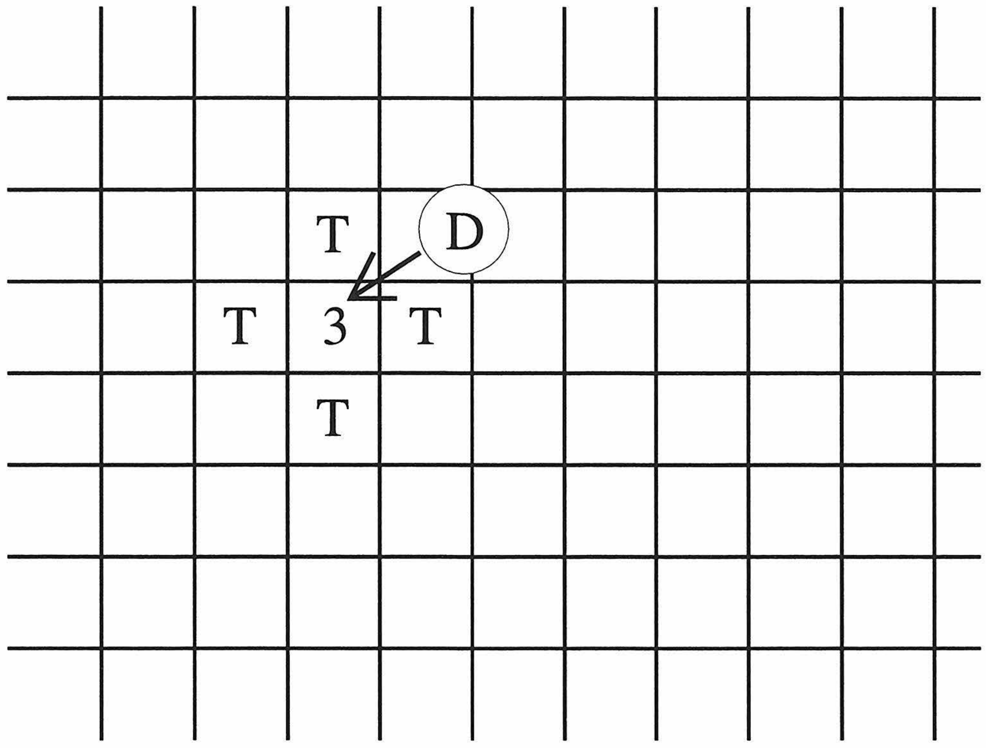


Figure 5.3: Multiple tumbles per update in an extension to the BTW sandpile model. When a grain of sand is dropped (D) onto a critical site (3) and all its neighbouring sites are tumbling (T) during the same update, the critical site will tumble twice in the next update. With a high enough driving rate, multiple tumbles per site per update become the norm. All grains being added to the site are ordered randomly.

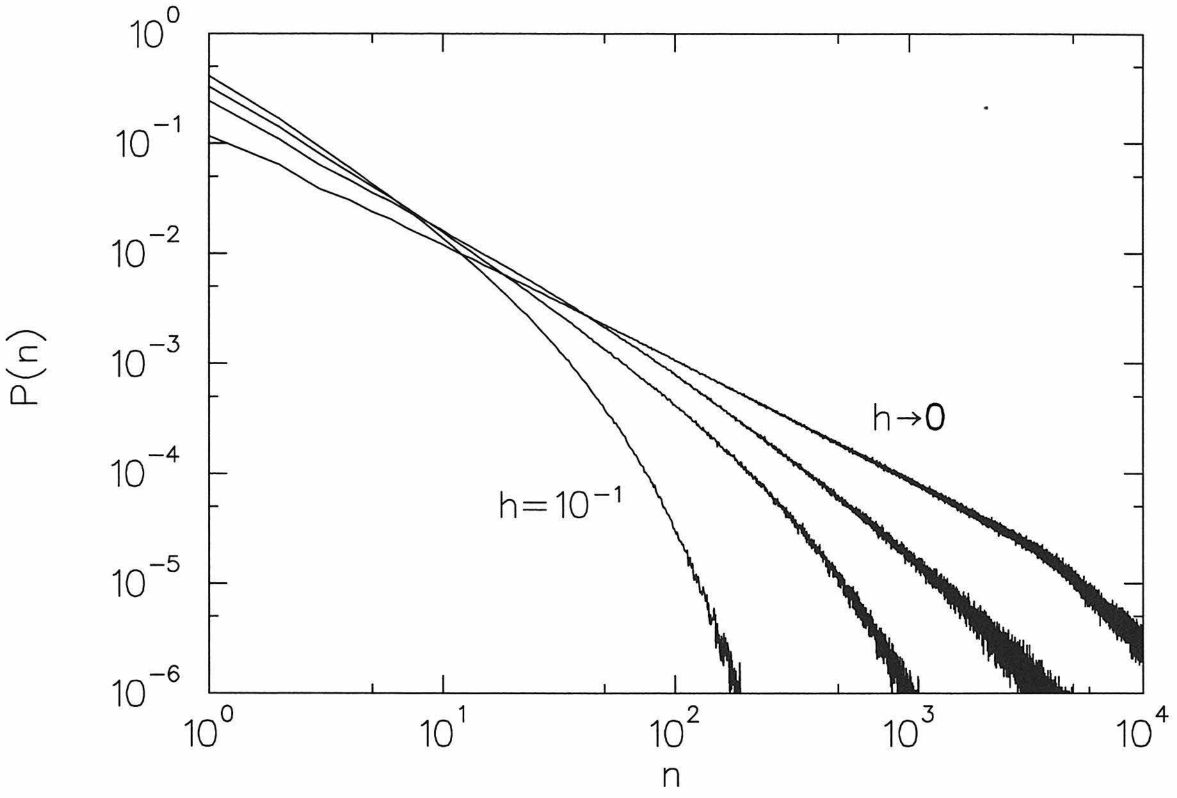


Figure 5.4: Avalanche size distribution in the 2-d BTW sandpile model with finite driving rates: $h = 0, 10^{-3}, 10^{-2}, 10^{-1}$. $h \to 0$ is nearly on the critical point of Fig. 5.2, $h = 10^{-3}$ is in Region A, and $h = 10^{-2}, 10^{-1}$ are in Region B. The transition from Region A to Region B is smooth and involves no sudden changes in the dynamics of the sandpile. The lattice size for these simulations was 100×100.

Chapter 6 Scale-Free Behaviour

Scale-free dynamics in physical and biological systems can arise from a variety of causes. Here, I explore a branching process which leads to such dynamics. I find conditions for the appearance of power laws and study quantitatively what happens to these power laws when such conditions are violated. From this branching process model, I predict the behaviour of three systems which seem to exhibit near scale-free behaviour—rank-frequency distributions of number of subtaxa in biology, abundance distributions of genotypes in an artificial life system, and avalanche sizes in the Bak-Tang-Wiesenfeld sandpile model. I find that the rate of introduction of competition determines the shape of the distributions in all three cases.

6.1 Introduction

Scale-free distributions, or *power laws*, have been observed in a variety of biological, chemical and physical systems. Such distributions can arise from different underlying mechanisms, but always involve a separation of scales, which forces the distribution to take a standard form. Scale-free distributions are most often observed in the distribution of sizes of events (such as the Gutenberg-Richter law [22]), the distribution of times between events (e.g., the inter-event interval distribution in neuronal spike trains [39]), and frequencies. An example of the latter is the well-known and ubiquitous $1/f$ noise. Some systems are even more interesting because they seem to exhibit self-organization or self-tuning, concomitant with scale-free behaviour as an inherent and robust property of the system, not due to the tuning of a control parameter by the experimenter.

Two systems to which such spontaneous scale-free behaviour has been attributed are sandpile models and taxon creation in biological systems. The former has served as the paradigm of "self-organized criticality" (SOC) [10], while the latter, manifested in the form of near power law shapes of rank-abundance curves, has been advanced as evidence of a fractal geometry of evolution [12, 13].

A much simpler system where power laws are observed is the random walk [38]. For example, the waiting times t for first return to zero of the simple random walk in one dimension (starting at $x = 0$, at each time step, $x(t+1) = x(t) + 1$ with probability 1/2 and $x(t+1) = x(t) - 1$ with probability 1/2) have a probability distribution $\sim t^{-3/2}$. Closely related to random walks, branching processes [23] can also create power law distributions. They have been used to model the dynamics of many systems in a wide variety of disciplines, including demography, genetics, ecology, physiology, chemistry, nuclear physics, and astrophysics. Here, we use a branching process to model the creation and growth of evolutionary taxa, and the propagation of avalanches in SOC sandpile models.

In Section 6.2, I examine the properties of the *Galton-Watson* process. I find that this process can generate power laws by appropriate tuning of a control parameter, and examine the dynamics of the system both at the critical point and away from it. In Section 6.3, I apply this branching process model to various systems, including the taxonomical rank-frequency abundance patterns of evolution and the avalanche size distribution of sandpile models, and discuss the universality of their underlying dynamics. Finally, in Section 6.4, I discuss the implications of this work, including a discussion of the order and control parameters for the branching process and its applications, and suggest further questions.

6.2 The Branching Process

The Galton-Watson branching process was first introduced in 1874 to explain the disappearance of family names among the British peerage [44]. It is the first branching process in the literature, and also one of the simplest. Consider an organism which replicates. The number of replicants (*daughters*) it spawns is determined probabilistically, with p_i ($i = 0, 1, 2 ...$) being the probability of having i daughters. Each daughter replicates (with the same p_i as the original organism) and the daughter's daughters replicate and so on. We are interested in the rank-frequency probability distribution $P(n)$ of the total number of organisms descended from this organism plus 1 (for the original organism), i.e., the historical size of the "colony" the ancestral replicant has given rise to (Fig. 6.1). Note that this is equivalent to asking for the probability distribution of the length of a random walk starting from 1 and returning to 0 with step sizes given by $P(\Delta n) = p_{i-1}$ $(i = 0, 1, 2...)$ [6].

The abundance distribution $P(n)$ can be found by defining a generating function

$$F(s) = \sum_{i=1}^{\infty} P(i)s^i. \tag{6.1}$$

This function satisfies the relationship

$$F(s) = s\sum_{i=0}^{\infty} p_i[F(s)]^i, \tag{6.2}$$

from which each $P(n)$ can be determined by equating coefficients of the same order in s [23]. This result can also be written as

$$P(n) = \frac{1}{n}Q(n, n-1) \qquad (k \geq 1), \tag{6.3}$$

where $Q(i, j)$ is defined as the probability that j organisms will give birth to a total of i true daughters [38]. However, these approaches are not numerically efficient, as the calculation of $P(n)$ for each new value of n requires recalculation of each term in the result.

For the present purposes, let us approach the problem in a different manner. Let $P_{k|j}$ be the probability that given j original organisms, we end up with a total of k organisms after all organisms have finished replicating. Obviously,

$$P_{k|j} = 0 \qquad (k > j), \tag{6.4}$$

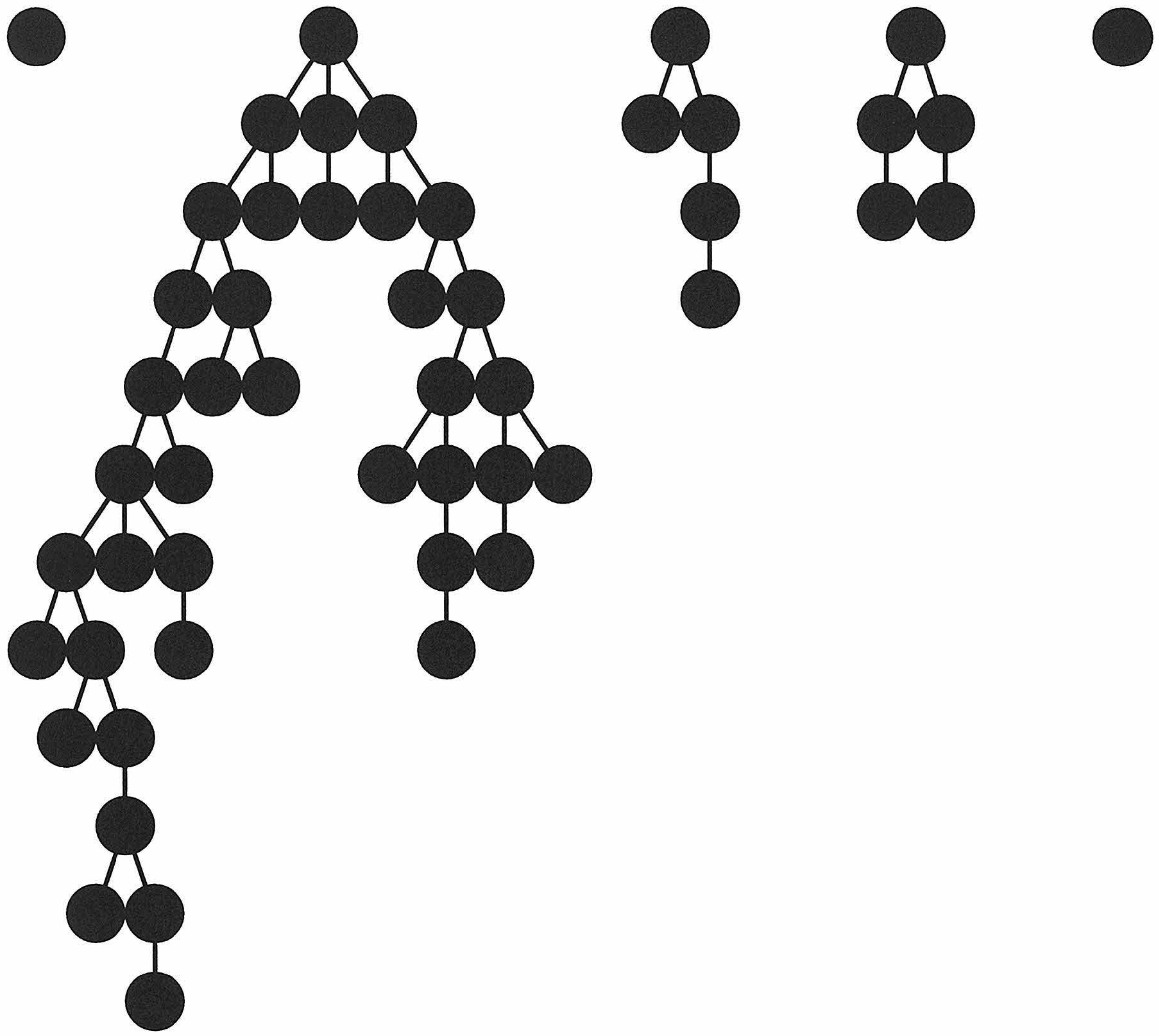


Figure 6.1: Branching process trees from a branching process with $p_0 = 0.5$, $p_1 = 0.2$, $p_2 = 0.2$, and $p_3 = 0.1$. The total number of organisms is (from left to right) $n = 1$, 39, 5, 5, 1. The second tree has 3 organisms in its second generation, 5 in its third, and so on.

since it is impossible to have less total organisms than one starts out with, and

$$P_{1|1} = p_0, \tag{6.5}$$

i.e., the probability for one organism to have no daughters. A little less obviously,

$$\begin{aligned} P_{k|1} &= \sum_{j=1}^{k-1} p_j P_{(k-1)|j} \qquad (k \geq 2), & (6.6) \\ P_{k|j} &= \sum_{i=1}^{k-1} P_{i|1} P_{i|(j-1)} \qquad (j \geq k > 1). & (6.7) \end{aligned}$$

These equations allow us to use dynamic programming techniques to calculate $P(n)$ ($= P_{n|1}$), significantly reducing the computational time required. Also, from Eq. (6.6), we can write

$$\frac{P_{n|1}}{P_{(n-1)|1}} = p_1 + p_2 \frac{P_{(n-1)|2}}{P_{(n-1)|1}} + p_3 \frac{P_{(n-1)|3}}{P_{(n-1)|1}} + \ldots . \tag{6.8}$$

Since, for $n \to \infty$, $P_{n|j}$ is uniformly decreasing, we see

$$\frac{P(n)}{P(n-1)} = \frac{P_{n|1}}{P_{(n-1)|1}} \to C \quad \text{as } n \to \infty, \quad (C \leq 1) \tag{6.9}$$

where C is a constant. C is an indicator of the asymptotic behaviour of $P(n)$ as $n \to \infty$. If $C < 1$, the probability distribution is asymptotically exponential, while for $C = 1$, the probability distribution is a power law with exponent $-3/2$.

Let us now examine the behaviour of $P(n)$ when $n \lesssim 10^4$, the more relevant case in the examples to follow. Using Eqs. (6.4)-(6.7), we can numerically calculate $P(n)$ for different sets of p_i. We define m as the expected number of daughters per organism, given a set of probabilities p_i;

$$m = \sum_i i \cdot p_i. \tag{6.10}$$

We see that m (the *control parameter*) is a good indicator of the shape of the proba-

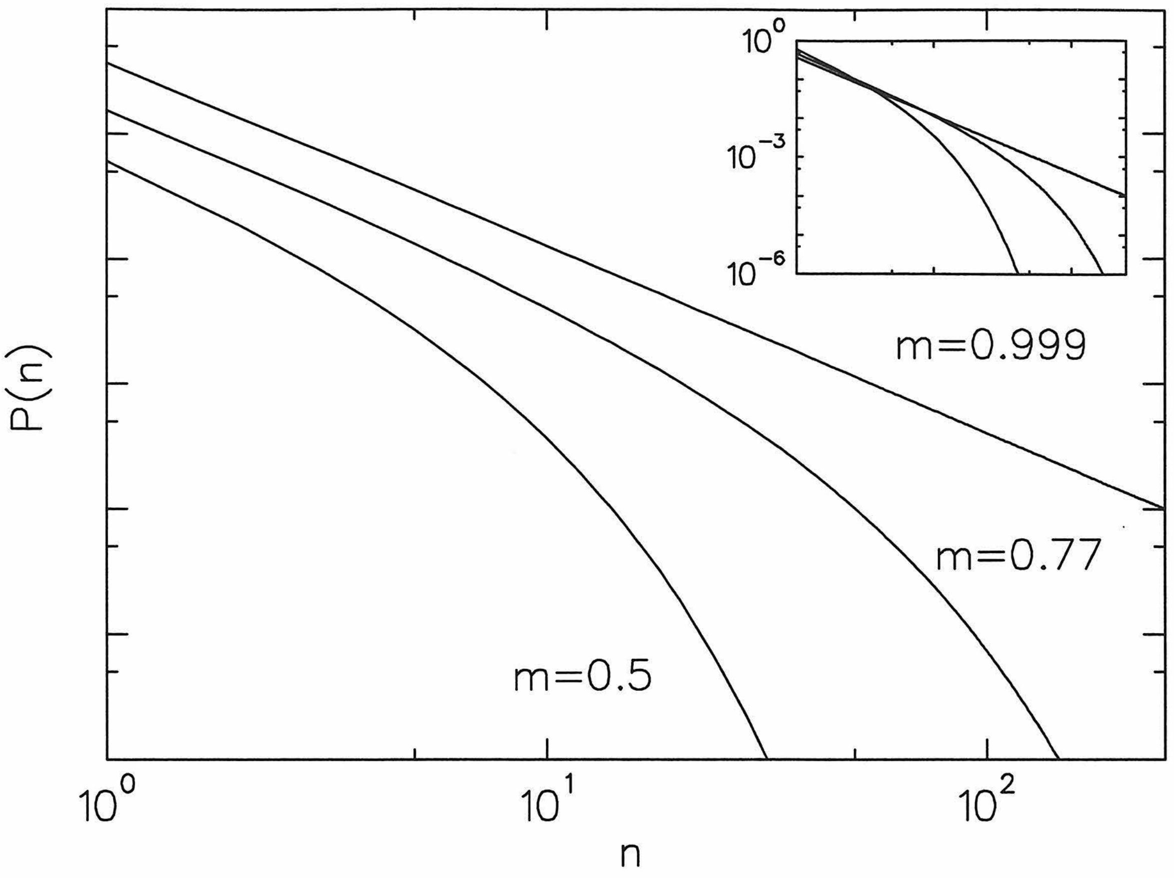


Figure 6.2: Predicted abundance patterns $P(n)$ of the branching model with different values of m. The curves have been individually rescaled to better show their shapes. The inset shows the same curves without rescaling.

bility curve (Fig. 6.2). When m is close to 1, the distribution is nearly a power law, and the further m diverges from 1, the further the curve diverges from a power law towards an exponential. When $m = 1/2$, the curve is completely exponential. For a population of organisms, m is a measure of the tendency for new generations to grow, or shrink, in number. A value of $m > 1$ indicates a growing generation size, which implies that there will, on average, be no generation with no daughters, and that the expected number of total organisms is infinite. Conversely, $m < 1$ indicates a shrinking population size: There will be a final generation with no daughters, and the expected number of organisms is finite. When $m = 1$, the system is in between the two regimes, and only then is a power law distribution found.

What if not all organisms share the same m? Interestingly, it turns out that even

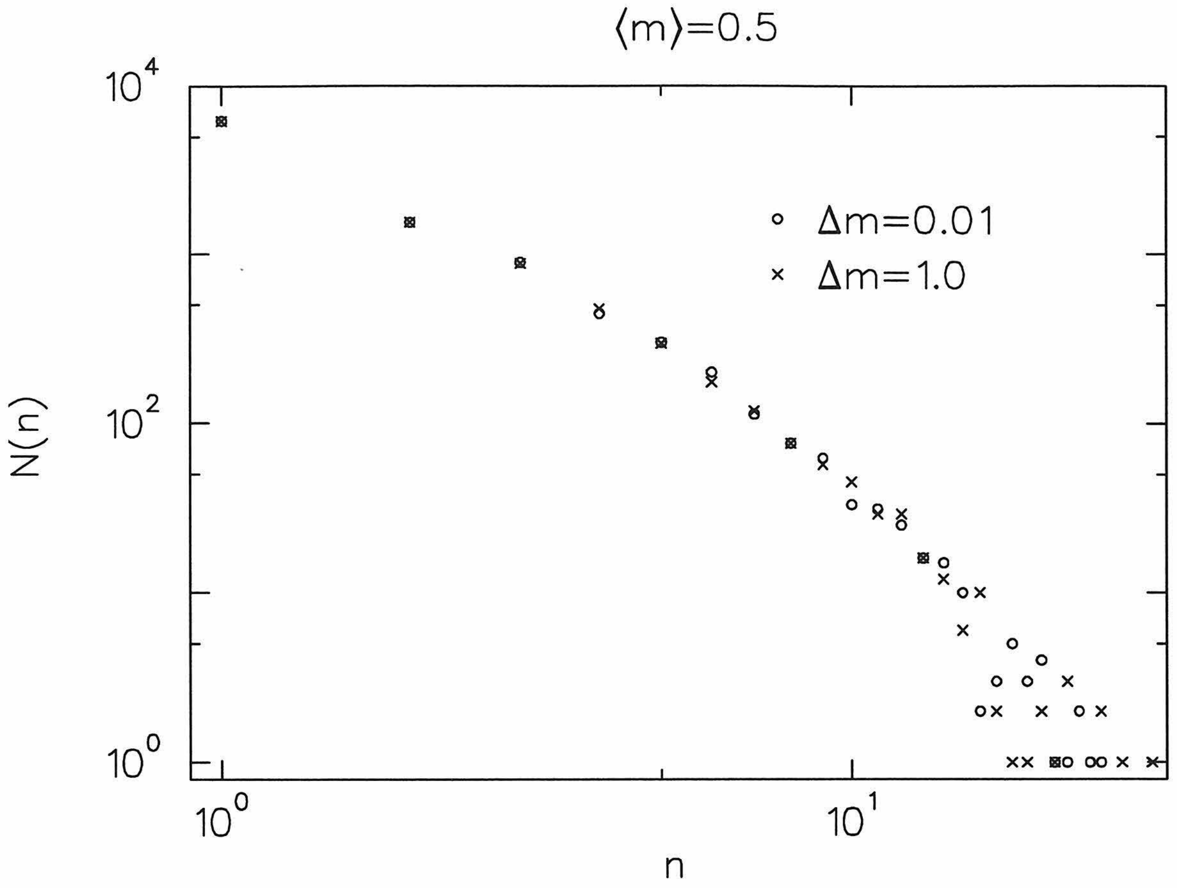


Figure 6.3: Abundance patterns obtained from two sets of numerical simulations of the branching model, each with $\langle m \rangle = 0.5$. m was chosen from a uniform probability distribution of width 1 for the runs represented by crosses, and from a distribution of width 0.01 for those represented by circles. Simulations where m and p_i are allowed to vary significantly and those where they are severely constricted are impossible to distinguish if they share the same $\langle m \rangle$.

if the p_i and m differ widely between different organisms, the rank-frequency curve is identical to that obtained by assuming a fixed m equal to the average of m across the organisms (Fig. 6.3), i.e., the variance of the p_i across organisms appears to be completely immaterial to the shape of the distribution—only the average $\langle m \rangle$ counts.

In the following section, I explore systems where the "organisms" are individual members of species, taxons in a taxonomical tree, or tumbling sites in a sandpile model, and m is the average number of exact copies an individual makes of itself, the average number of new taxons of the same supertaxon a taxon spawns, or the average number of new tumbles directly caused by a tumbling site.

6.3 Sanda-based Models

6.3.1 Neutral Model

Let us first examine a simple simulation—sanda with the neutral model instruction set—to test our analysis and lay the groundwork for the exploration of more complicated systems. Consider a population of organisms on a finite two-dimensional Euclidean grid, one organism to an intersection. Each organism can be *viable* or *sterile*. All viable organisms replicate approximately every τ time steps (there is a small random component to each individual's replication time to avoid synchronization effects), while sterile organisms do not replicate. For these experiments and the ones in the next section, when a sanda organism replicates, its daughter replaces the oldest organism in the parent's 9-site neighbourhood regardless of the replaced organism's viability or sterility. We define the *fidelity* F as the probability that the organism will create a daughter of the same type as itself, as well as the corresponding *genomic mutation rate* R $(= 1-F)$ at which it creates copies different from itself. The genomic mutation rate is actually the sum of two rates, a probability R_n for the daughter to be viable but to be of a new *genotype*, different from that of the parent (*neutrality rate*), and a probability R_s of the daughter being sterile. Of course, $R_n + R_s = R$. Note that all viable mutant daughters still share the same replication time τ—all mutations are neutral (See Fig. 6.4). Such a system gives rise to abundance distributions of power law and near-power law type that can be predicted as follows.

The total number of organisms is determined by the size of the grid. We write equilibrium conditions for the total number of organisms ρ_A, and for the total number of viable organisms ρ_V,

$$\Delta\rho_A \sim a\rho_V - \rho_A = 0, \tag{6.11}$$

$$\Delta\rho_V \sim v\rho_V - \rho_V = 0, \tag{6.12}$$

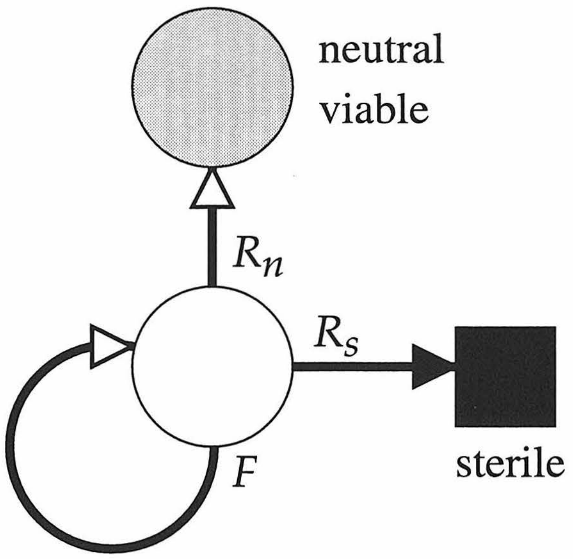


Figure 6.4: Neutral model replications and mutations. An organism's daughter is of the same genotype as the organism with probability F, it is of a new, viable genotype with probability R_n, and it is sterile with probability R_s such that $F + R_n + R_s = 1$.

where a is the average number of daughters (viable and sterile) a viable organism has, and v is the average number of viable daughters a viable organism has. Introducing m—the average number of true daughters (daughters which share the parent's genotype) for a viable organism—we see that

$$v = \frac{F + R_n}{F} m = (F + R_n)a. \tag{6.13}$$

From Eqs. (6.11)-(6.13), we obtain steady state solutions for a and m,

$$\begin{aligned} a &= \frac{F^{-1}}{1 + \frac{R_n}{F}}, & (6.14) \\ m &= \frac{1}{1 + \frac{R_n}{F}}. & (6.15) \end{aligned}$$

Knowing the values of a and m (or conversely, F and R_n) is sufficient to determine the shape of the abundance distribution. Fig. 6.5 shows abundance data for two neutral model runs with differing values of R_n (and consequently m), along with predicted distributions (which use only R_n and F as parameters) based on the branching model.

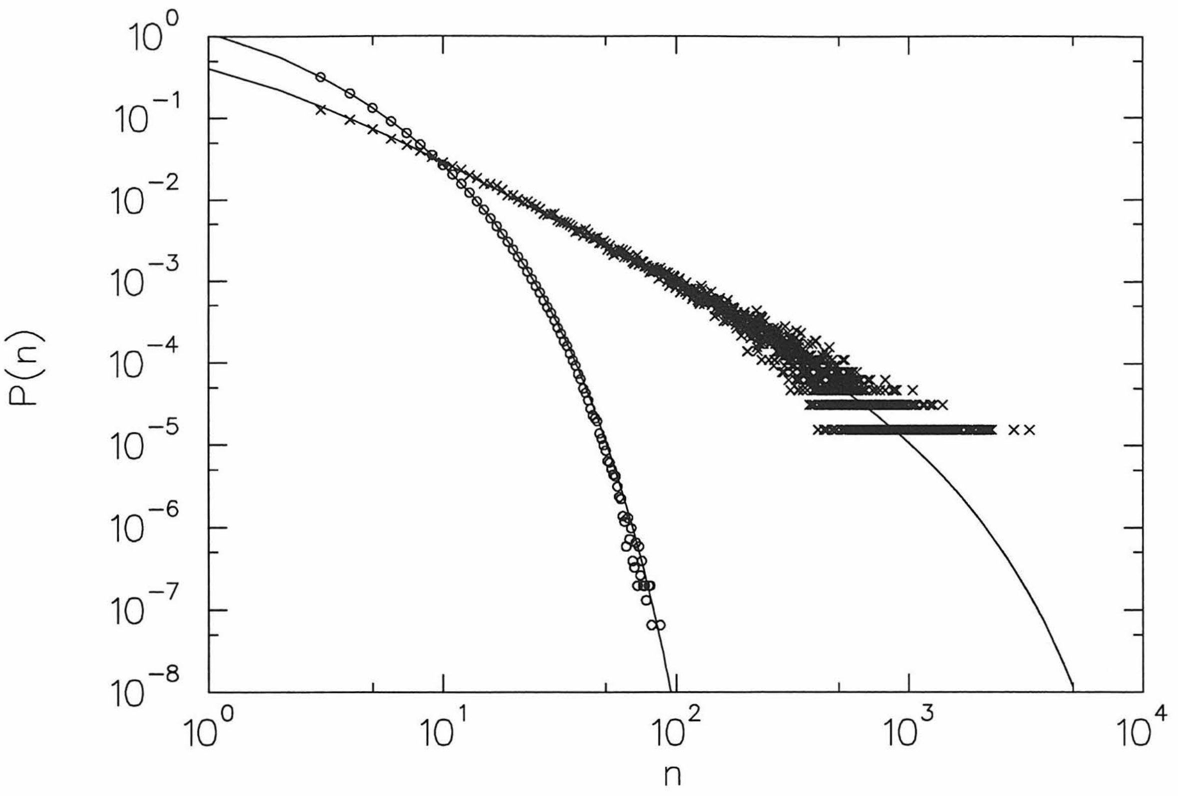


Figure 6.5: Abundance distributions and predicted curves for two neutral model runs. The run shown by circles ($\sim$ 1.5 million data points) had a grid size of 3000×3000, $F = 0.5$, and $R_n = 0.5$, while the one represented by crosses ($\sim$ 0.6 million data points) had a grid size of 100×100, $F = 0.2$, and $R_n = 0.1$. The branching process model predictions were made from values of F and R_n (there were no free parameters), and are accurate across a broad range of parameters.

Although the distribution patterns are very different, both are fit extremely well by the branching process model's predicted curves. In Eq. (6.15), note that R_n is the rate of influx of new genotypes (and therefore new competitors for space), while F is the rate of growth of existing genotypes. The value of m is determined by the ratio of these two rates. Unless the total number of creatures is increasing, $m \leq 1$ ($m = 1$ if and only if $R_n \to 0$ and new competing genotypes are introduced at a vanishing rate).

6.3.2 Non-neutral Model

The next system is sanda with the default instruction set. Compared to the neutral model above, the organisms are no longer simple, and instead each has a complex genome consisting of a string of assembly language-like instructions (Fig. 2.3). Each organism independently executes the instructions of its genome, and this genome determines the organism's replication time τ. Unlike the neutral model, the model allows non-neutral mutations which lead to new viable genotypes with both lower and higher replication times than the parent.

The system and the instructions are designed so that the organisms can self-replicate by executing certain sequences of instructions. The replication time of an organism is not a predetermined constant, rather it is determined by the genotype of the organism: Organisms can replicate faster or slower than other competing organisms with different genotypes. For an organism to successfully replicate, its genotype must contain information which allows the organism to allocate temporary space (memory) for its daughter, replicate its genotype (one instruction at a time) into this temporary space, and then to divide, placing its daughter in a grid site of its own (Fig. 2.3). As in the neutral model, on division, the daughter replaces the oldest organism in its parent's 9-site neighbourhood.

Organisms, depending on their genotype, may not be able to replicate (may be sterile) or may only be able to replicate imperfectly (resulting in no true daughters). Also, the copy instruction, which the organisms must use to copy instructions from their own code into that of their nascent daughters, has a probability of failure (*copy mutation rate*), which can be set by the experimenter. When the copy instruction fails, an instruction is randomly chosen from all the instructions available to the organisms (the *instruction set*) and written in the string location copied to. Copy mutations also lead to non-true daughters. The instruction set is robust; copy errors (mutations) induced during the replication of viable organisms have a non-vanishing probability of creating viable new organisms and genotypes. Indeed, by selecting for certain traits (such as the ability to perform binary logical operations) by increasing the rate at which instructions are executed in organisms which carry these traits, the system can be forced to *evolve* and find novel genotypes which contain more information (and less entropy) than their ancestors. Even without this external selection, the system evolves organisms (and genotypes) which replicate more efficiently in less executed instructions.

As a result of this evolution, the fidelity and neutral mutation rate are not fixed, but can vary with the length of an organism's genome and the instructions contained therein. Also, new genotypes formed by beneficial mutations that allow faster replication than previously existing genotypes will have (on average) an increasing number of organisms—$m > 1$—until the new, faster replicating genotypes fill up a sizable portion of the grid. All these factors combine to make predicting the abundance distributions for sanda much harder than for the neutral model.

Indeed, rather than being constant during the course of a sanda experiment, R_n and F will vary unpredictably as the population of organisms occupies different areas in genotypic phase space. Certain genotypes may be *brittle*, allowing very few mutations that result in new viable genotypes. The length of the organisms may change, changing both the genomic mutation rate and the neutrality rate. Genotypes exist which make systematic errors when copying, which decreases the fidelity. In short, the dynamics of these digital organisms are complex and messy, much like those of their biochemical brethren. These variations are observed at the same time across different organisms in the population, and are also observed with the progression of time. Still, we attempt to predict the abundance distributions by approximating the ratio of neutral mutations to true copies by the *observed* ratio of viable genotypes to total number of viable organisms ever created:

$$\frac{R_n}{F} \simeq \frac{N_g}{N_v}, \tag{6.16}$$

where N_g is the total number of viable genotypes observed during a sanda run and N_v is the total number of viable organisms. This relation should hold approximately under equilibrium conditions. Then, Eq. (6.15) becomes

$$m \simeq (1 + \frac{N_g}{N_v})^{-1}, \tag{6.17}$$

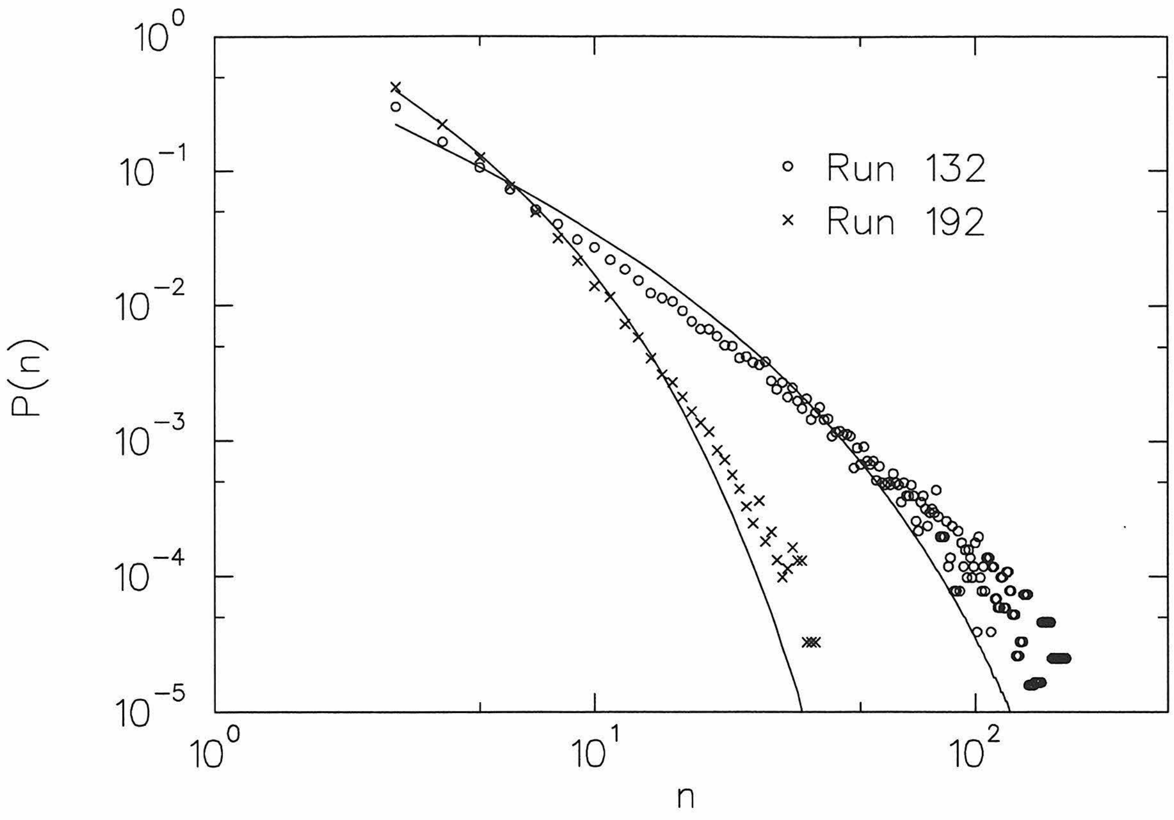


Figure 6.6: Abundance data from two sanda runs with predicted abundance curves. Both runs were started with the same initial genotype for all organisms, the same per-instruction copy mutation rate (γ), and the same grid size (100×100). Run 192's genotypes evolved into a regime of genotypic phase space with longer average length, and therefore lower fidelity F and higher neutrality R_n, than Run 132, resulting in the differences in the abundance distributions. The predicted curves were generated by approximating a representative value of R_n/F from the ratio of the number of viable genotypes to the number of viable organisms observed over the run. The data was binned using the template threshold method with $T = 1$ (see Chapter 4).

and from Eq. (6.14)

$$a = \frac{m}{F}. \tag{6.18}$$

The fidelity F is inferred from the average length l of genotypes during a run and the (externally enforced) per-instruction copy mutation rate γ, $F = (1 - \gamma)^l$. Because we estimate m and a from macroscopic observables averaged over the length of a run, I expect some error in these results due to the shifting dynamics of the evolution of genotypes as the system moves in genotypic phase space.

The abundance data from two different sanda runs are shown in Fig. 6.6 with the predicted abundance curves. The two runs shared the same grid size and per-instruction copy mutation rate, and were started with the same initial genotypes, but the runs evolved into different regions of genotypic phase space and consequently had significantly different statistics. Considering the many gross approximations made, the agreement between the predictions and the experimental data is surprisingly good (especially as no fitting is involved). Sanda is most closely related to an asexually replicating biological population, such as colonies of certain types of bacteria occupying a single niche. The genotype abundance distributions measured in sanda are analogous to the species or subspecies abundance distributions of its biological counterparts. In general, species abundance distributions are complicated by the effects of sexual reproduction, and of the localized and variable influences of other species and the environment on species abundances. However, I believe the branching model—used judiciously—can be helpful in the study of such distributions.

6.4 Evolution

For taxonomic levels higher than species, the rank-abundance distributions of number of subtaxa per taxon approximate power laws [47, 12, 13]. Yule [47] proposed a continuous time branching process model to explain these distributions at the generic level. He recognized that naturally observed distributions diverged from the power law predicted by his theory for equilibrium distributions, and hypothesized that this deviation was caused by a finite-time effect. I find that the branching process model generates the observed distributions and find that the distribution's deviation from power-law form is not caused by disequilibration (as Yule proposed), but rather that it is time-independent and determined by the evolutionary properties of the taxa of interest. The model predicts—with no free parameters—the rank-frequency distribution of number of families in fossil marine animal orders obtained from the fossil record. I find that near power-law distributions are statistically almost inevitable for taxa higher than species.

Rank-abundance distributions at taxonomic levels higher than species (e.g., the distribution of the number of families per order) are simpler to model than species abundance distributions, as the effects of the complications noted above are weak or nonexistent. I find that the available data is well fit by assuming no direct interaction or fitness difference between taxa. The shapes of rank-frequency distributions of taxonomic and evolutionary assemblages found in nature are surprisingly uniform. Indeed, Burlando has speculated that all higher-order taxonomic rank-frequency distributions follow power laws stemming from underlying fractal dynamics [12, 13]. I believe this conclusion is hasty: The divergence of the distributions from power law can be observed by applying appropriate binning methods to the data. (See Chapter 4.) Yule [47] attempted a branching process model explanation of these distributions, and claimed that the divergence from power law of rank-abundance patterns was transient and indicated a finite time since the creation of the evolutionary assemblage. The model indicates that this is not generally the case. I find that the divergence from power law is not a result of disequilibration, but is an inherent property of the evolutionary assemblage under consideration and that this divergence provides insight into microscopic properties of the assemblage (e.g., the rate of innovation).

Say, for example, that we are interested in the rank-frequency distribution of the number of families in each order for fossil marine animal orders. We assume that all new families and orders in this assemblage originate from mutations in extant families. Then, we can define rates of successful mutation R_f for mutations which create new families in the same order as the original family, and R_o for mutations which create an entirely new order. In this case, unlike the cases treated above, we assume $a \to \infty$; many, many individual births and mutations occur, but the proportion that are family- or order-forming is miniscule. Finally, assuming a quasi-steady state (the total numbers of orders and families vary slowly [31]), we rewrite Eq. (6.15),

$$m \simeq (1+\frac{R_o}{R_f})^{-1} \qquad (6.19)$$

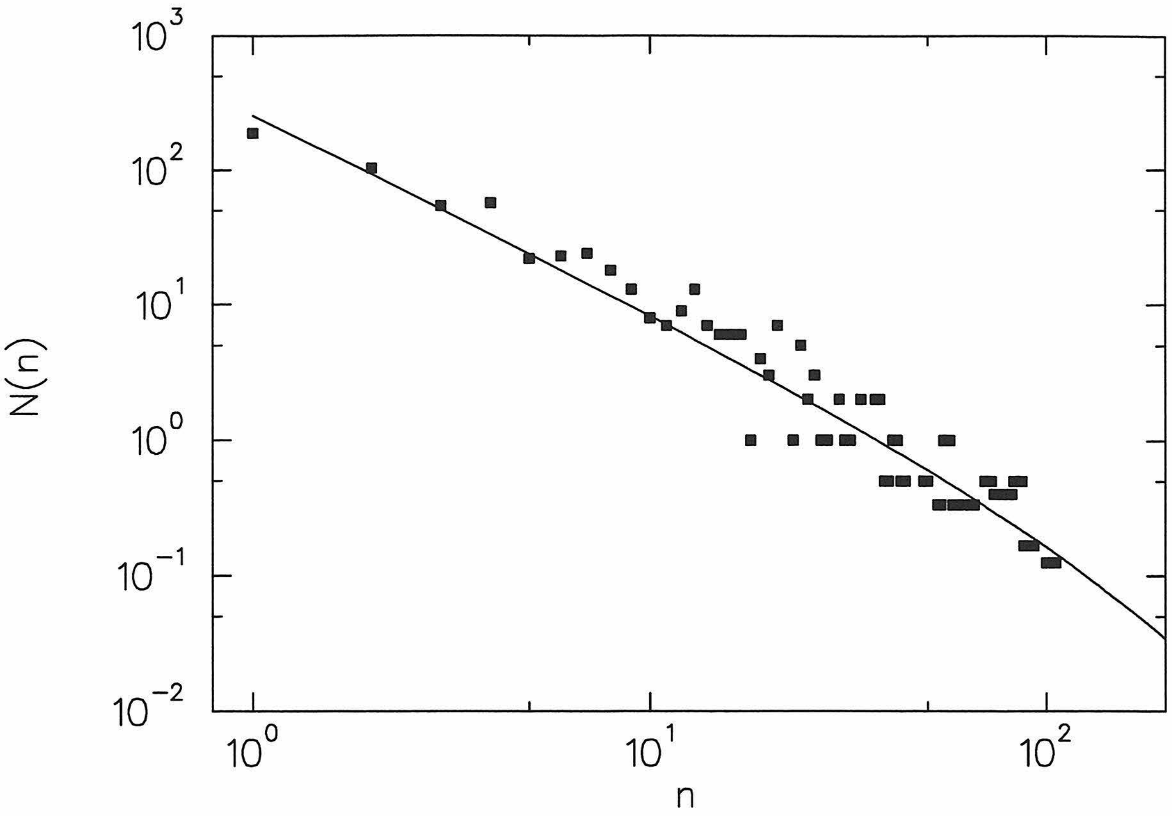


Figure 6.7: The rank-frequency distribution of fossil marine animal orders (squares) [34] and the predicted abundance curve (line). The predicted curve was generated—with no free parameters—by approximating R_n/F by $N_o/N_f = 0.115$. The empirical distribution agrees with the predicted curve with significance 0.12 using the Kolmogorov-Smirnov test. A Monte Carlo analysis shows that for a sample size of 626 (as we have here), the best fit $R_o/R_f = 0.135$ (Fig. 6.8) is within the 66% confidence interval of the predicted $R_o/R_f = 0.115$. The fossil data is shown binned using the template threshold binning method explained in Chapter 4 with $T = 1$.

$$\simeq \quad (1 + \frac{N_o}{N_f})^{-1}, \tag{6.20}$$

in terms of N_o and N_f, the total numbers of orders and families respectively. As in the previous systems studied, R_o is the rate of creation of new—and competing—orders, while R_f is the rate of growth of existing orders, and m is determined by their ratio.

Data for the abundance distribution of number of families in fossil marine animal orders [34] are shown in Fig. 6.7. I obtained values for N_o and N_f directly from the fossil data to generate the predicted curve with *no free parameters*. The agreement is very good, much better than that for the sanda runs where evolutionary parameters

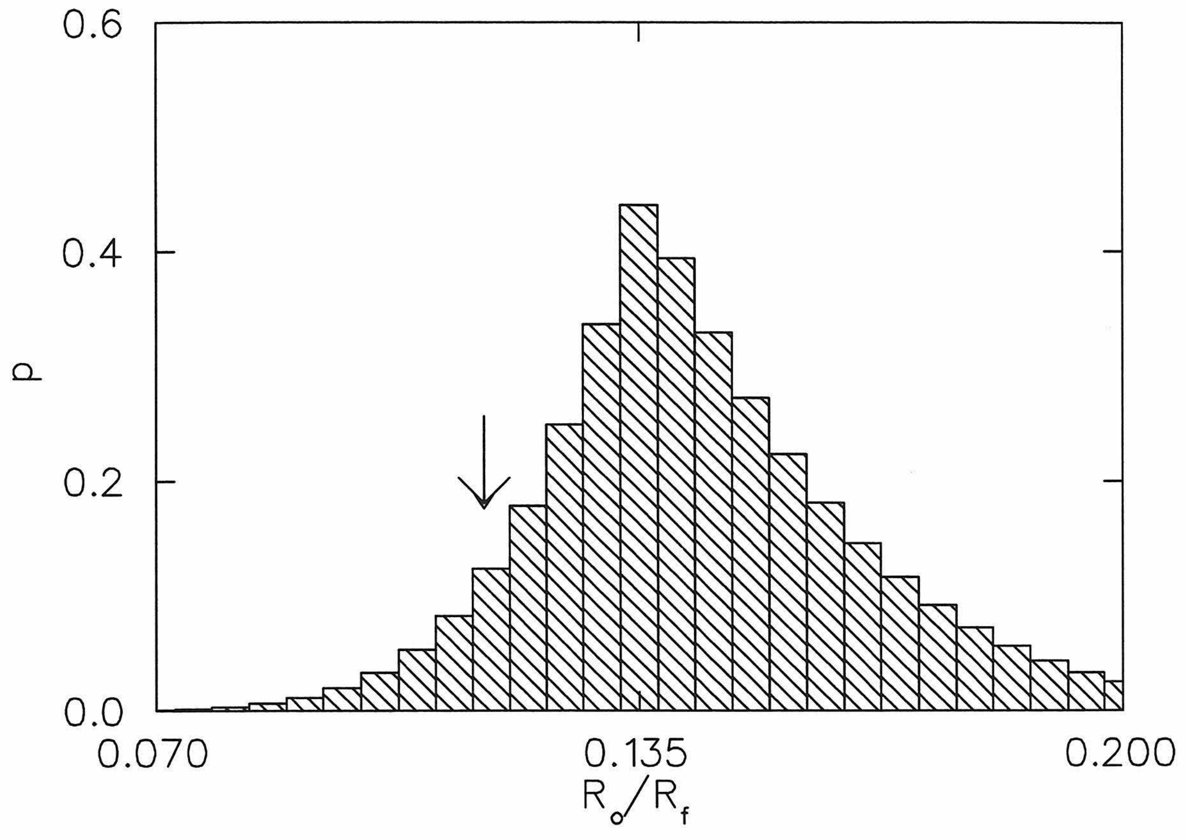


Figure 6.8: Kolmogorov-Smirnov (K-S) significance levels p obtained from comparison of the fossil data to several predicted distributions with different values of R_o/R_f, which shows that the data is best fit by $R_o/R_f = 0.135$. The arrow points to my prediction $R_o/R_f = 0.115$ where $p = 0.12$. The K-S tests were done after removal of the first point, which suffers from sampling uncertainties.

such as the fidelity F and the neutrality R_n were constantly changing. Comparing m and the resultant abundance curves with those obtained above for the rank-abundance distribution of sanda genotypes leads to the expected conclusion that the probability of creation of a new genotype in sanda per birth is much higher than the probability of a new family creating an order in natural evolution. Indeed, Burlando [12, 13] finds that a wide variety of taxonomic distributions are fit quite well by power laws ($m = 1$), although some of his figures seem to show an exponential tail such as that predicted by our model if $m < 1$. This seems to imply that actual taxonomic abundance patterns from the fossil record are characterized by a relatively narrow range of m near 1. This is likely within the model description advanced here. It is obvious that m can not remain above 1 for significant time scales as this would lead to an infinite number of subtaxa for each taxon. Even if, by a beneficial mutation, a new taxon has an evolutionary advantage over existing taxa, it soon fills up the available evolutionary phase space and must slow the increase in the number of its subtaxa. What about low m? I propose that low values of m are not observed for large (and therefore statistically important) taxon assemblages for the following reasons. If m is very small, this implies either a small number of total individuals of this assemblage, or a very low rate of beneficial taxon-forming (or niche-filling) mutations. The former might lead to this assemblage not being recognized at all in field observations. Either case will lead to an assemblage with too few taxons to be statistically tractable. Also, since such an assemblage either contains a small number of individuals or is less suited for further adaptation or both, it would seem to be susceptible to early extinction.

The branching model can—with appropriate care—also be applied to species-abundance distributions, even though these are more complicated than those for higher taxonomic orders for several reasons. Among these are the effects of sexual reproduction and the localized and variable effects of the environment and other species on specific populations. Historically, species abundance distributions have been characterized using frequency histograms of the number of species in logarithmic abundance classes. For many taxonomic assemblages, this was found to produce a humped distribution truncated on the left—a shape usually dubbed *lognormal* [29, 30,

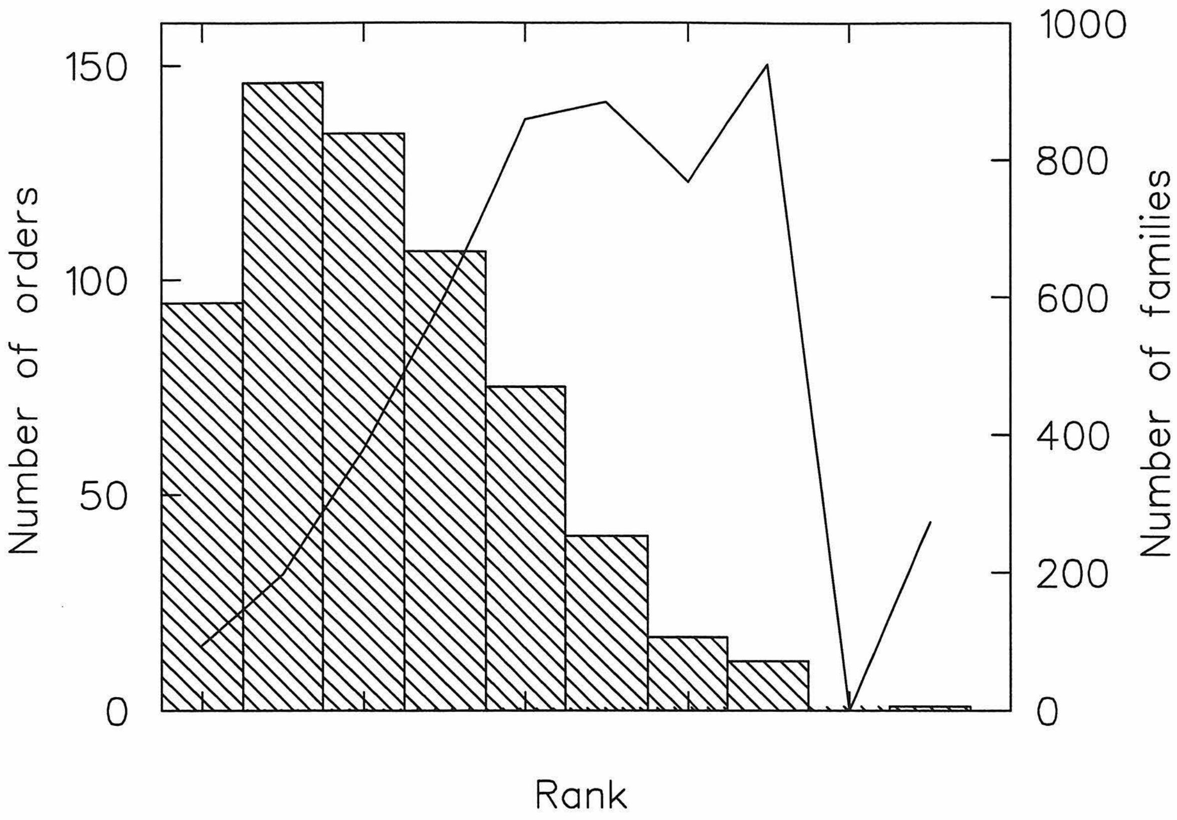


Figure 6.9: The abundance distribution of fossil marine animal orders in logarithmic abundance classes (the same data as Fig. 6.7). The histogram shows the number of orders in each abundance class (left scale), while the solid line depicts the number of families in each abundance class (right scale). Species rank-abundance distributions where the highest abundance class present also has the highest number of individuals (as in these data) are termed *canonical lognormal* [30].

40]. In fact, this distribution is not incompatible with the power-law type distributions described above. Indeed, plotting the fossil data of Fig. 6.7 in logarithmic abundance classes produces a lognormal (Fig. 6.9). Thus, species-abundance distributions may turn out not to be qualitatively as different from taxonomically higher-level rank-frequency distributions as expected. For species, m is the mean number of children each individual of the species has. (Of course, for sexual species, m would be half the mean number of children per individual.)

For species, m less than 1 implies that extant species' populations *decrease* on average, while m equal to 1 implies that average populations do not change. An extant species' population can decline due to the introduction of competitors and/or the decrease of the size of the species' ecological niche.

Let us examine the former more closely. If a competitor is introduced into a saturated niche, all species currently occupying that niche would temporarily see a decrease in their m until a new equilibrium was obtained. If the new species is significantly fitter than the previously existing species, it may eliminate the others. If the new species is significantly less fit, then it may be the one eliminated. If the competitors are about as efficient as the species already present, then the outcome is less certain. Indeed, it is analogous to a non-biased random walk with a possibility of ruin. The effects of introducing a single competitor are transient. However, if new competitors are introduced more or less periodically, then this would act to push m lower for all species in this niche and we would expect an abundance pattern closer to the exponential curve as opposed to the power-law than otherwise expected. This is analogous to the introduction of new competitors through viable mutations in sanda, where we also find a higher rate of viable mutations leads to distributions closer to exponential (see previous section).

If no new competitors are introduced but the size of the niche is gradually reduced, I expect the same effect on m and on the abundance distributions. Whether it is possible to separate the effects of these two mechanisms in ecological abundance patterns obtained from field data is an open question. An analysis of such data to examine these trends would certainly be very interesting.

So far, I have sidestepped the difference between historical and ecological distributions. For the fossil record, the historical distribution we have modeled here should work well. For field observations where only currently living groups are considered, the nature of the death and extinction processes for each group will affect the abundance pattern. In simulations and artificial-life experiments, I have universally observed a strong correlation between the shapes of historical and ecological distributions. I believe this correspondence will hold in natural distributions as well when death rates are affected mainly by competition for resources. The model's validity for different scenarios is an interesting question, which could be answered by comparison with more taxonomical data.

The branching process model allows us to reexamine the question of whether any type of special dynamics—such as self-organized criticality (SOC) [10]—is at work in evolution [36, 3]. While showing that the statistics of taxon rank-frequency patterns in evolution are closely related to the avalanche sizes in SOC sandpile models (examined in the next section), the present model clearly shows that instead of a subsidiary relationship where evolutionary processes may be self-organized critical, the power-law behaviour of both evolutionary *and* sandpile distributions can be understood in terms of the mechanics of a Galton-Watson branching process [42]. The mechanics of this branching process are such that the branching trees are probabilistic fractal constructs. However, the underlying stochastic process responsible for the observed behaviour can be explained simply in terms of a random walk [38]. For evolution, the propensity for near power-law behaviour is found to stem from a dynamical process in which $m \approx 1$ is selected for and highly more likely to be observed than other values, while the "self-tuning" of the SOC models is seen to result from arbitrarily enforcing conditions which would correspond to the limit $R_o/R_f \to 0$ and therefore $m \to 1$ (see next section).

6.5 Sandpile Models

The Bak, Tang, and Wiesenfeld (BTW) sandpile model [10] was introduced in Chapter 5. For the BTW sandpile, define ρ_c as the probability that any site is critical (one more grain added to that site will cause it to tumble). Then, it is easy to construct a mean field branching process, where the probability distribution of the number of nearest-neighbour sites a tumbling site will cause to tumble in the next update is given by

$$p_i = \binom{2d}{i} \rho_c^i (1 - \rho_c)^{n-1}. \tag{6.21}$$

This leads to

$$a \simeq 2d, \tag{6.22}$$

$$m = \sum_{i}^{2d} i p_i = 1, \tag{6.23}$$

and a predicted power law distribution for the size of avalanches $s(n)$, again obtained from Eqs. (6.4)-(6.7). In higher dimensions ($d \gtrsim 6$), the branching process model is expected to hold exactly and $s(n) \sim n^{-3/2}$. This is supported by numerical simulations. However, for lower dimensions, sandpiles will "interfere" with themselves, and a smaller exponent is found. Attempts to calculate the effects of this "final-state" interaction through renormalization have as yet not been completely successful.

So far, I have ignored dissipation and assumed an infinitesimal driving rate (i.e., allowed one avalanche to finish before another grain is dropped). If we define ρ_a and ρ_c as the proportions of sites which are active (tumbling) and stable (subcritical), $g \approx 2d$ as the number of nearest neighbours, h as the probability per update that any particular site will have a grain dropped on it (driving rate), and ϵ as the probability that a grain of tumbled sand will not reach a valid site, e.g., by falling off the edge of the lattice (dissipation rate), we see that $i = 1, 2, ...$ new active sites are generated by the tumbling of one active site with probabilities

$$\begin{aligned} q_i &= \sum_{k=i}^{g} P((g-k) \text{ grains dissipated}) P(i \text{ new active sites}|k) \\ &= \sum_{k=i}^{g} \binom{g}{k} \epsilon^{g-k} (1-\epsilon)^k \binom{k}{i} \rho_c^i (1-\rho_c)^{k-i}, \end{aligned} \tag{6.24}$$

while no active sites are generated with probability

$$q_0 = [\epsilon + (1-\epsilon)(1-\rho_c)]^g. \tag{6.25}$$

This gives us the control parameter for the branching process,

$$\begin{aligned} m &= \langle i \rangle \\ &= \sum_k k q_k \\ &= g(1-\epsilon)\rho_c \end{aligned} \tag{6.26}$$

which has a critical value $m = 1$. Assuming a steady state and a finite driving rate h, we write

$$\begin{aligned} \rho_a(t+1) &= \rho_a(t) m + h \rho_c(t) \\ &= \rho_a(t) \end{aligned} \tag{6.27}$$

and, substituting for ρ_c from Eq. 6.26, we find

$$m = (1 + \frac{h}{\rho_a g(1-\epsilon)})^{-1}. \tag{6.28}$$

Again, note that h is proportional to the rate of introduction of new avalanches, while $\rho_a g(1-\epsilon)$ is proportional to the rate of growth of existing ones. As in the simpler case where dissipation and driving were ignored, I expect that the branching process model will be quantitatively correct in higher dimensions. Indeed, such a mean field branching process model can be used to predict quantitative values of some sandpile exponents that hold in all dimensions [42]. Unfortunately, it is computationally very expensive to simulate high-dimensional sandpiles. Fig. 6.10 shows the results of simulating a two-dimensional BTW sandpile with finite driving rates from $h \to 0$ to $h = 10^{-1}$. As expected, higher driving rates h lead to lower m and distributions farther from power law. Other branching process treatments of sandpile models can be found in Ref. [42] and references therein.

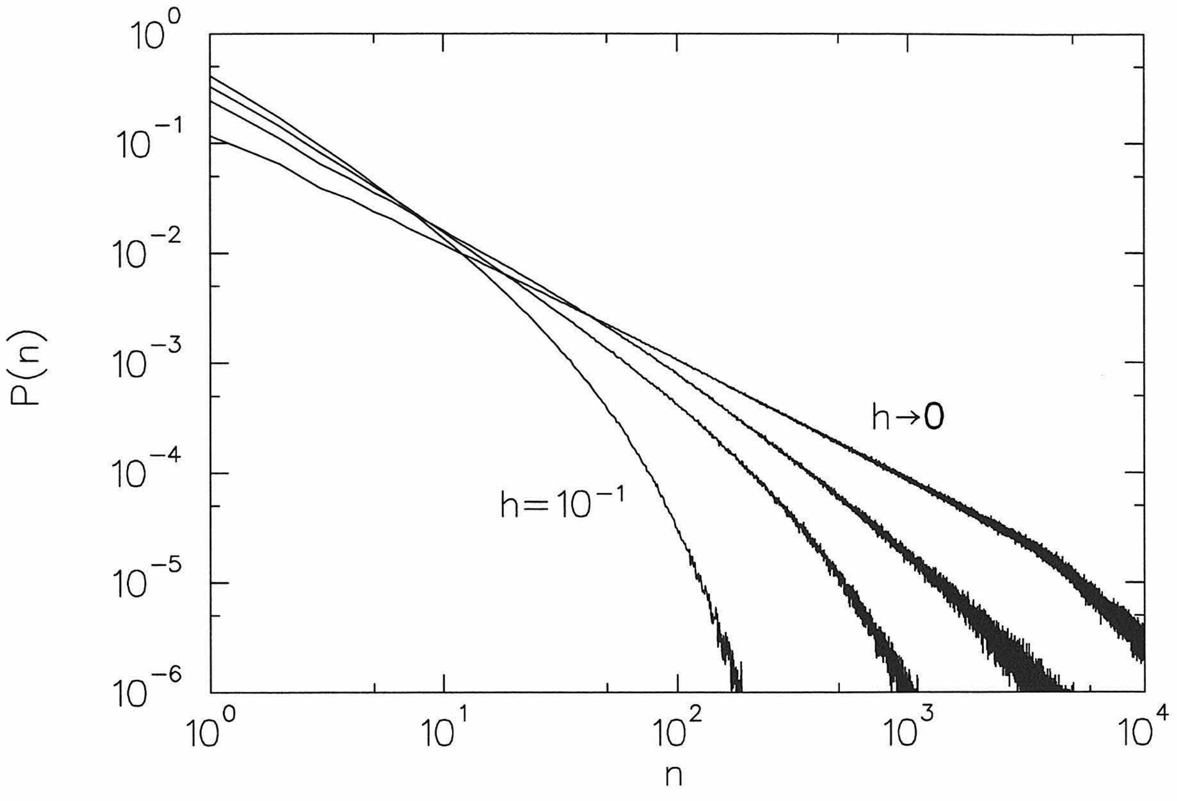


Figure 6.10: Avalanche size distribution in the 2-*d* BTW sandpile model with finite driving rates: $h = 0$, 10^{-3}, 10^{-2}, 10^{-1}. Higher driving rates lead to distributions farther from power law and closer to exponential, as predicted by the branching process model. The lattice size for these simulations was 100×100 (note the cutoff in the $h \to 0$ distribution at $n \sim 5000$ due to system size effects). Unfortunately, quantitative predictions can not be made for low-dimensional sandpiles (where "final-state" interactions exist), while simulating high-dimensional sandpiles is computationally prohibitive.

6.6 Discussion

The Galton-Watson branching process generates power law distributions when its control parameter $m = 1$. In all four of the systems I have examined above,

$$m = (1 + \frac{R_c}{R_p})^{-1} \tag{6.29}$$

is determined by the ratio of the rate of introduction of competitors R_c to the intrinsic rate of growth of existing assemblages R_p. As this ratio goes to 0, $m \to 1$ and the system becomes critical.

This relation can be translated into the standard relation between an *order parameter*

$$\alpha = \frac{R_c}{R_p} \tag{6.30}$$

and a new form for the control parameter

$$\mu = m^{-1}. \tag{6.31}$$

Writing α in terms of μ,

$$\alpha = \begin{cases} (\mu - \mu_c)^\beta & (\mu > \mu_c), \\ 0 & (\mu \leq \mu_c), \end{cases}$$

where $\mu_c = 1$ and $\beta = 1$ (Fig. 6.11). The order parameter represents the rate at which competition is introduced to the system (the strength of selection). A value of the control parameter $\mu < \mu_c$ implies a system with no competition and no selection—an exponentially growing population. Values of μ higher than μ_c indicate that new competition is always being introduced and that all existing species or avalanches must eventually die out. When $\mu = \mu_c$, competition is introduced at a vanishingly small rate, and we find the critical situation where separation of scales occurs. Interestingly, in all the systems studied, the order parameter has been the

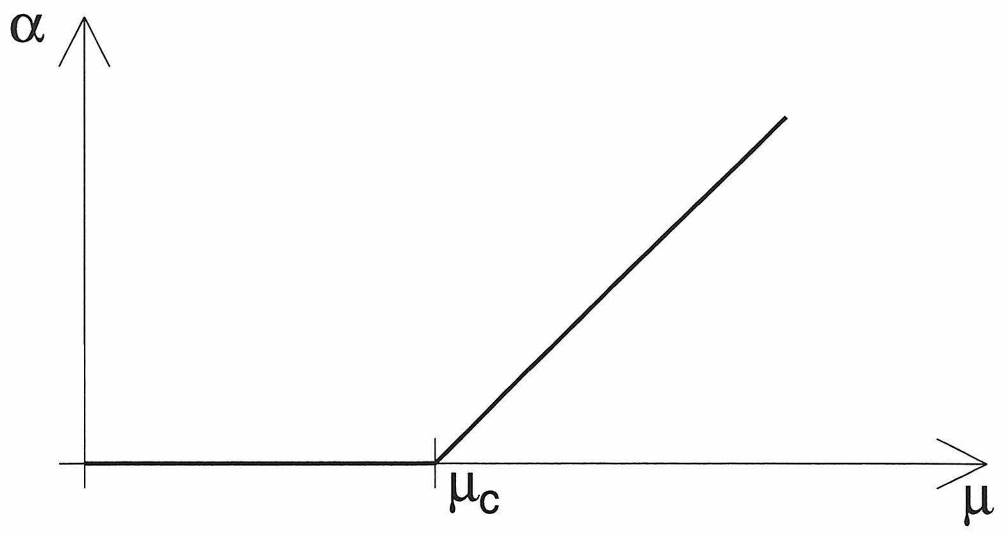


Figure 6.11: The order parameter α as a function of the control parameter μ. For μ below μ_c, the order parameter is 0—organisms (or events) in the system spawn greater and greater number of daughter organisms (events), and there is exponential growth. For $\mu > \mu_c$, competition from newly created organisms (events) stops abundances from growing without bound. $\mu = \mu_c$ marks the critical point where abundances can grow to infinity, but do not show exponential growth, and power law distributions arise.

easier to "control." Indeed, this feature of SOC sandpile models (tuning occurring at $\alpha \to 0$) may be their most important one.

For sandpile models, this α is arbitrarily set close to 0 by using large lattice sizes (reducing dissipation) and waiting for avalanches to finish before introducing new perturbations (resulting in an infinitesimal driving rate and a diverging diffusion coefficient). In simulations away from these arbitrary conditions, a loss of criticality is predicted by the model and observed in numerical simulations. Self-organized criticality and its sandpile models have stimulated research in many different fields and systems where near power law avalanche dynamics was observed. It seems that many of these systems should be mappable to branching processes, and that the fractal behaviour of these systems and the changes in their dynamics which follow from finite driving rates could be understood in terms of such. For the biological and biologically-inspired systems we have considered, the control parameter is not set arbitrarily at a critical value. However, the dynamics of the evolutionary process, in which it is much harder to effect large jumps in fitness and function than it is to effect small ones, lead to naturally observed values of α being small, especially for higher taxonomic orders. The dynamics of evolution act, robustly, to keep μ near μ_c. This in turn leads to a near power law pattern for rank-frequency distributions.

It would be beneficial to compare the predictions of the branching process (BP) model to high-dimensional sandpile simulations, where it should be quantitatively correct. Comparison of the BP model with more biological data is also desirable. For biological systems, there is a vast amount of empirical data, most of it, unfortunately, not in a form suitable for direct comparison to the BP model. Since the model allows a characterization of the abundance distributions with no free parameters, I believe it should be possible to deduct, from abundance distributions (and their divergence from power law), microscopic parameters of the system which created the distribution—e.g., driving rates in sandpiles, genomic and higher-order neutralities in nature. Species abundances are affected by many factors, but I believe that a careful application of the BP model (e.g., by comparison of collections of species with different ecological pressures) could yield insight.

I have shown that the apparent power laws of avalanches in SOC sandpile models, species-abundance distributions in artificial life systems, and rank-abundance distributions in taxonomy can be explained by modelling the dynamics of the underlying system with a simple branching process. This branching process model successfully predicts, with no free parameters, the observed abundance distributions—including their divergence from power law. This may allow the deduction of the microscopic parameters of the system directly from the macroscopic abundance distribution. I find that we can identify a control parameter—the average number of new events an event directly spawns, and an order parameter— the rate of introduction of competing events into the system, and that these are related in a form familiar from second order phase transitions in statistical physics.

Bibliography

[1] C. Adami, Artificial Life **1** 129 (1994).

[2] C. Adami, Physica D **80** 154, (1995).

[3] C. Adami, Phys. Lett. A **203** 29, (1995).

[4] C. Adami, *Introduction to Artificial Life* (Springer, New York, 1998).

[5] C. Adami and C. T. Brown, In R. A. Brook and P. Maes (Eds.), *Artificial Life IV*: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, edited by R. A. Brook and P. Maes, p.377. (MIT Press, Cambridge MA, 1994).

[6] C. Adami, C. T. Brown, and M. R. Haggerty, Lecture Notes in Artificial Intelligence **929**, 503 (1995).

[7] K. Agladze et al., Proc. Roy. Soc. Lond. B **253** 131, (1993).

[8] P. Alstrøm, Phys. Rev. A **38**, 4905 (1988).

[9] P. Bak, *How Nature Works: The Science of Self-Organized Criticality* (Springer-Verlag, New York, 1996).

[10] P. Bak, C. Tang, and K. Wiesenfeld, Phys. Rev. Lett. **59**, 381 (1987).

[11] G. J. Bauer, J. S. McCaskill, and H. Otten, Proc. Natl. Acad. Sci. USA **86**, 7937 (1989).

[12] B. Burlando, J. theor. Biol. **146**, 99 (1990).

[13] B. Burlando, J. theor. Biol. **163**, 161 (1993).

[14] J. Chu and C. Adami, in *Artificial Life V*: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, edited by C. G. Langton and K. Shimohara, p.462 (MIT Press, Cambridge MA, 1997).

[15] M. C. Cross and P. C. Hohenberg, Rev. Mod. Phys. **65**, 851 (1993).

[16] C. Darwin, *On the Origin of Species* (D. Appleton & Company, New York, 1892).

[17] T. Dobzhansky and S. Wright, Genetics **28** 304 (1943).

[18] R. A. Fisher, Ann. Eugen. **7**, 355 (1937).

[19] R. García-Pelayo, Phys. Rev. E **49**, 4903 (1994).

[20] L. Gil and D. Sornette, Phys. Rev. Lett. **76**, 3991 (1996).

[21] S. Grand, D. Cliff, A. Malhotra, in the proceedings of the Autonomous Agents 97 conference (1997).

[22] G. Gutenberg and C. F. Richter, Ann. Geophys. (C.N.R.S.) **9**, 1 (1956).

[23] T. E. Harris, *The Theory of Branching Processes* (Springer, Berlin; Prentice-Hall, Englewood Cliffs NJ, 1963).

[24] J. H. Holland, *Adaptation in Natural and Artificial Systems* (2nd ed.) (MIT Press, Cambridge MA, 1992).

[25] R. Laing, in *Advanced Automation for Space Missions*: Nasa Conference Publication 2255, edited by R. Freitas and W. P. Gilbreath, p.189 (1982).

[26] C. G. Langton, ed., *Artificial Life: An Overview* (MIT Press, Cambridge MA, 1995).

[27] K. B. Lauritsen, S. Zapperi, and H. E. Stanley, Phys. Rev. E **54**, 2483 (1996).

[28] J. S. McCaskill and G. J. Bauer, Proc. Natl. Acad. Sci. USA **90** 4191 (1993).

[29] F. W. Preston, Ecology **29**, 255 (1948).

[30] F. W. Preston, Ecology **43**, 185, 410 (1962).

[31] D. M. Raup, Paleobiology **11**, 42 (1985).

[32] T. S. Ray, in *Artificial Life II:* Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems, Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol. 10, edited by C. G. Langton et al., p.371 (Addison-Wesley, Reading, MA, 1992).

[33] T. S. Ray, Physica D **75**, 239 (1994); Artificial Life **1**, 195 (1994); *Artificial Life IV*: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems, edited by R. A. Brook and P. Maes, p.377. (MIT Press, Cambridge MA, 1994).

[34] J. J. Sepkoski, *A Compendium of Fossil Marine Animal Families* (2nd ed.) (Milwaukee Public Museum, Milwaukee, WI, 1992) with emendations by J. J. Sepkoski based largely on M. J. Benton, ed., *The Fossil Record 2* (Chapman & Hall, New York, 1993).

[35] K. Sims, Artificial Life **1**, 353 (1994).

[36] K. Sneppen, P. Bak, H. Flyvbjerg, and M. H. Jensen, Proc. Nat. Acad. Sci. U.S. **92**, 5209 (1995).

[37] D. Sornette, A. Johansen, and I. Dornic, J. Phys. I **5**, 325 (1995).

[38] F. Spitzer, *Principles of Random Walk* (Springer-Verlag, New York, 1964).

[39] M. Stemmler, M. Usher, and Z. Olami, Phys. Rev. Lett. **74**, 326 (1995).

[40] G. Sugihara, Am. Nat. **116**, 770 (1980).

[41] G. Theraulaz and E. W. Bonabeau, Science **269**, 687 (1995).

[42] A. Vespignani and S. Zapperi, Phys. Rev. E **57**, 6345 (1998).

[43] J. von Neumann, in *Cerebral Mechanisms in Behavior—The Hixon Symposium* (John Wiley, New York, 1936).

[44] H. W. Watson and F. Galton, J. Anthropol. Inst. Great Britain and Ireland **4**, 138 (1874).

[45] B. Webb, Sci. Am. **275**, 94 (1996).

[46] J. Yin and J. S. McCaskill, Biophys. J. **61** 1540 (1992).

[47] G. U. Yule, Proc. Roy. Soc. London Ser. B **213**, 21 (1924).

[48] S. Zapperi, K. B. Lauritsen, and H. E. Stanley, Phys. Rev. Lett. **75**, 4071 (1995).