COSMIC: A Model of Cellular Genetic Interaction and Evolution

Thesis submitted in accordance with the
requirements of the University of Liverpool
for the degree of Doctor of Philosophy by

Richard Gregory

January 2004

Abstract

COSMIC: A Model of Cellular Genetic Interaction and Evolution

by

Richard Gregory

Evolution has frequently been seen as a result of the continuous or discontinuous accumulation of small mutations. Over the many years, it has been found that simple point mutations are not the only mechanism driving genomic change, for example, plasmids, transposons, bacteriophages, insertion sequences, deletion and duplication, and stress-sensitive mutation all have a part to play in directing the genetic composition and variation of organisms towards meeting the moving target that is the environmental ideal at any one time. Considering the probability of single point mutations arising and repair mechanisms that act to counteract their accumulation, it is unlikely that simple mutation can create rapid diversity. Evolutionary change depends more on larger scale changes in genomic sequences caused by sexual and other forms of horizontal gene transfer. These generate the variation necessary to allow rapid evolutionary response to changing environmental conditions.

Predictive models of *E.coli* cellular processes already exist, these tools are excellent models of behaviour. However, they suffer the same drawbacks; all rely on actual experimental data to be input and more importantly, once input that data are static. The aim of this study is to answer some of the questions regarding bacterial evolution and the role played by genetic events using an evolving multicellular and multispecies model that builds up from the scale of the genome.

To test these questions, it is necessary to build a model that attempts to en-

compass what are considered the important qualities of bacterial evolution and bacterial life, but not be overly specified as to constrain the results. The model is therefore a careful balance of biological and computational realities with an emphasis on open-endedness and individuality. The biological literature has many examples of the possible forms of mechanism within the relatively 'simple' example of *E.coli*, but even this must be carefully constrained. It is clear that computer models lack complexity when compared to real world processes.

In focusing attention on aspects of the *E.coli* system, new insights are emerging from the disciplines of genomics and proteomics. The genome should perhaps be regarded not as a book that is continually read from, but rather a program that is continuously executed and adapted over the life time of individual cells. From this it appears that interactions within cells involve the combined effects of enzymes, structural and regulatory proteins acting on genes, which in turn act on those enzymes and other proteins, creating a huge number of both positive and negative feedback loops necessary for controlled execution. The ideal model therefore is one that takes both these stages into account, each genome being an implementation of what many conceive as the computational cell. There are then three themes to the model: the environment, the genome and functional proteins, all of which use an individual based philosophy. The environment contains individual cells, each cell contains an individual genome and each gene can lead to individual gene products each with their own spatial and temporal parameters. This vast number of parameters and possibilities adds another meaning to the name of the simulation, COSMIC : COmputing Systems of Microbial InteraCtions.

This thesis describes the novel COSMIC model, the genetics background, the parallel implementation and results showing the initial stages of evolution towards the goal of learning to follow a food source gradient.

Acknowledgements

My thanks go to Dr. Ray Paton for his invaluable help and advice and humour. Without which this research would surely have been very different. My thanks also go to Prof. Q. Henry Wu for his support and guidance, and his understanding when this work moved away from what was originally planned. I also thank them both for the financial support when this project dragged on past the three years for which the EPSRC had graciously funded.

My thanks go to Dr Jan-Ulrich Kreft for providing a detailed criticism of this work which helped to focus COSMIC into the shape it is now.

My thanks also go to my family and friends who have had to cope with me talking about COSMIC related matters more than was really necessary. Special mention should be made of Peter Owens, who endured lengthy COSMIC monologues for the duration of the project.

Finally, my thanks go to all the thousands of people who have written and maintained the UNIX software which I depended on. I cannot imagine how this work would have turned out had I not had access to such a wide range of software tools.

Sadly, Dr. Ray Paton died suddenly shortly before binding this final version of the thesis. He will always be remembered for his kindness, support and understanding, it was these and many other qualities which made this thesis possible. He leaves behind many areas of interesting work that had much further to explore. COSMIC is one of these and so it is hoped COSMIC will continue to provide a better viewpoint when modelling complex systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Evolution has frequently been seen as a result of the continuous or discontinuous accumulation of small mutations. Over the many years, it has been found that simple point mutations are not the only mechanism driving genomic change, for example, plasmids, transposons, bacteriophages, insertion sequences, deletion and duplication, and stress-sensitive mutation all have a part to play [Sha97,Sha99] in directing the genetic composition and variation of organisms [Koc93] towards meeting the moving target that is the environmental ideal at any one time. Considering the probability of single point mutations arising and repair mechanisms that act to counteract their accumulation, it is unlikely that simple mutation can create rapid diversity. It is clear that evolutionary change depends more on larger scale changes in genomic sequences caused by sexual and other forms of horizontal gene transfer. These generate the variation necessary to allow rapid evolutionary response to changing environmental conditions.

Predictive models of *E.coli* cellular processes already exist, the E-Cell project [Tomita *et al.*, 1999] aims to use gene data directly in a mathematical model of transcription. The Virtual Cell [SFSCL97, SL99] project makes use of user-defined protein reactions to simulate compartments at the nucleus and cellular

level. Gepasi3 [Men97] also models protein reactions, but from within an enclosed box environment. The BacSim [KBW98] project simulates individual cell growth at the population scale. Eos [BSS00] is also based at the population scale, but is intended as a framework for testing idealised ecologies, represented by evolutionary algorithms. These tools and those that they rely on are excellent models of behaviour. However, they suffer the same drawbacks; all rely on actual experimental data to be input and more importantly, once input that data is static. The aim of this study is to answer some of the questions regarding bacterial evolution and the role played by genetic events other than simple point mutation using an evolving multicellular and multispecies model that builds up from the scale of the genome. In effect, it is not bacterial evolution that is being interrogated, but the co-evolution of bacteria and any organism that has a direct effect on the genetics of those bacteria.

To test these questions, it is necessary to build a model that attempts to encompass what are considered the important qualities of bacterial evolution and bacterial life, but is not overly specified as to constrain the results. The model is therefore a careful balance of biological and computational realities [Way01] with an emphasis on open-endedness [Kam96]. The biological literature has many examples of the possible forms of mechanism within the relatively 'simple' example of *E.coli*, but even this must be carefully constrained. It is clear that computational models lack computational power when compared to real world processes.

In focusing attention on aspects of the *E.coli* system, it is clear that there are two new insights provided by the emerging disciplines of genomics and proteomics. Proteomics is the study of enzyme and protein interactions. Traditionally this meant differential equation models of interaction. However, nowadays there seems also to be an implicit link with the application of protein descriptors derived from sequence information in identified genes [Karplus *et al.*, 1997], an application that has only recently become tractable with the arrival of accurate genome data. Genomics is the study of genome structure, interaction and encoding and has been stimulated by the Human Genome

project [KH01] as well as whole genome sequencing projects for many other organisms, notably those for numerous bacteria. From this it appears that interactions within cells involve the combined effects of enzymes, structural and regulatory proteins acting on genes, which in turn act on those enzymes and other proteins, creating a huge number of both positive and negative feedback loops necessary for controlled execution [Fre00]. The genome should perhaps be regarded not as a book that is continually read from, but rather a program that is continuously executed and adapted over the life time of individual cells, tissues or entire organisms. The ideal model therefore is one that takes both these stages into account and allows for the evolution of the genome in the presence of other genomes, each genome being an implementation of what many conceive as the computational cell [Bra90, Sha91, Bra95, Pat98, DHB00, RLM96, AR94].

There is a clear distinction between the clean world of abstraction and the biology on which it is based. In any system there is always a tendency for homogeneous structures, well defined parameters and reasonable assumptions; biology is no different as these goals promote clear description, what is different is the world that biology tries to explain. Biology is full of explanations of mechanism, but like contemporary standards, there are so many to choose from. This is not to say they are wrong, but in different circumstances assumptions made by each in either measurement, environment or initial point of enquiry can come to bear making one explanation less powerful than another explanation. The reasons for this come from the problems of basic measurement, sheer biodiversity and the change in scale making analogies essential - scale referring both to time and space. This last point is actually quite important, mechanisms acting outside of normal experience makes intuition irrelevant.

It is clear that life forms are far to complex for current understanding to even scratch the surface, the concepts needed to grasp the complexities involved are some way off. As a result, bacteria promised to be the most reasonable starting point for asking fundamental questions about the development of evolution and life itself. Fortunately recent years has seen an explosion of bacterial research brought about by new technology, this has lead to new avenues of re-

search previously impossible through lack of data or computational power.

With this in mind, the simulation of bacterial adaptation has become this authors goal, this thesis details a novel computational model linking bacterial genetics, environmental response and survival techniques. In the future this can in turn lead to more involved simulations or alternatively lead to simplified simulations which aim to find the basic requirements for bacterial adaptation.

1.2 Thesis Outline

This section outlines the contents of each chapter, it is recommended the chapters be read in numerical order.

Chapter 2 introduces the biology on which the COSMIC model is based. The general structure of *E.coli* is discussed to give some appreciation of the size of the organism and at the same time its complexity, this will then lead onto modelling *E.coli* growth and the various parameters that are important. Section 2.5 will then change scale and look at the genetics of *E.coli*. From the static structure it will then move onto the mechanics that operate on the static structure and allow the cell to produce chemical machinery. Section 2.8 introduces the adaptive nature of the genome and the cell as a whole. Sections 2.9 and 2.10 give two well known specific examples of optional transcription which represent typical examples of the genetic control that can be achieved. Sections 2.16 and 2.17 detail some of the causes of genetic mutation and long term adaptation. Finally, section 2.18 pulls all these aspects together and puts forward the case for a bacterial simulation that encompasses many of these these topics.

Chapter 3 moves away from the biological material and instead focuses on simulation and analysis of biological systems, specifically with a genetics basis. Before COSMIC there have been many simulations of genetics, both for the sake of biology itself and biologically inspired algorithms such as Genetic Algorithms. This chapter mentions a few of those models and importantly their limitations.

Chapter 4 makes use of the information from chapter 2 to build a computational model of bacterial growth and evolution. This is entirely based on the idea of modelling the individual, be it individual cell or individual molecule, and so will be explained in terms of sets and relations between sets and members of sets. This chapter starts with section 4.2 and section 4.3 describing the main biological phenomena that COSMIC models. Section 4.4 then discusses more detail of how such a model could be implemented in such a way that computation is feasible. Section 4.5 then starts with the model proper by detailing the construction of a genome, from the genes and their encoding, the types of genes, the construction of operons and finally the genome of an individual cell. This section then goes on to specify the other constituent parts of a single cell, building to a population of these cells in a specified environment. The purpose of this section is to describe that static structure of the model, the later sections then build on to include the dynamics.

Section 4.6 describes the dynamics within the context of chapter 2, this highlights the important points of transcription and gives an overview of the most important dynamic in COSMIC, namely the interaction diagram of figure 4.4. Section 4.7 then discusses how this dynamic aspect is incorporated into the previous formal static representation. Section 4.8 and 4.9 describes the mathematical functions that implement the state transition dynamics which are applied to the structures of sections 4.5 to 4.7. Section 4.10 describes the specifics of the interactions in the context of the representation and the mathematical functions.

Section 4.11 moves to a different scale, that of the cell population, by discussing the details of the environment in which these cells live. Having now described all the structures and possible interaction pathways, section 4.13 describes the initialisation of the system as a whole, how the original genomes come about and how enzymes can exist when there are no enzymes to create them. Finally, so that evolution may occur, section 4.14 describes the mutation operator that is applied to the previously described structures.

Chapter 5 describes the parallel implementation of COSMIC and shows

that it is possible to map a dynamic problem such as this onto fixed resources. The only way to achieve the necessary level of performance is with parallel computers and a suitable designed implementation that maps the problem onto the hardware, as shown in section 5.3. For real problems this mapping can be non-trivial requiring careful consideration of the constraints in both the system being modelled and the hardware that executes the model. For the most part efficiency (discussed in section 5.12) is achieved by making use of implicit multiplexing of resources and shows the importance of knowing where to partition the problem between server and clients. Through this an efficient simulation has been created, making maximal use of the available hardware without constraining the model to require excessively specific resources.

Chapter 6 introduces the most common visualisations used to represent the raw data generated by COSMIC. The two scales of COSMIC give the initial division of the visualisations, the top level environment where the cells play out their struggle for survival and ultimately demonstrate their evolution is covered in sections 6.2 and 6.3. At the other scale there is the internals of each cell, which contains the richest data but also the hardest to view in any one way that captures all the changes. This is covered in sections 6.4 to 6.6. Some of these visualisations are research topics in themselves as the data generated by COSMIC is so rich that a single image only scratches the surface of what interactions actually occurred.

With the previous chapter having introduced some of the visualisation techniques used by COSMIC, chapter 7 then gives an account of some of the simulations. Not just the data obtained but also the evolution of the simulation and its testing. Section 7.2 describes the overall control parameters that enable aspects of COSMIC functionality. These provide overall control of the system by specifying limits to the environment, cell growth, cell division, cell genome size, genome mutation rates, enzyme half lives and genome-proteome interaction rates. Section 7.3 introduces the data sets that make up the archived COSMIC output. The testing phase of the simulation is then described in sections 7.4 to 7.14. When simulation runs where made, problems were found

and corrected and the bulk of the chapter is made up of those experiences. This provides some idea of the subtle effects of programming errors and more often, simple unforeseen consequences of some implementation decision. This chapter finishes with a summary of the main outcomes in section 7.15.

Chapter 8 brings this thesis to a conclusion by summarising where COSMIC started from in section 8.1 and in section 8.2 stating what it now is capable of. Section 8.3 outlines the main outcomes of the work, with section 8.4 describing what are considered the main challenges that were overcome by COSMIC. Section 8.5 discusses a specific cause of complexity in COSMIC, with the view that the complexity is necessary. Finally, section 8.6 outlines future directions for COSMIC and work derived from it.

Note: This thesis is available as a full colour postscript file, available at: http://www.csc.liv.ac.uk/~ greg/thesis.ps

Chapter 2

A review of cell biology relevant to the COSMIC system

2.1 Introduction

There is a vast amount of information available on both eukaryote and prokaryotic cells; and especially *E.coli*, the bacterium on which this study is based. The following text is largely concerned with *E.coli* with some brief details of eukaryote cells - largely to show there are similarities and differences in modelling, depending on what is viewed as important.

The general structure of *E.coli* will be discussed to give some appreciation of the size of the organism and at the same time its complexity, this will then lead onto modelling *E.coli* growth and the various parameters that are important. Section 2.5 will then change scale and look at the genetics of *E.coli*. From the static structure it will then move onto the mechanics that operate on the static structure and allow the cell to produce chemical machinery. Section 2.8 introduces the adaptive nature of the genome and the cell as a whole. Sections 2.9 and 2.10 give two well known specific examples of optional transcription. Sections 2.16 and 2.17 detail some of the causes of genetic mutation and long term adaptation. Finally, section 2.18 pulls all these aspects together and puts forward the case for a bacterial simulation that encompasses many of

these topics.

This chapter largely originated from the collection of papers found in Neidhardt *et al.* [Nei96V1, Nei96V2], a well written general dictionary of biology [TH96] and a concise overview of molecular biology [TMBW97], with more up to date information from a variety of sources.

2.2 Classification

Eubacteria represent a subdivision of prokaryotes, the other group being archaea which are similar in structure to eubacteria but have ribosomal RNA molecules that have evolved differently. Prokaryotes are surrounded by a phospholipid cell membrane through which small molecules can pass with the aid of proteins. Usually, a single circular chromosome is contained inside the cytoplasm (or cytosol) and is attached to the cell membrane at a single point. On the outside of the cell there can be both pili (hair like sticks for sticking) and flagella. *E. coli* has a genome size of 4600 kb, which amounts to around 3000 reproducible proteins [TMBW97, pp.2]. In contrast the genome of the simplest bacterium called *Mycoplasma genitalium* is 580kb long and encodes 470 proteins.

2.3 *E.coli* Structure

[NU96] lists a variety of figures for the biochemical composition (per individual cell) of an average *E.coli* cell. To arrive at these figures an average is taken as a population of *E.coli* (strain B/r) in balanced growth at 37^{o}C in aerobic glucose minimal medium with a mass doubling time of 40 minutes. The quantities for a cell are then defined by dividing the total biomass, or the amount of any of its measured components, by the total number of cells in the population. This average cell is therefore approximately 44% through its division cycle and assuming that increase in cell mass is exponential, is approximately 33% larger than when that cell was created. It should be noted

that this brings about a level of uncertainty in all figures, not least because the strain and exact conditions are not always specified in the literature. Measurements can be easier (or indeed possible) only for some strains and conditions, so often these variables are in fact implicitly given by the type of experiment. The main result is then many missing parameters for a given scenario and strain, and so it natural to infer parameters based on related parameters and assume they are close enough until there is evidence to the contrary.

Of the tables given in [NU96] the most important figure was the protein molecules per cell of 2,350,000 with 1850 different kinds of those molecules[1], this amounts to 55% of the total dry weight, and 156 Amt (10^{-15} grams) per cell. DNA is quoted to have 2.1 molecules per cell (as it is being continuously duplicated in the above conditions) and account for 3.1% of the total cell dry weight, and 8.8 Amt (g. 10^{-15}) per cell. Metabolites, cofactors and ions (all in the same grouping) account for 3% of total dry weight of the cell and 9.9 Amt per cell.

[Mac96] gives a detailed account of *E.coli* flagella physiology, including genes responsible for its environment. Of relevance to COSMIC are some figures and some general states of motility. The number of flagella per cell are in the range 0 to 15, typically around 8 but different conditions bring on different numbers from the same initial strain, the cost of flagella synthesis and operation come to bear in selection. Flagella lengths are in the range 0 to 20 μm and are more typically around 5-10 μm. Each flagella base is positioned randomly on the cell wall (peritrichous flagellation). Rotation speed of a flagella bundle is around 100 Hz for a free swimming cell and saturated (125 milliVolt) motor. Efficiency is presumed to be high at moderate to high load though no data is available. Under high load, torque per motor is around $3x10^{-18}$ N m, power output per motor is around 10^{-16} W at 20 Hz, power per cell is around 10^{-15} W under normal swimming conditions. Under conditions of cell growth, total flagella operation amounts to 0.1% of total energy expenditure. Flagella synthesis accounts for 2% of biosynthetic energy use. Note these figures are

[1]Note this is lower than the 3000 figure given by [TMBW97, pp.2], these totals were presumably calculated using different methods.

relative and total energy expenditure per cell is not available.

Broadly, *E.coli* has three observed states of movement, swimming, tumbling and pausing. Swimming is accomplished by bundling all flagella into a single axis. The flagella motor can rotate in both directions yet the flagella is normally a left handed helix, a counter clockwise motor rotation rotates the flagella creating a pushing against the cell. The combination of hydrodynamic and mechanical forces force all the flagella to congregate into a single tail whose axis is normally the same as the longest axis of the cell. The entire bundle is able to rotate at 100 Hz, this amounts to 25 μm/s in a liquid medium at room temperature. Bear in mind that MacNab [Mac96] hints these are rather specialised circumstances and that the speed is normally lower.

Tumbling occurs when the flagella motor changes direction. The shape of each flagellum is polymorphic in that it can a left handed helix (normal) or right handed helix (curly); the term curly comes from the observation that the right handed helix has half the wavelength of the left handed. A reversal of motor direction to clockwise rotation reorients the helix from the base outward, forming the right handed helix. While this is happening on mass, flagella that are partially converted will have kinks and tend to role over themselves ensuring they are reoriented ready for another round of swimming. Pausing is an observed behaviour that lacks a proven explanation. Observations show that pausing is inversely proportional to frequency of reversal, suggesting that pauses are in fact failed reversals. This is further supported by there being no evidence of a physiological need for pausing.

In well energised cells motor switching and changes of motor direction are always occurring regardless of any environmental gradient, each motor is independent. It is only when they are considered as a whole that the flagella interaction becomes weighted, spending around 1 second swimming (i.e. uniform motor rotation) and 0.1 seconds tumbling.

The cause of rotation in either direction is proton potential around the base. The microphysiological details of the process remain unclear, it has been modelled in several forms and the actual motor must agree with all of

them but the details are unknown. The proton potential can come from either chemical potential (pH difference) or electrical potential (mV difference). It must be noted that this level of chemistry is far from the simulation level and so has no hope of being incorporated into the simulation without massive simplification; in the presence of free oxygen, the electron transport chain (i.e. the general cell wide mechanism of electron transfer) under free swimming conditions generates a potential larger than the motor saturation voltage, so the motor speed does not really vary with changes in oxygen level. Without oxygen (anaerobic), glycolysis takes place; this is a more specialised pathway that starts with starch or glycogen and ends with production of two ATP molecules per glucose molecule and production of either pyruvate (for the tricarboxylic acid cycle) or lactate. The proton potential produced by this pathway is not enough to saturate the motor and so it does not move as quickly.

For the purposes of simulation, the flagella clearly needs gross simplification as it is intended to only provide a secondary effect of supporting evolution. As a result the simulated flagella swim and tumble at the same time, also the bacteria is assumed to be relying on glycoses and so encourages more careful use of the flagella.

In [BD96], there is a recommendation that cell mass is used as the basic parameter with which to compare cell stages. They are really all as bad as each other because there is no one uniformly increasing parameter independent of the others, but cell mass is easier to measure, being simpler, faster and more accurate.

2.4 Modelling Population Growth

[KW82] specify some models of the relationship between glucose uptake and growth rate. This formed the partial source of the BacSim growth rate model, and is what the COSMIC environment was based on. Quantitative comparisons are made between batch grown and continuous cultures, also and importantly, the comparisons are also between explanatory equations that re-

late glucose concentration to cell growth velocity. Given here is the equation used by COSMIC to link glucose concentration to velocity(growth rate). The simplicity (and continuous nature) of the Monod based equation made it the better choice for the present study.

The form given is of: $v = V_{max}s/(K_m + s)$ where v is the velocity relative to maximum growth rate (i.e. $0 \leq v \leq 1$)). V_{max} is a slope parameter and has the value 1.23 h^{-1} for batch grown *E.coli* and 0.536 h^{-1} for continuous cultured *E.coli*. K_m is the half-saturating constant, a constant that refers to the half-way level of glucose; the initial batch grown value was $13\mu M$ and the continuous grown value was $0.597\mu M$. s refers to the glucose concentration as experienced by each bacterium, it was in the range $0 \leq s \leq 25\mu M$.

In COSMIC, these figures have been changed to be per minute rather than per hour, and per fg(femtogram) rather than per μM. The original units are traditional in the microbiological domain but are unusual and so have converted to SI units. V_{max} is then 0.0205 m^{-1}, K_m is 2.34 mg and s is in the range $0 \leq s \leq 0.0045$ fg.

V_{max} was also adjusted so that the equation output corresponded with growth rate. This turned out to use the 0.4444 figure from BacSim, though BacSim's growth equation is not used.

Also, V_{max} is multiplied by the cell mass after being adjusted for the expected average cell mass. This amounts to a cell mass of 0.4 femtolitre (the deterministic maximum) giving a maximum growth rate that grows a cell from 0.2 fl to 0.4 fl in 24 minutes.

The rest of the [KW82] is concerned with membrane uptake rates on continuous cultures and so is not really relevant. It does give the impression that batch grown is avoided nowadays by bacterial researchers because it necessarily means that conditions change over the course of experiment, this lead to the assumption that parameters were constant when they are not.

Originally a reference from BacSim on the Y_{max} parameter (fg dry mass yield per fg glucose) and m, the apparent maintenance rate. [NTT96] cites Monad that he found between 0.21 and 0.28 g dry mass per g substrate (none

	C(CP-1000)		Unnamed	
Dilution rate	q_{glucose}	Y_{glucose}	q_{glucose}	Y_{glucose}
0.1	1.56	64.1	1.25	80.0
0.3	4.08	73.5	3.56	84.3
0.5	6.60	75.8	5.92	84.5
0.7	9.12	76.8	8.27	84.6
Y_{max}		79.4		85.4
m	0.3		0.07 *	

Table 2.1: Growth rates, per millimoles of glucose per hour per gram

were glucose), and then cites Schulze and Lipe who in line with contemporary parameters says the maximum conversion efficiencies for continuous glucose is 0.51 grams dry mass per gram glucose.

An important point to note is that in a continuous culture (as most are, this is then a steady state situation) growth rate is proportional to dilution rate. As a result, most tables show figures proportional to dilution rate, normally as a fraction removed per hour.

However, as table 2.1 shows, the relationship is not constant or even linear. This is for the *E.coli* strain C(PC-1000) and an unnamed strain, of the four strains given by Neijssel *et al.* [NTT96] these represent the lowest and highest growth rates, note their is not a great deal of change between them. Original values are in Table 2.1.

g_{glucose} is in millimoles (note in error [NTT96] in [NU96] uses units of nanomoles) of glucose per hour per gram (dry weight) of cells, Y_{glucose} is in grams (dry weight) of cells formed per mole of glucose consumed. Y_{max} calculated maximum growth yield, corrected for maintenance and is in grams (dry weight) of cells formed per mole of glucose and maintenance rate of glucose consumption. m is extrapolated for 0 growth and is in nanomoles of glucose per hour per gram (dry weight) of cells.

For use in COSMIC, these units are then converted from millimoles to grams and hours to minutes. The result is in table 2.2.

q_{glucose} is then grams glucose per minute per dry cell weight gram and

	C(CP-1000)		Unnamed	
Dilution rate	q_{glucose}	Y_{glucose}	q_{glucose}	Y_{glucose}
0.1	4.68m	0.356	3.75m	0.444
0.3	12.24m	0.408	10.68m	0.468
0.5	19.8m	0.421	17.76m	0.469
0.7	27.36m	0.427	24.81m	0.47
Y_{max}		0.441		0.474
m	0.9m		0.21m *	

Table 2.2: Growth rates, per gram of glucose per minute per gram

	q_{glucose}			
	C(CP-1000)		Unnamed	
Dilution rate	0.3 fl	0.4 fl	0.3 fl	0.4 fl
0.1	6.79m	9.05m	5.44m	7.25m
0.3	17.75m	23.66m	15.49m	20.65m
0.5	28.71m	38.28m	25.75m	34.34m
0.7	39.67m	52.9m	35.97m	47.97m
m	1.305m	1.74m	0.305m *	0.406m *

Table 2.3: Growth rates, average cell glucose use per second per gram

Y_{glucose} is then grams dry weight per gram glucose consumed.

For approximate comparison with COSMIC we then assume an average cell volume of 0.3 fl cell and convert femtograms to units of an average volume 0.3 fl cell (using the conversion coefficient of 290 fg fl^{-1} which converts femtograms to femtolitres [KBW98]) and also converting minutes to seconds throughout we arrive at table 2.3.

Taking an example cell of volume $c = 0.417388$ fl, mass of 121.042434 fg and fresh substrate of 0.0045 fg glucose fl^{-1}, calculated growth is $v_{rate} = 0.018847$ fg/fl. Looking at the volume increase rate v_{rate}, doubling from 0.2 to 0.4 fl per cell is achieved at $((c+u)/c)^y = 2$, or 33 mins. This should be around 24 mins or $((c+u)/c) = 1.0004815$ ($u = 0.041889$) if there was maximum growth, which is a dilution rate of around 0.82.

Another note says *E.coli* produce 0.21 to 0.28 grams of matter per gram of glucose, so a maximum growth of 24 minute doubling, or $((c+v)/c) = 1.0004815$ with $c = 116$ fg/fl of dry cell mass, v is then 0.04188 fg/fl of dry mass increase per second. This amounts to (taking 0.25 as the mean of 0.21 and 0.28) as 0.1276 fg/fl of glucose use per second.

BacSim's choice of m is slightly too high, but allowing for the above error it is valid. Choosing a representative m from above we see it is slightly larger than 0.0006 fg glucose / fg dry mass per min. From the table, $m = 0.9$ mg glucose / g dry mass per sec seems more appropriate, but, using glucose to mass conversion efficiency of 0.25 (Neijssel et al 96 says 0.21 to 0.28), the fastest doubling time of 24 mins, max growth per min is then 1.0293. An average cell volume of 0.3 fl gives an initial absolute increase of $u = 2.549$ fg/min ($c = 0.3fl * 290$ fl/fg, $(c+u)/c = 1.0293$). Using the 0.25 efficiency of conversion, a cell mass increase of 2.549 fg/min equates to 10.2 fg glucose/min at the middle of the cycle, 6.8 fg glucose/min at the start and 13.6 fg glucose/min at the end. Or 245 fg per cell per duplication cycle. COSMIC uses the corrected figure from [NTT96], although originally based on BacSim, some of the formulae and parameters of BacSim were abandoned in favour of using a function that directly links glucose concentration with growth velocity, this is then modified to be also based on the cell mass. Glucose usage is calculated by working backwards from the mass increase using the 0.245 conversion efficiency approximation.

A strain of *E.coli* is continuously grown in a chemostat at a dilution rate of 30% per hour, hence the growth rate is low at around 0.25 of maximum, or doubling every 96 minutes. This is based on a maximum dilution rate of 0.82 and maximum doubling time of 24 minutes. According to BacSim and other sources, a cell grows from 0.2 to 0.4 femtolitres and then divides, or in dry weight terms, goes from 58 to 116 femto grams dry weight. Obviously a cell doesn't have to be inside these limits but they are typical and are enough for this aspect of COSMIC.

q_{glucose} is the glucose consumption rate at 4.08 millimoles of glucose per hour per gram (dry weight) of cells. Converting the units this amounts to 734

milligrams glucose per hour per gram (dry weight) of cells, or 12.2 milligrams glucose per minute per gram of cells.

Y_{glucose} is the total growth yield at 73.5 grams (dry weight) of cells formed per mole of glucose consumed. Converting the units this amounts to 0.41 grams (dry weight) of cells per gram of glucose.

So over the growth time of a cell (96 minutes), the cell gains $116 - 58 = 58$ fgrams, so it should use 58 $\text{fg}_{\text{dw}}/0.41 = 141$ fgrams of glucose - using the Y_{glucose} figure.

Using the q_{glucose} figure requires some calculation. If a cell goes from 58 to 116 fgrams in 96 minutes, then it must gain by: $58 * x^{96} = 116$, $x = 1.007246$ per minute, so it must use (integrating): $58 \text{ fg} * ((x^{97} - 1)/\ln x) * 12.2mg = 96.9\text{x}10^{-15}$ grams of glucose. Which is smaller a value than for Y_{glucose}.

BacSim [KBW98] was the original source of figures and environment for COSMIC. BacSim, based on Gecko [Boo97], which is based on Swarm [Swa00], tries to answer the question of how to best model a macroscopic world based on data on microscopic entities. Other models are DE or cellular automata based. Individually based modelling is different in that it returns more data, needs less accurate parameters but also requires more computational power. The idea of this model is to generate predictive results rather than fitting results based on several growth models. It also mentions that two models by others used 22 and 200 parameters respectively, clearly outside the scope of COSMIC.

The BacSim model uses discrete time at 0.1 mins per iteration. Each iteration goes through a cycle of diffusion and uptake (modelled by equations), metabolism (amounting to an increase in size based on update, derived from an equation including update efficiency), death if below a minimum size, division if size is a multiple of a requisite size (from [DR96]). The cells are then moved to remove overlap and then the cycle continues. Cell division creates two identical cells. Three variations of the D & R model were tested by BacSim. Variation of parameters was via a Gaussian distribution in the range $\pm 2\sigma$, with the positive of the result being used.

BacSim cell motion was achieved by taking a vector of required movement.

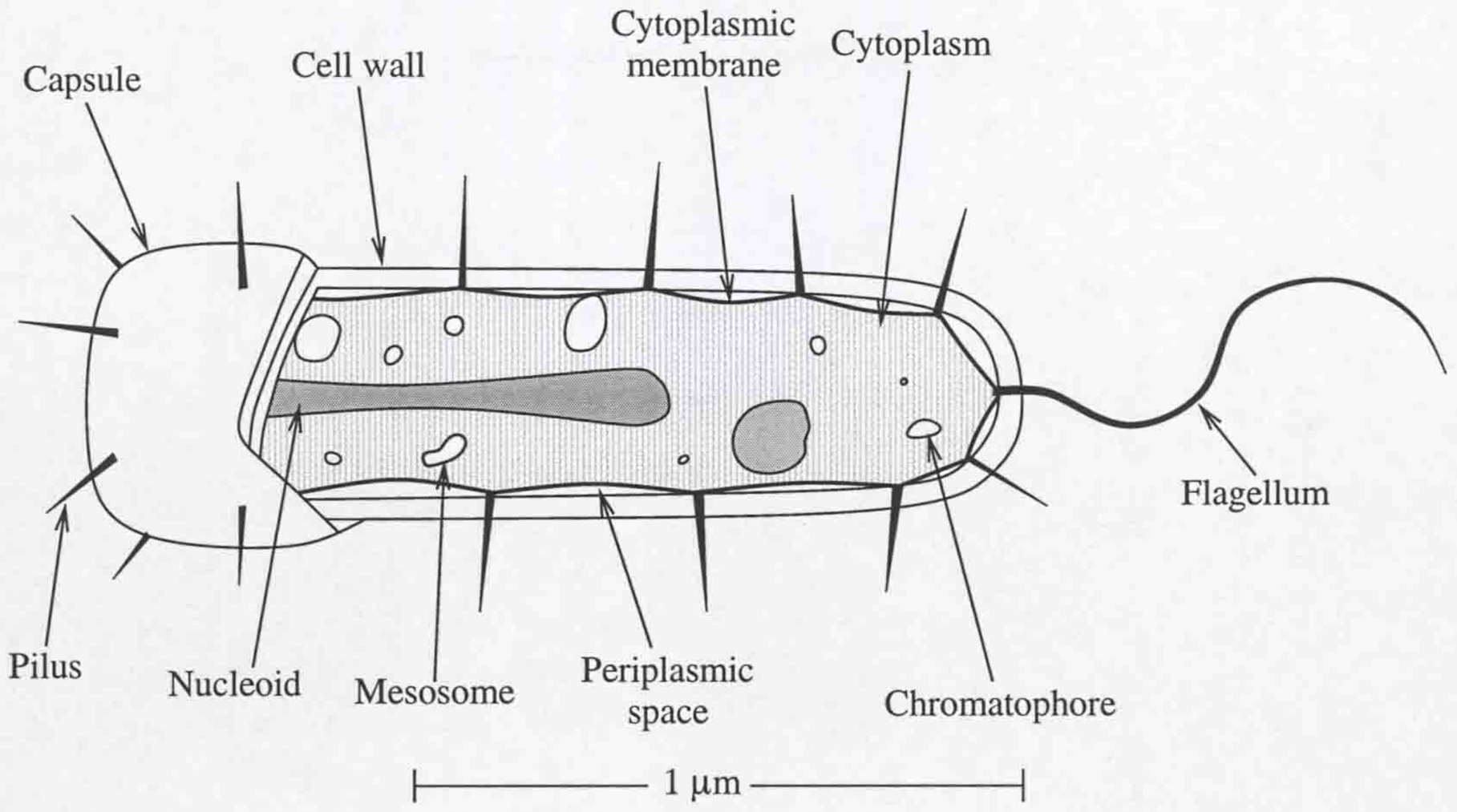


Figure 2.1: Single prokaryote cell

Total area of biomass [Pir67] is then 77%, from 0.1 g dry mass ml^{-1} of *E. coli* - assuming a water content of 70% [NU96].

All parameters in BacSim were based on published data, only the diffusion approximation (a grid of coefficients) needed to be tuned on a single parameter. This made it an ideal basis for the development of the COSMIC model.

The median cell volume is given as 0.3 fl, which equates to 0.3 am^3. Given a typical cell picture (from [NU96]), the length tip to tip is 20 units, length cylinder to cylinder is 15 units and the diameter is 5 units, giving a cylinder height to cylinder radius ratio of 6. Sphere volume is $V = \frac{4}{3}\Pi r^3$, Cylinder volume is $V = \Pi h r^2$, so $r = 0.235\mu m$ given $0.3am^3 = \frac{22}{3}\Pi r^3$, h is then $1.41\mu m$. This agrees with $2\mu m$ figure that is commonly given, more so if the mean is taken as 0.4 fl.

Comparing the eukaryote cell and the prokaryote cell (Figure 2.1), prokaryote cell is visibly the simplest structure; being unicellular or filamentous and up to 3 μm in length. It has no nucleus to contain DNA, the DNA is instead in a compact area of the cell inside the cytoplasm (i.e. not covered by a separating membrane). Propulsion for many is achieved using a tail (flagellum), which is

rotated at high speed through proton motive force across the cell wall. Unlike prokaryotes, eukaryotes have their DNA stored in at least one nucleus that is separate from the cytoplasm.

2.5 DNA, RNA and Proteins

DNA is the carrier of genetic information for prokaryotic and eukaryotic organisms. In its simplest representation DNA is a chain of four nucleic acids that take the names C, A, T and G. Biological reality is more complicated, consisting of phosphodiester bonds (sugar-phosphate) that form a regularly spaced chain to which these nucleic acids can bind to. For each DNA strand there is a complement strand that is attached at the nucleic acid bases using a hydrogen bond. C complements G and A complements T (the relationship is reciprocal). As only the bases stick but the bases on the same strand are chained together; the complementary bases point to each other and the phosphodiester bonds run along the outer edges of the strand pair. The pair of complementary strands wrap around each other in a coil, and this coil is itself regularly coiled. A good example of the scale involved comes from the (eukarotic) human X chromosome. Based on the DNA from figure 2.2, this double helix is then coiled around itself to a diameter of 11 nm, coiled around itself again to a diameter of 30 nm, coiled again to a diameter of 300 nm and then wound in a 700 nm diameter spring like formation. The end result forms the 'sticks' that make up the chromosome i.e. 4 sticks joined at the same point would make up an X chromosome, 1400 nm wide and containing $4x10^6$ bases. It is called X because at this scale it is big enough to see under a microscope and really does look like an X; chromato meaning coloured and hence visible. Prokaryotic genomes tend to be much smaller and so can never been seen using a standard microscope. The genome of *E.Coli* is around 4.6 Mbp long and arranged in a loop, here bp is the most atomic unit of genetic measurement that counts base pairs of DNA. The loop is actually many loops each 50 to 100 kbp long with the join being on a set of proteins and attached to the cell

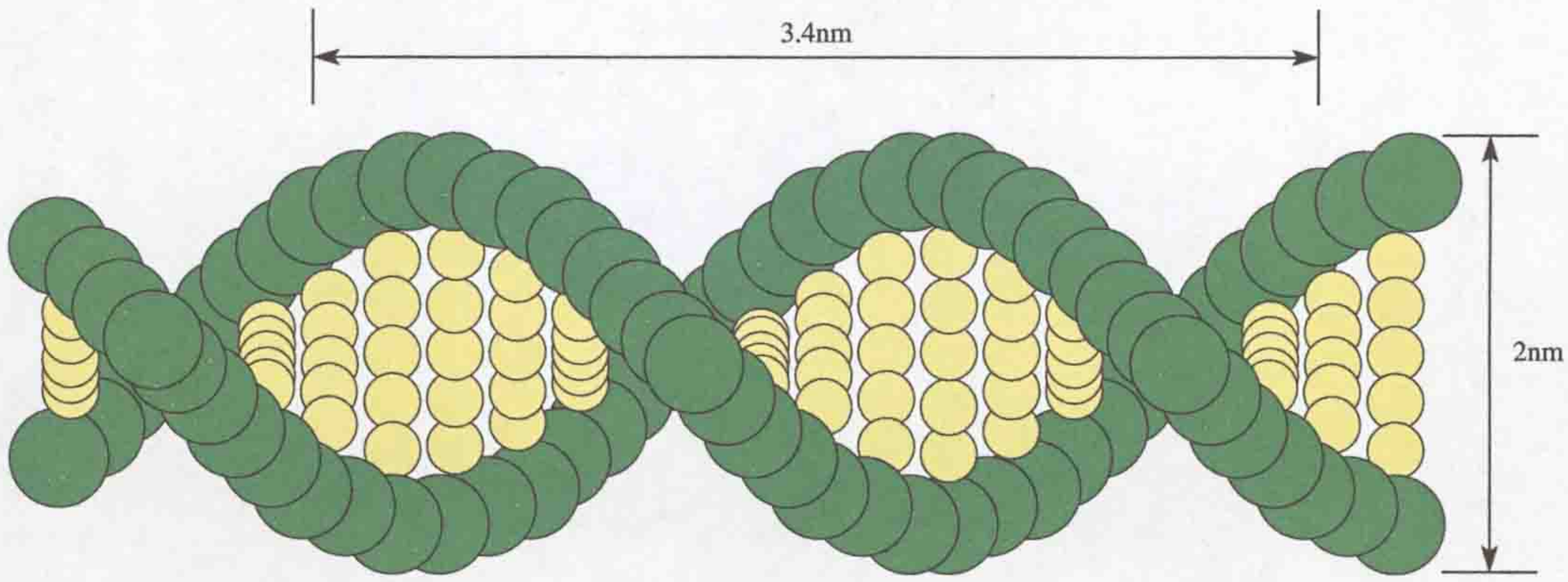


Figure 2.2: Double strand of DNA

membrane.

2.6 Transcription

Part of the central dogma proposed by F.H.C.Crick in 1958 states that RNA is created from DNA by a transcription phase, this RNA is known as mRNA (messenger RNA). The transcription process is carried out by an RNA polymerase (enzymes specific to the creation of mRNA), unless otherwise stated.

In [Bec96], Beckwith gives an account of the evolving Operon concept, the use of the operon as an explanation for gene expression. Jacob and Monad based their original model on research of the *lac* operon, an area of genome that had had a large amount of research effort put into it's understanding. Based on these findings, they came to a model that has a promoter site, followed by a operator site and then followed by a gene sequence of many genes (in the *lac* case it was 3 genes). The explanatory force of the operon idea together with Jacob's forceful style of reputing alternatives ensured its domination for a number of years. Here, Beckwith [Bec96] points out that research into other sites on the same genome shows that the operon model is only one example of a gene regulation mechanism. The operon concept has essentially a negative effect on gene transcription but research shows that other 'operons' have a positive effect on gene transcription. The concluding statement is that the

operon model managed to both create a new area of research and constrain that area into a conceptual model that did not fit all circumstances, this is clear from any simplified text which refers to the *lac* operon as if it is a defining example.

The first step in the creation of a protein from the DNA is transcription, this step being the transcription from DNA to mRNA. An RNA polymerase (a complex protein based machine) (Section 2.12) catalyses the transcription, the process requires the double stranded DNA as well as the nucleotides ATP, GTP, CTP and UTP (i.e. fuel). Like replication, transcription is directional and also starts at the 5' end. The 5' ended string is called the sense strand, the other strand is called the anti-sense strand. As bases attract their opposites, to duplicate the 5' sense strand the 3' antisense strand is actually read, the result being a copy of the sense strand. In *E.coli* the RNA polymerase moves at 40 bases per second at 37^{o}C, transcribing as it moves.

To start the transcription event, an RNA polymerase binds to the double stranded DNA, ideally at the promoter site. When the RNA polymerase has acquired all of it's cofactors it is referred to as a transcription complex. The start of the transcribed region after the promoter is known as the +1 position, the promoter and any operator sites are negative relative to this position. Again, as with replication the DNA strand must be unwound.

Transcription stops at the terminator sequence. This sequence contains a self-complementary region that can form a stem-loop or a hairpin structure out of the RNA product. This structure hints that the transcription complex should stop and bring on the dissociation of its constituent parts. The completed mRNA is released because there are four A residues on the DNA which do not bind well with the U residues on the mRNA.

Transcription termination can also be Rho dependent. The rho (ρ) protein appears to bind mRNA that is 72 nucleotides(nt) in length, it is expected that this is done by shape rather than direct mRNA encoding. It moves along the mRNA towards the transcription complex, where it forces the termination with an unknown mechanism.

2.7 Protein Structure

There are two classes of proteins, globular proteins can be regarded as spherical particles as they are folded compactly. Most enzymes are globular. The other class are fibrous proteins which instead have a high axial ratio (length/width) and are typically used in a structural role.

Given the chemical structure made up of a sequence of amino acids bonded to form a polypeptide chain, the protein takes a shape dictated by the polypeptide encoding; this encoding has a given lowest energy conformation that compels the polypeptide to form it's own specific shape. The structure remains stable because a variety of forces hold it together; hydrophilic side chains tend to the outside and hydrophobic amino acids remain on the inside. There are other forces too, as this is at the chemical level normally irrelevant forces have an effect. The protein's structure consists of several sections, an α helix and β sheets.

Except for catalytic RNA molecules an enzyme is almost certainly a protein. An enzyme is a catalyst for reactions that would occur but very slowly. COSMIC considers them essential, the large difference in reaction rates makes it not worth considering the case when the reaction occurs without the enzyme - as a result COSMIC ignores the passive components and just models the enzymes. Specificity from protein to protein can vary [TMBW97, p.22], from an exact match (e.g. glucose oxidase binds only glucose) or group specific (e.g. hexokinase binds a number of hexose sugars).

The encoding from DNA to protein function is ill defined (hence the protein folding problem [Hau97,Karplus *et al.*, 1997]) but some things can be specified. The bases are read three at a time (the codons) and allows for the production of 20 different types of amino acid that chain to make the polypeptide. This means that there are in fact 64 combinations, many more combinations than required but it allows inclusion of the start codon, three stop codons and the use of 1-6 codons encoding each peptide. It is important to realise that although the base alphabet is 20 letters in size, the polypeptide can form words of several hundred peptides in length; this combined with the fact that the shape (influenced by

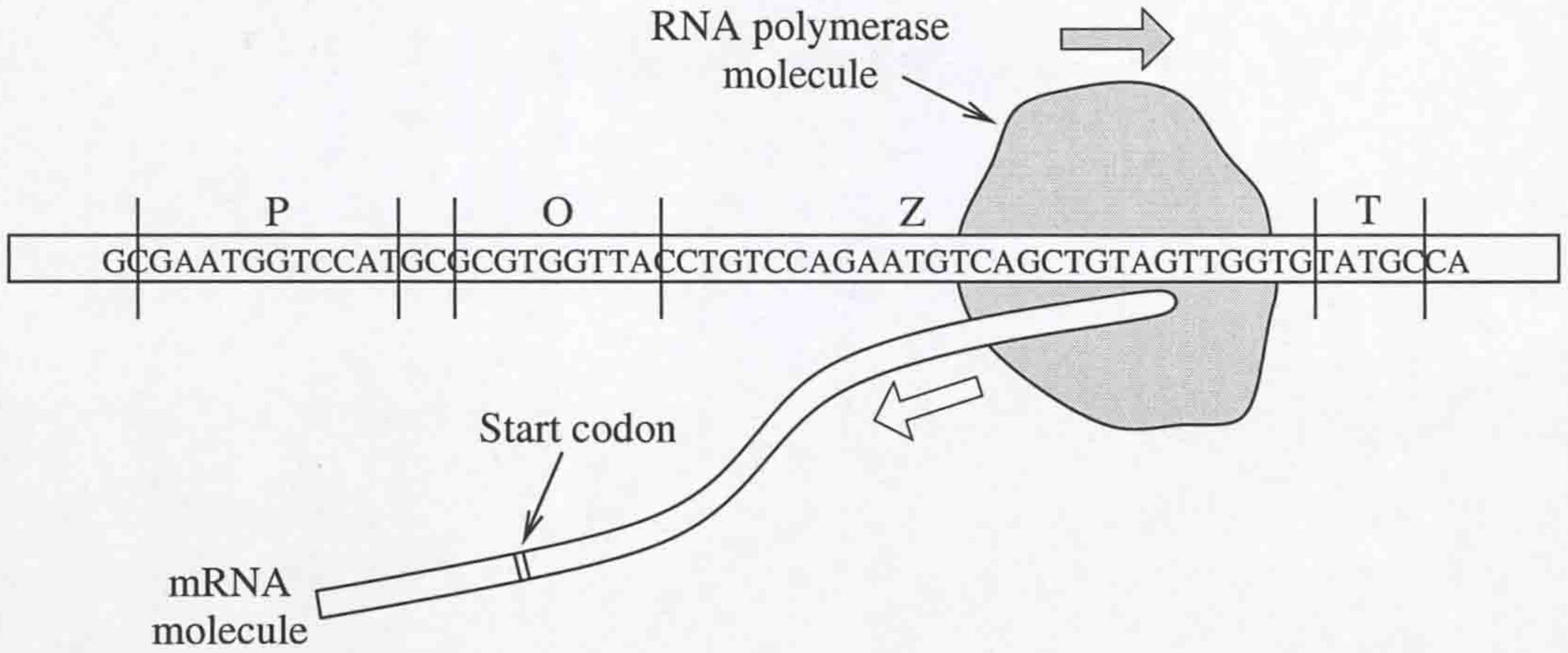


Figure 2.3: Generalised transcription

the encoding) dictates the proteins role then makes nearly intractable the task of determining the role of a real protein. The shape can take any imaginable 3D form (obviously the chain can not go through itself), using regular patterns that change over its length.

2.8 Optional Transcription

To further the level of complexity, there is the idea of gene expression and control, this is not some reference to the form of representation but implies the transcription agents (the transcription enzymes) are involved in the regulation of whether to transcribe a region of DNA into the required mRNA. With this idea comes the controlling proteins that are either transcription activators or transcription repressors - the activators having been found to allow themselves to be controlled by other activators.

The process of gene expression is partly explained by the Jacob-Monod theory, which is generally applied to prokaryotic gene expression rather than eukaryotic gene expression. The theory uses the unit of the operon that is made up of an adjacent group of structural genes known as cistrons (cistron being effectively equivalent to gene in modern usage), preceded by an operator region. The operator site forms a lock onto which a repressor protein (DNA-

binding protein) can attach. Once attached the RNA polymerase molecule cannot start transcription and so the proteins (or the various mRNAs) in that region are not created. If an inducer molecule is present it can bind to the repressor molecule and null the repressors effect. There is also the case of the corepressor in which the repressor will only stick to the operator region when it has already combined with another repressor of the required type. The regions on the DNA strand are shown in Figure 2.3 with Z representing the cistrons, P representing the promoter and O representing the operator.

These repressing (or possibly activating) proteins are created elsewhere in the cell, and so their presence or absence may indicate some environmental or genotypic state, which is obviously the whole idea - creating fairly rapid complex processing without a nervous system. Each operon type varies, Figure 2.3 shows the *lac* operon but it should not be taken as typical or even atypical, rather an instance.

Gene expression in the eukaryotic case is similar but has important differences. There are both short term and long term (irreversible) regulation effects. Short term expression regulates inducible and/or repressible enzymes. Hormones can bind receptor proteins and then enter the eukaryote's nucleus and activate transcription. Eukaryotes encode short term transcription factors in their genes (again without being localised) that help the RNA polymerase bind to the promoter and so lead to transcription. The long term signals are those which make a cell divide or become a particular cell type. As such they (and eukaryotes in general) are not as useful in this research, as they are too complex to be understood well. They might well be a source of inspiration but even genetic research (given the complexity) finds it hard to generate dependable results. Operon theory is hardly applicable to eukaryotes, it appears there are few if any structured genes and so global operons would have no purpose.

2.9 The *lac* Operon

The principal idea of optional transcription comes from research on the *lac* operon found in *E.coli.* Gene expression research has historically been based on the *lac* operon, as found in *E. coli.* Many other operons exist but it seems that supporters of the operon concept managed to put across their view so strongly that operons that do not conform to the *lac* model are given less attention. The *lac* example should be used as a guide of what can be rather than what must be. Unfortunately the *lac* operon was originally taken to be the latter and so has been taken too far and as well as being over simplified.

The *lac* operon is concerned with the use of lactose as a carbon source, the enzymes that can use lactose are only manufactured when lactose is available. The *lac* operon consists of the structural genes *lacZ* encoding β-galactosidase, *lacY* encoding a galactoside permease and *lacA* encoding a thiogalactoside transacetylase. β-galactosidase is an enzyme that hydrolyses lactose into galactose and glucose. Galactoside permease aids in lactose transport through the cell wall of the bacterium.

These three genes are encoded side by side in a single transcriptional unit called *lacZYA*. Relative to this there is an operator site O_{lac} between -5 and +21, just after the promoter site P_{lac}. If the operator site binds a *lac* repressor protein then transcription is strongly repressed. The *lac* repressor protein itself is encoded slightly upstream of the *lac* operons promoter in the *lacI* gene, this gene is also part of the *lac* operon. The *lacI* gene encodes the repressor protein but the protein itself is active only as a tetramer, the gene product only works when in groups of 4. Once this has taken place the repressor has a very strong affinity for the *lac* operator and also a high affinity with non-operator DNA. The *lac* operator site is in fact palindromic, it consists of 28 bp which read the same starting at either the 5' or the 3' ends, the *lac* repressor has the same symmetry when grouped in a four unit tetramer.

With the absence of lactose, the repressor protein binds to the operator site, though it is thought that this does not stop the RNA polymerase from binding and instead just stops its progress. Strangely, the binding of the *lac*

repressor to the operator site increases by two orders of magnitude the affinity of the RNA polymerase to the promoter site, making it quite likely that an inhibited operon also has bound RNA polymerase.

When repressed the *lac* operon generates a very low level of gene product. When lactose is present, the low level of expression allows its slow uptake, some of which is converted to allolactose. Allolactose binds the *lac* repressor, changing its affinity for the operator site and so forcing the unbinding of the repressor. As the RNA polymerase will probably be already present, transcription can start immediately. The removal of the lactose inducer leads to a quick inhibition of transcription, as rebinding of the repressor is almost immediate and the *lacZYA* RNA transcript is very unstable.

The promoter site P_{lac} and other related promoters do not by themselves have a strong affinity for RNA polymerase, the -35 sequence can be weak and even the -10 sequence can be weak. For high rates of transcription initiation, to increase the rate requires a specific activator protein called a cAMP receptor protein (CRP or Catabolite Activator Protein, CAP). CRP exists as a dimer that cannot by itself have any effect on transcription rate. When glucose is absent, the level of cAMP increases and CRP binds to cAMP producing a CRP-cAMP complex that binds to the promoter site slightly upstream from where the RNA polymerase would bind. The DNA is bent by the presence of CRP, forming a 90^o bend which is believed to multiply RNA polymerase binding affinity by 50.

In practice the location of the CRP binding site can vary much more between operons than stated here, the site can be on the promoter, next to the promoter or be much further upstream. The difference will obviously have some effect but it is not known exactly what.

2.10 The *trp* Operon

The tryptophan operon encodes five structural genes which are required for tryptophan synthesis. The RNA transcript produced is a single 7kb long

strand, starting from the operator site O_{trp}. As with the expression of the *lac* operon, the RNA product is unstable and so regulation of DNA quickly regulates the protein end product, which in this case is tryptophan.

The *trpR* operon is the source of the *trp* repressor and is located upstream of the *trp* operon. The operator sequence is symmetrical and forms the repressor binding site, which also overlaps with the *trp* promoter site between bases -21 and +3. The core repressor binding site is a palindrome 18 bp long. The *trp* repressor only actively binds the operator site when it has itself formed a complex with tryptophan. The repressor is a dimer and has a structural similarity to CRP protein and the *lac* repressor, the dimer needs two tryptophans to be complete. It is the tryptophan that gives the dimer structure the correct distance between its two reading heads and its central core.

The five structural genes encode for enzymes that produce tryptophan, tryptophan therefor inhibits its own synthesis by a magnitude of 70. Although not specified it is assumed this was under artificial tryptophan conditions. Regardless, this is much smaller than that caused by binding of the *lac* repressor.

The *trp* operon is like the *lac* operon, except that self inhibition is also playing an active role. As well as the normal transcription controls there is also an attenuator sequence following a leader sequence using around 162 nt before the first structural gene *trpE*. The attenuator formed by the transcript and is a short area rich in palindromic GC bases followed by each U bases. If this sequence manages to form a hairpin structure in the transcribing RNA, it will act as a terminator and force early termination at around 140 bp long, stopping before the structural genes have been reached.

The leader itself also has a role to play; divided into 4 successive sequences, 1 and 2, 2 and 3, and 3 and 4 are complementary, and so can bind to themselves to form a hairpin which stops further RNA transcription. If 2 and 3 bind then this does form a hairpin but does not stop transcription. Under normal conditions the binding of 1 to 2 and 3 are 4 are more favourable than 2 to 3. Also in this leader is an efficient ribosome binding site and successive codons encoding for tryptophan. Under conditions of low tryptophan availability the

ribosome would pause at this point. Since transcription and translation are tightly coupled in *E.coli*, the net effect then is to negatively control tryptophan transcription. In reality the hairpin formation between sequences 3 and 4 is more likely when tryptophan level is high, the pause occludes sequence 1 leaving sequence 2 to bind with sequence 3. In the alternative case, the pause occurs at the start of sequence 2 and so sequence 2 is occluded allowing sequences 3 and 4 to form a hair pin.

The polypetide formed from the RNA of these tryptophan encoding codons doesn't seem to serve any other purpose.

Given both forms of optional transcription, tryptophan dependent repressor and tryptophan dependent attenuation region, the total level of tryptophan can be amplified by 700 times. The attenuation sequence giving a 10 fold increase and the tryptophan dependent repressor giving a 70 fold increase. Generally, attenuation is present in at least six other operons with a role in amino acid synthesis. For example the *his* operon, but in this case the attenuation mechanism is the only means of control, there is no operator.

2.11 Operon Regulation

Becskei and Serrano [BS00] examines the effects of feedback in *E. coli* by adding a fluorescence gene (EGFP) to *tetR* gene. The *tetR* gene having direct feedback onto it's own operon sites. Three cases were tested, using an unregulated system, this feedback system and a mutated system. The results were obviously that the regulated system was able to control its own output, whereas the mutated version was less able and the unregulated was unable to regulate its output.

The numeric simulation used binding constants for the repressor ($2x10^{11}$ M^{-1}) and polymerase ($1.5x10^{10}$ M^{-1}), repressor degradation rate (10^{-5} s^{-1}), concentration of RNA polymerase (100 nM), promoter isomerisation rate from closed to initiating complex (0.3 nMs^{-1}) and proportion of mRNA over protein concentration (3.3). Most interesting was the use of more parameters than COS-

MIC. COSMIC considers proportions and concentrations to come from the integration of manufacture rate and degradation rate.

The relatively early paper by [Gil87] puts some perspective on the time scale of research into the genome. In it there is the surprising mention that introns are in the range 50 to 50000 bases long and yet exons peak at around 40 to 50 amino acids in length, this is in the context of their existence being only just realised and it is now known that exons are longer.

Generalising useful properties of the intron-exon structure, Gilbert cites that recombination is more likely using large introns, as a successful recombination need only be within the 10000 length of an intron, rather than on the exon boundary.

Gilbert then goes on to guess about the use of exons as modules. Using the idea of almost independent modules Gilbert then goes on to quantify the likelihood of chance mutation bringing about the 20^{200} amino acids and the 20^{20} exon modules - a much more likely scenario.

Bhalla [Bha00] discusses some of the factors that affect the feedback loop. These factors (these signals) can change the bistable region of a switch like action, a factor of 7 for P2A (Protein Phosphatase 2A) and a factor of 2 for MKP-1 (Map Kinase Phosphatase 1).

Bhalla shows the stable points in feedback reaction between MAPK (Mitogen Activated Protein Kinase) (ranging 0.1 fM to 100 nM, log scale) and PKC (Protein Kinase C) (ranging from 0 to 0.4 uM, linear scale), in the presence of different levels of PP2A (0.05 uM, .224 uM, 0.4 uM) (the response of PKC and MAPK was not directly affected). There is an negative-exponential like rise (with top end flattening) of the PKC/MAPK reaction, and intersections with different levels of PP2A show the stable points. For instance, PP2A=0.05 uM leaves a stable state at 110 nM MAPK/0.29 uM PKC and PP2A=0.4 uM leaves only one stable state at 0.0003 nM MAPK/0.9 uM PKC).

The above steady state results then transferred to a time series test on the scale of minutes, with 83 minutes between state changes. Given these different levels of stable points, the reaction can be pushed into lower or higher stable

areas of reaction rate, if those pushes push past the threshold level.

Also shown was the hysteresis of the MKP-1 (Map Kinase Phosphatase 1) reaction with PKC (Protein Kinase C). As MKP-1 varies slowly around 0.0025 to 0.0035 uM, PKC level is static until it changes sharply from 0.1 uM to 0.25 uM.

There are dynamic responses as well. PKC responds to calcium changes in seconds. The MAPK cascade creates an inertia with its multiple stages of phosphorylation, this creates a delay of around 5-10 minutes. MAPK is affected by it's regulator MKP-1, one of the results is that MKP-1 lasts longer (over 2 hours). Another slower result is that translocation of MAPK into the nucleus leads to synthesis of MKP-1, which is hinted could be the mechanism for cell proliferation or other major developments. The overall conclusion is one of stable reaction systems that can respond to small chemical changes. Although this was eukaryotic example, the same is true of prokaryotes.

Collado-Vides *et al* [CGE98] cite the example of the *glnALG* operon, which is in fact more complex than they described it. There is a regulatory region with two promoters σ^{70} *glnAp1* and σ^{54} *glnAp2*; and there are three genes, *glnA*, *glnL* and *glnG*, encoding glutamine synthetase, a nitrogen modulator II called NR-II, and a transcriptional regulator NR-I. NRI negatively regulates *glnAp1* and positively regulates *glnAp2*. Also, after *glnA* there is a terminator followed by σ^{70} promoter *glnLp* that is negatively regulated by NRI. Using their notation this is described by concatenating two transcriptional units but still suffers the problem that, in practice the *glbALG* can still be transcribed by a promoter further upstream of the two given promoters.

In another example that opposes the *lac* operon, Glansdorff [Gla96] enumerate the dispersed operon Arginine, one of the first to be found. It uses two promoters that face each other, gene amplification mechanisms that reactivate silent genes and a proven case where a transposon (IS*3*) carries a promoter - as well as a few other features of this pathway.

Pulling out from the large amount of detail, he gives the expression level ratios (micromoles per hour per milligram of protein) for all gene products

when the final gene product arginine is present (100 μg of arginine added) and when none is added. Ignoring the minor details, *argA* is 50, *argCBH* is 60, *argE* is 60, *argD* is 16, *argF* is 150-200, *argI* is 300-400, *argG* is unknown, *argR* is 15 and *carAB* is 50. This shows just how much the gene expression levels can vary, these effects come down to the layout of genes around the promoters and operators as well as other effects outside of transcriptional control.

2.12 RNA Polymerase

This section describes RNA polymerase as it exists in the cell. *E.coli* RNA polymerase is one of the largest enzymes in the cell and consists of 5 or more subunits. These subunits have the type names α, β, β', ω and σ, when in a complete polymerase they are collectively called the holoenzyme. This holoenzyme combines the types in the pattern $\alpha_2\beta\beta'\omega\sigma$, the entire enzyme being required for initiation of transcription. For elongation of the transcript only the $\alpha_2\beta\beta'\omega$ (the core enzyme) is used and the σ is released to go elsewhere. The polymerase covers a length of 60 bp but the binding site itself is estimated to be only 16 bp long.

The role of subunit varies and also the knowledge of that role varies.

- The α subunits are encoded by the *rpoA* gene. They have a role as central assembly points but the evidence from phage T4 infection only additionally suggests its role is in binding affinity.

- The β subunit is encoded by the *rpoB* gene and is thought to be the catalytic centre of the RNA polymerase, evidence comes from antibiotics studies.

- The β' subunit is encoded by the *rpoC* gene. It binds two Zn^2+ ions which are suspected of being involved with polymerase catalytic function. Evidence suggests that this subunit's role is binding to the DNA template.

- For the purpose of simulating transcription, the σ factor subunit is the most important. When the sigma factor binds to the RNA polymerase to form the holoenzyme, the affinity for the promoter site increases by a factor of 10^2 and decreases by 10^4 for non-specific DNA sites. Each type of sigma factor changes the affinity for a subset of the genome promoters, hence the optional transcription that COSMIC simulate. Unfortunately there are not as many sigma factors as there are promoters, there are only a few sigma factors and so only a few high level system states - no quantity estimate is mentioned.

After initiation and the RNA chain reaches around 8 to 9 nt in length, the sigma factor is released and free to complex with another polymerase, this is important as there is not enough sigma factor for all polymerase. Polymerase out numbers sigma factor by 2:1.

As a side note, not all RNA polymerases consist of multiple subunits, the bacteriophages T3 and T7 encode for smaller single polypetides, these can also transcribe RNA at around 200 nt per second at 37^{o}C.

Of the five subunits that make up RNA polymerase, only the sigma factor is considered important and so modelled in COSMIC. The other subunits provide some unknown enabling mechanism and so COSMIC assumes they exist. The next section gives details on a specific sigma factor.

2.13 σ^{70} Promoter

The most common *E.coli* sigma factor is σ^{70}, 70 comes the molecular mass of 70kDa. The σ^{70} promoter is a sequence of between 40 and 60 bp, the region around -55 to 20 has been shown to bind the polymerase. Studies have shown that it is only the sequences around -40 to 0 that are critical and that two 6 bp sequences around -10 and -35 are especially important.

The -10 sequence is TATAAT, with the initial TA and the final T being even more conserved. This can be referred to as the Pribnow box, after its discovery by Pribnow in 1975. Although the following 5-8 nt to the start of

transcription are not important, there number is.

The -35 sequence is TTGACA, with the first three bases being the most conserved. There is a 16-18 bp gap between this and the above -10 sequence, though this is not actually important despite occurring in 90% of all promoters.

The last few paragraphs have described the strong promoters. In reality there is much more variability in promoter efficiency, possibly as much 1000 times. Summing up, the -35 sequence is a recognition region and a site for the sigma factor; the -10 region is for DNA unwinding, which isn't reproduced here as it isn't seen as important; and finally the sequence at +1 for around 30 bases also influences transcription rate. The encoding affects the rate of separating the double strands and so has in indirect effect on the transcription rate when looking at a population of promoter sites. In *E.coli* the holoenzyme binds to promoters extremely rapidly, too fast to be explained by the binding and unbinding inside a liquid. Instead it is believed that the polymerase slides along the DNA looking for promoter sites.

2.14 Other Sigma Factors

The σ factor is required for the RNA polymerase to identify the -35 and -10 consensus elements of the promoter and so be ready for transcription. General control of transcription is achieved through repression mechanisms as found in the *lac* operon, but global large scale state changes can be brought about by a change in the availability of different types of sigma factor which have an affinity for different subsets of the genome.

Heat shock is an example of a major shift in transcriptional priority. Under conditions of extreme heat a new set of approximately 17 proteins are created. The cause of this is a unique RNA polymerase holoenzyme that contains sigma factor σ^{32}, this sigma factor is specific to heat shock genes. In *E.coli* a transition from 37 to 42^{o}C transiently triggers this protein response, under more extreme heat when the *E.coli* can not function this protein response is the only mechanism still functioning.

B.subtilis cells create bacterial spores ([PS84]) when the environment becomes hostile. The RNA polymerase is functionally identical to that of *E.coli* though here there is a "diverse set" of sigma factors, both for normal growth and sporation.

In the case of bacteriophages (for instance bacteriophage T7), it carries only its specific sigma factors (phage T4 in *E.coli* and SPO1 in *B.subtilis*). The latter is known to encode a series of sigma factors, each factor bringing on a stage of infection, the middle stage is brought on by the presence of σ^{28} which then activates genes 33 and 34.

2.15 Bacterial DNA Replication

Replication of the chromosome is tightly coupled with the growth cycle (and of growth at all). The *E.coli* initiation site (origin) is in the locus *oriC* and is bound to the cell membrane and hence the protein forming the junction with all DNA domains. *OriC* has 4 binding sites each 9bp in length for the initiator protein DnaA, this protein acts in proportion to growth rate making replication proportional to growth rate. When growth rate is high a second round of replication at the two new origins can occur before the first round is completed.

A sufficiently large concentration of the DnaA protein forms a complex of 30-40 molecules, each of which itself binds to an ATP molecule. This complex is then enveloped by the *oriC* DNA. Three AT-rich sequences 13 bp long are then melted, allowing the entry of the DnaB protein (DNA helicase). The helicase is a class of enzyme that melts double stranded RNA and DNA using the energy from ATP hydrolysis. This leaves a small gap between the normally bonded double helix, the amino acids in this gap are covered with single stranded binding protein to prevent base pairs from reannealing (renaturing). A DNA primase enzyme then attaches to the DNA and creates a short RNA primer that starts the leading strand of the replication. Bidirectional replication then follows, both strands of the circular DNA are followed. The process stops when

the replication forks meet at around 180 degrees opposite the *oriC* site.

2.16 Mutagenesis

The previous section discussed the mechanics of genetics, the next two sections mention what can go wrong and what are then the sources of evolution. The introduction to the genome earlier in this chapter points to the potential adaptation brought about by redundant information and helpful (or otherwise) plasmids.

Mutation is the permanent random alteration of bases caused by a variety of error sources including point mutation, replication error ($1{:}10^{10}$), radiation damage and chemical damage (which causes a mismatching of base pairs, most such chemicals are carcinogenic). There are many other specific causes of mutation, including the problems of encoding information with DNA. We will however concentrate on the more typical causes.

Point mutation is a single base change leading to it's paired base changing. The phenotypic effect varies, a mutation of non-encoding DNA will have no effect, mutation in the third base pair of a codon might also have no effect as the third base is not used as much as the first two. If the mutation has no effect then it is said to be silent. If the mutation has an effect, it could vary anywhere from not much to lethality. A mutation that generates a new stop codon is called a missense mutation and obviously leads to a short protein product.

In [Koc93] Koch outlines speculations and some findings of active adaptation in prokaryote bacteria. The work was with respect to extreme conditions, mainly starvation but also temperature shock. These being the typical method of changing the otherwise static environment of the agar plate.

As an introduction to the possibilities, Koch lists the following reactions to extreme conditions (generally termed unfavourable growth conditions):

- inducing metabolic mechanisms to utilise alternate resources
- expressing regulons for heat shock and other extreme conditions

- becoming able to move (motile)
- acquiring resistance to heat or dessication
- producing spores - in order to reproduce
- producing antimetabolities and other metabolites
- becoming competent - that is make themselves open to alteration by foreign DNA that survives internal restriction enzymes
- allowing invasion by plasmids, viruses, transposons.

Here Koch is not concerned with transported (external) genetic changes, only prepared genes, i.e. using the inactive genetic material from past challenges. It is initially aimed at the non-genetics, mechanistic side of bacterial survival - seemingly the production of spores that survive the challenge, but moves onto starvation response and mechanisms for directed mutation.

Koch cites experimental evidence for starvation inducing a metabolic braking function. This is brought about by a lack of, or non-functionality of proteins, ribosomes or enzymes. Koch also speculates the mechanisms for generating 'directed' mutations, but suggests that reverse transcription leading to a large gene change is probably not plausible as it requires large amounts of energy and will probably be lethal. This is suggested as a last measure, though the large amount of required energy would seem to make this impossible [COM89a].

Another source of directed mutation is the DNA repair mechanism during starvation. Repair of DNA requires a great deal of energy, and so during starvation (except for photoreactivation) DNA changes will go unrepaired. Under the assumption that the changes are positive, then the return of resources allows the changes to be repaired, either ignoring or using the changes. In an experiment it appears that repair has priority over replication, as improved resources lead to more mutants [Sta88]

Mutation rate can also be increased by the transcription mechanism itself. There is a non-generalised result that transcribing DNA is more mutable, as

the transcription is taking place on only one strand. The effect of this mutation source is unclear, it would seem to affect the average phenotype, if only slightly. Koch does however mention that there are repair mechanisms that deal with currently transcribing DNA [Dav89]. It would appear this is a secondary effect at most.

Taking simple mutation further, a model from [Hal91] shows the short term losses and long term gains of mutation by relying on a two stage mutation. The first mutation probably being lethal but almost certainly moving the phenotype away from it's optimum. The second taking the individual to an unoccupied space and so possibly a better optimum. Given reasonable conditions, the model predicts an unexplained huge increase in the second phase mutants. Hall also found that there is a general increase in the mutation rate when bacteria are under extreme starvation conditions. This mutation is probably lethal but in very cases it will lead to survival.

Koch [Koc93] finishes with the idea of the living genome with reference to mutation, that there exists physiological controls of selectivity. Koch suggests that the genes themselves have an effect on the mutation rate and on the transcription process. The example given is one of lactose uptake and the effect of lactose presence in the past on the *lac* operon. It appears this is a case of delayed feedback between the environment and the operon though Koch takes this further.

Looking at the role of mutation in more detail, [Rai99] makes the point that more rapid bacterial evolution isn't necessarily brought about by more rapid mutation, instead it has a more complex relationship. It was generally regarded that a mutator strain would only survive above a certain level of proportionality in the population, but lately (1995) it has been found that *E. coli* spontaneously develop mutator strains over long term lab experiments. A model by [Taddei *et al.*, 1997] shows that the relationship between fitness value and mutation rate isn't perfect; there are rules to the population size and the population's degree of fitness to the environment.

Evidence from [Sha97] goes against the positive mutation ideas. It seems

that bacterial mutagenesis gives two results; a lot of encoding effort is placed on the repair on genome, indicating that bacteria cannot survive mutation at a normal rate; and that cells possess a variety of biochemical systems capable of reorganising or changing DNA sequences (termed "natural genetic engineering"). The suspected mechanisms being both plasmids and transposons; therefore having the advantage of transmitting good material into other bacterial or other chromosomes - leading to a potentially large mutation in one step.

Shapiro supports Koch in the general conclusion that mutagenesis can be triggered by stress. Here mutagenesis means more than just mutation - it implies the bacteria actively incorporates the *Mu* phage under these stressful conditions. One result showed that the *Mu* phage didn't actually get incorporated in anything like its full form and so lost out overall. The process requires quite a few chemical factors to be present so would seem to be unlikely to occur by chance. Despite this, mutagenesis is speculated to be a widely occurring phenomena.

There is however no evidence that mutation causing agents from the environment or other cells actually targets specific areas of the genome, the adaptive view just states that more mutation will occur under certain circumstances. In the closing statement Shapiro describes his view of the genome as a source of massive computational power. The DNA is a storage mechanism and it is the action of internal and external influences that can activate this storage.

In [IS00] Imhof and Schlötterer evolve *E.coli* in nutrient rich conditions and found 66 advantageous mutations over 10,000 generations with 10 parallel independent growths. This works out to $4x10^{-9}$ advantageous mutations per cell per generation. Interestingly, this is the same number as found with experiments in a minimal mixture. Also, it is thought that deleterious mutation rate is of the the order 10^{-4}, shadowing the cells that suffered both a deleterious mutation and an advantageous mutation. Not surprisingly, mutation effects followed an exponential distribution, the more dramatic the change the less often it occurred. Also shown is the decline in population diversity that also

occurs when using genetic algorithms, good solutions occur but are suppressed by other better solutions; unlike those they cited, competition was always measured against the evolved population rather than against a non-evolved strain.

[RBL01] is another practical demonstration of *E.coli* adaptation. The test was for evolutionary change to the relatively high temperature of 41.5 degrees, over 2000 generations. The experiment identified five changes in expression level that were attributed to long gene replications rather than base mutation.

High density DNA microarrays were used to measure the expression level of all ORFs in the genome every 200 generations. The data shows three of the duplications where at least 23.7 kbp long and at 2.85 Mb on the chromosome, suggesting that this area is responsible for heat shock. Control *E.coli* at 37 degrees did not have this mutation. The paper then goes on to use further DNA analysis (Southern Blot) to conclude that all six cultures had evolved to the higher temperature but three had done so in a different way that the analysis didn't show.

2.17 Plasmids, Viruses and Transposons

This section briefly discusses some of the more recent thinking behind bacterial evolution that provoked this study, put simply the idea states that evolution is powered by the transfer of genetic information horizontally, rather than vertically through inheritance. Viruses and other forms of microscopic invaders that parasitise the cells of other living things are one source of this horizontal transfer, as are cases where a cell absorbs DNA from the environment. The original support for these ideas came from the seemingly obvious calculations showing random mutation is unlikely to be the sole cause of adaptive evolution.

Viruses are infectious agents about 20-300 nm long or wide, unable to multiply except in the living cell of a host; they are otherwise inert. A virus contains the genetic information in the form of DNA or RNA (but not both) along with the necessary virus specific (reverse) transcription factors that can be transcribed and so replicated by the host. mRNA viruses come in many

subtypes, but ultimately convey their message using some form of mRNA that is taken up during the transcription phase. The virus mRNA would typically take over the host cell, shutting down the original transcription processes but inserting itself into the host chromosome (using reverse transcription). It would then have the cell replicate copies of itself and then at some point late in the infection process, a dissolving enzyme would be produced that removes the cell wall and so allows the newly created multitude of virus mRNA to spread into the dead cells environment.

DNA viruses on the other hand are thought to have originated as plasmids - that is DNA encoding something beneficial to the organism (having evolved alongside the organism) but which is separate from its chromosomes - this genetic information presumably spreads like a helpful virus from cell to cell. The DNA virus is the example of a mutation making the plasmid turn on its creator, using the host for its own survival rather than for the good of the host.

Both mRNA and DNA viruses are good, bare bones examples of the selfish genes theory - this states that every living thing only exists to bring about the survival of the genes of that organism. It must be said though that viruses are not considered to be alive, since they are inert outside of the host cell, not that this seems to matter for basic survival.

A simpler plasmid like device is known as a transposable element and is also thought to have generated more rapid genetic adaptation than simple mutation alone. Unlike the plasmid, the transposon contains only 750 to 40K base pairs and consists of only DNA, there are no self contained transcription promoters. Transposable elements can infect more than just a host cell, plasmids and viruses can also be infected - this and plasmids in general are the cause of antibiotic resistance. A crucial difference in transposon action is that it has a less aggressive relationship with the host, it does not take over; the genome is altered by its insertion and this might well kill the host but if it doesn't then the survivor has a permanently altered genome that may be beneficial.

As these mechanisms were the original inspiration for COSMIC, it might be expected that they feature among the simulated mechanisms. Unfortunately,

the complexity of implementing the more basic mechanisms meant there was never time for these mechanisms, despite the results being much more interesting. This is however considered future work and as such there are many hooks inside the implementation by which these features can be added.

2.18 Further Background

Using the ideas talked about in in this chapter, namely the fact that DNA transformation has a hierarchical structure that comes into being through the presence of sigma factors and repressing/promoting enzymes (the logic enabling mechanism). We can then conclude with the view that the genome amounts to a hierarchical network of interacting genes, that synchronise the manufacture of necessary proteins in order for a cell to survive and replicate in its environment. This network being sensitive to its own internal state and the effects of the environment. It has been suggested that this network forms the chemical equivalent of a nervous system, having the same attributes but working on a slower time scale.

Work on replicating this exact structure seems to be sparse. There is work on evolving recurrent networks [PSD99] but not in a biological context. There is similar work on chemical reaction chambers and actual bacterial response ([BS00]; [HPTW95]; [BL97]), none of these use any kind of DNA like framework as the initial basis for representation. Work on replicating the structure does exist but it is by biologists for the representation and recording of known transcriptional data, complete genomes are available. There is no known work that tries to evolve a genome based network using a genome like representation and transcription mechanism.

Sticking to the biological evolution viewpoint, there are some questions that can be asked, such as how this structure came about, given the seemingly intractable process of creating such a network through random mutation - which is an improbably event. There must then be some additional mechanism, such as horizontal gene transfer that can adapt bacteria (the organism of study)

to different and changing environments.

It has been noted many times in the literature that mutation alone is unlikely to have the power to cross species water sheds, requiring many mutations at the correct loci to be effective. Putting this in the context of there being 4000 Kbases in the *E.coli* bacteria makes its correct change (even using natural selection on huge numbers of bacteria) very unlikely indeed. To solve this problem in the general case of sexual organisms, the traditional approach has been to say that sexual crossover will bring together mutations and so allow the offspring to obtain very different gene sets. This would still seem to be difficult to justify, especially in the context of bacteria as they are generally asexual.

Pointing to an entirely different answer to adaptation, the horizontal gene transfer theory says that external polynucleotide sequences will be incorporated into bacteria; these sequences will have come from other bacteria and so be loosely compatible. Research has suggested that bacteria have some partial control over their ability to pick up or reject these environmental sequences; depending on the bacteria's current level of health and the strain in question. Research has also noted that all bacteria seem to be able to incorporate foreign sequences and still maintain functionality. This in itself has three answers, bacterial cells can overcome the effect of the use of external sequences, or those foreign sequences really are compatible, or those sequences are not compatible and the cell dies - which is a traditional selectionist view.

Ochman *et al.* [OLG00] brings together some of the basic facts associated with common bacteria genome size and transposon rates. The main point being that different bacteria have different pickup rates depending on their environment (or lifestyle), but one of the larger genomes (that of *E.coli*) has 16% of its genetic material from external sources. Putting this into context, there is the proviso that this took place over 1 million years but this figure doesn't however take account of the fact that new information is being continually incorporated and old material lost - so the sequences that make up the 16% are continuously changing.

Considering only traditional adaptation mechanisms, there is also the question of how the chemical network is affected by mutation (hence the term dynamic mutation from Koch [Koc93] which is then countered by Shapiro [Sha97]). They talk about the hierarchical genome and mention the relative likelihood and evidence for various classifications of mutation. In the absence of experimental or even theoretical results, the conclusion would seem to be that bacteria (and probably any other species) puts a high priority on the repair of DNA. This implies that mutation is not good, but on the other hand, considering the ultimate source of genetic diversity, mutation is probably essential. Another aspect to mutation rate is that it places an upper limit on complexity, and so mutation repair mechanisms are required if more complexity is required.

As mutation would appear to not be the source of adaptation and transposons appear too infrequent, the question of adaptation still remains open. Ignoring the asexual characteristic of many bacteria, there is the use of crossover which would change parts of the chemical network. But then even if this form of insertion (a kind of super horizontal transfer event) was allowed, there are then the questions of how the network is robust enough to cope with the insertion of different sub-networks. There is also the question of redundancy - how much of a given network specification is actually useful for anything (i.e. used during 10 years of a bacterial family line). According to Ochman *et al.* [OLG00] most of this redundancy (at the level of DNA analysis) is somehow removed using natural selection despite that material being selectively neutral. It would appear that competition between genes for space on the genome somehow forces out genes that are of no use. How such a balance between useful and useless genes is achieved is unknown but the evidence says that most bacterial genomes are a constant length for a given species.

Linking in with the question of the source of adaptation and the selection against useless genes, there is the more abstract questions of how the network adapts to novel circumstances and then remembers those circumstances? How does the network organise itself so that remembered environmental circumstances could be reinstated?

Given the principle of the selfish gene, how does the enzyme activity manage to regulate itself in a way that forces it's metabolism (and ribosome/enzyme creation) to stay under control and not be overwhelmed by either its own genes or the genes taken from another species. Of course these events occur in the form of viruses, but a bacteria suffering from mutation or horizontal transfer of genetic material must also survive.

For our own ends, it would be useful to answer the above questions so that an artificially evolving network can do some computation. Of course in the form of artificial neural networks it already can; the above would provide another evolutionary neural network model, hopefully improving on the performance of current models by including all the right facets of biological evolution namely occasional feedback and evolutionary network design.

2.19 Summary

This chapter has covered the basic physical structure of *E.coli*, the genetic structure on which it is based and introduced the idea that this structure is a repository for solutions to adverse conditions. We have also discussed the sources of genetic diversity and the problem of where diversity and evolution comes from. The final proposition then is to build a model that can simulate *E.coli* at the genetic level and so test theories of evolution. That model become known as COSMIC.

Chapter 3

Computing with Biological Metaphors

This chapter moves away from biological material and instead focuses on simulation and analysis of biological systems, specifically with a genetics basis. Before COSMIC there have been many simulations of genetics, both for the sake of biology itself and biologically inspired algorithms such as Genetic Algorithms. This chapter mentions a few of those models and importantly their limitations. Analysis of the genetic output of COSMIC was also seen as important, and so this chapter also covers some methods that should be applicable.

3.1 Cell Models

COSMIC is not the only model based on cells, there are others but they all have different aims, different abilities and very different levels of funding. The vast majority aim to simulate single phenomena using the simplest tools deemed fit for the purpose. This would be a reasonable approach in engineering, but the biological problem has always been how to put the single parts (single processes) back together again.

Given in [SL99] is the general overview of the Virtual Cell. It starts by

justifying its position as an open ended framework for simulation rather than a fixed simulation. This avoids the problem of reinventing the wheel, should make testing more rigorous and ensures that all variables, equations and data are in one place and accessible. Obviously there are limitations to what can be simulated but the target of cell physiology and enzyme kinetics is open ended in that they can be specified through a HTML user interface.

The system encompasses a user defined cell physiology, using pictures as input in either 2D or 3D. The cell can then be compartmentalised into the cytoplasm, the exocellular region and nuclei. Each of these regions contains user defined molecules which can then further specified using kinetic equations. The diffusion between regions and inside regions is also definable. All of this is entered via the HTTP interface into a distributed server which can convert the data into executable C++ code, compile it, execute it and store the results along with the model in a database - there is however no way to query the DB despite some mentions of data analysis.

This model is quite large in scope, though it doesn't aim to be *E.coli* and doesn't really have the scope, it does try to be a framework for a generic single cell and as far as can be seen much effort has gone into making what it does do mathematically rigorous and yet be open ended with an accessible user interface.

There are things that it does not do, there is no reference to multiple cells and cell interaction , the processes are based entirely on enzyme kinetics without a genome. There is no scope for evolution, the model cell is specified and run, with the option of looking at the results. There is a mention of manually stepping through the solution to the reaction equations. Only a vague outline could be found as the author had so much to cover in very little space.

Moving in the direction of biologically inspired models, Eos [BSS00] is a population level simulation, a rapid prototyping framework for the simulation and experimentation of hybrid evolutionary algorithms and ecosystems. Uses object orientated techniques and Java, with plugins to extend the ba-

sic framework. The object frame work uses three main types, environment, sub-populations and individuals. The code is parallelisable using Java remote invocation and Voyager distributed agent tool.

Also included in the system are classes representing genomes, mutation, recombination, selection and replacement; all the evolutionary ingredients on top of the population classes that hold together the mixed groups of individuals in the environment.

Configuration is via configuration files, which are used to replace default values in each class. To aid rapid development, the classes not only have default values but now also have a front end.

Aerial placement for mobile networks - an example. Used a real valued genome, the only part that needed to be externally supplied was the fitness function. There was also a graphical front end that could be wrapped around the simulation, this was itself part of the simulation and allowed the automatic visualisation of the results.

Ecosystem simulation was based on Echo [JM93]. Individuals interact, trade, mate and fight inside the Eos space and interaction framework. The framework provides the ability to put the individuals in any space, 3D or something completely different. The only change then being the visualisation.

Co-evolutionary function optimisation, an optimum finding algorithm in a 2D landscape, the oddity being that there are three populations that fight between each other, the hope being that local optima will be more likely to be avoided as they aren't worth fighting over. The main reason for this is a demonstration of Eos implementing a different example of an evolutionary algorithm with little extra effort.

As well as these generic features, features such as graph space (a graph extension to the space framework) and a network simulator has been added. The network simulator is itself extended by a traffic simulator. These components leading to a simulation of network growth.

The HERBY system by Devine *et al.* [DPa97, DPb97, DPc97, DPd97] is an ecosystem simulator, but unusually is also individually based. Agents evolve

in a discrete world, search for food and reproduce when successful. The agents are controlled using a learning classifier system, i.e. An evolving population of control rules in each agent. The agents never tried to model any one organism, here the goal was modelling adaptive behaviour. This project was a predecessor of COSMIC, and as such has passed on some traits such as individuality and modelling adaptive behaviour. HERBY was however extremely abstract and that limited its application to real world data.

Ziegler *et al.* [ZDB97] used a more biologically abstract approach, giving an example showing signal paths from receptor input to flagella and involving 9 different enzymes in the simplified case and 20 nodes, 30 edges in the more complete case. There are around 20 receptors, 4-5 are used for chemotaxis, enzymes can be either inhibitory or excitory. Enzyme reactions overlap, A can affect D, E and F, and B can affect F and G. In short, this is something like COSMIC but on a smaller scale of implementation.

Shackleton and Winter [SW97] give another biocomputation model in which enzymes (no DNA component) can catalyse and have binding sites. Here, other enzymes represent data (operands) to the other enzymes binding site. There is also a binding site, which is supposed to convey function (in the real thing).

They believe that the system will not need to be programmed, but give a manually described example system of sorting that uses enzymes which can break a string of numbered tags if $v_l < v_o < v_h$ how v_o comes about is not clear, possibly a cause/effect of the need for topping up. And a joining enzyme which joins if $v_l < v_h$. There are many multiple data sets and data is given a velocity, and more unit length lists were added during the simulation to maintain 100 unit length data items in the sac - there was just 8 unique numbers.

Round 40 produced complete solutions, as shown in population graphs for each length. As unit lengths were used quickly, the ratio of joiners to breakers were changed 2:1 but had no noticeable effect. Stopping joiners from joining after already joining (for a short time only) had a better effect and unit length strings lasted longer.

The main focus was to use a GA to evolve the program. They talk about us-

ing a mapping from enzyme sequence to type of function (similar to COSMIC), but also mentions the differences between the primary structure (sequence) to tertiary structure (shape) and locality effects. In these terms the system is very much like COSMIC, but in the details their system has a much more computational than biological basis.

In [Holter *et al.*, 2001], Holter *et al.* put a mathematical model on real gene expression data. Three sources of data are used and it shows the ability to reproduce that data and also (importantly) to simplify it, in so doing this demonstrates how few of the genes need to interact.

In closing they reiterate that the mathematics fails when the number of genes exceeds the number of samples, and it always will fail; the solution is undetermined. The level of fit was high, though it does leave the reader wondering how many genes it was applied to. The data suggests 6 in one case, but the gene expression diagram seems to have a much higher resolution. Overall this shows what can be achieved given adequate data, but also that there are circumstances where that data will likely never exist to make the method practical.

3.2 Formal Process Models

A number of attempts have been made to formalise biological systems. Here two approaches and a general graphical representation are given as examples.

Duan *et al.* describe [DHB00] a formal language for biological systems, an initial assumption being that bio systems are best modelled as a mixture of continuous non-global variables (following differential equations) and non-global discrete variables. Discrete variables used for input/output of the system and in the test to determine if a state should be left. In each state there is a different set of equations for the system variables, but as they are differential they follow on from the previous state with ease. They propose a formal computable logic to specify the X-machine, the logic is based on the parallel processing formalism [DKH94] but includes time rather than the "time frame"

approach and is applicable to this model. The reason for this logic is to avoid (but still make possible) simulations, and instead find real mathematical facts based on the model.

Reddy *et al.* [RLM96] gives some of the alternatives to modelling, the descriptions are vague but seems to have much in common with Petri Nets. They state that the problem of reliable data means that quantitative analysis is difficult and qualitative analysis is the best compromise. To this problem steps the basic Petri net with weighted edges and some associated vector and matrix equations to relate state changes with some properties. These properties are largely unneeded but that is all the net seems to offer; given a starting marking (one of many possible) identify the deadlock situations or bounds on concentrations of each chemical.

The figures per place indicate a maximum of around 5 tokens representing the various levels, it is not shown but presumably the weights are chosen to act at specific knee points of the various reactions. The given example is of a metabolic pathway in erythrocytes (red blood cells), input to the system is glucose and ADP and the output is mainly lactose with ADP. The reaction path is linear except for the reliance on ATP.

The conclusion amounts to saying the above, that the net can qualitatively model biological processes but the inclusion of more power in the net (i.e. inhibitor arcs) means reducing the decidability of the final Petri net model.

For a more broad view of formal languages, Usher [Ush99] gives an overview of Petri Nets. Chapter 3 introduces different types of Petri Nets, starting with the simplest and then adding other features as the basic Petri Net was found to have limitations over the course of research. An interesting point was that Petri Nets with an inhibitor arcs have the same computational power as a Turing machine.

Of the types mentioned, some include some of the features that would be required for a biological model. Namely a temporal aspect and a hierarchy, but not at the same time. As the complexity of the types increase they start to look more like an object model with a Petri net providing the flow control between

objects or a Petri net providing the internal object control. This could only mean Petri Nets do not scale up without the help of the object paradigm, and so Petri Nets could not help with the formalisation of COSMIC. As shown later, it was the object paradigm that gave enough flexibility to specify COSMIC data structures but no known formalism had the scope to specify reactions between objects.

3.3 Analysis Methods

In order to help explain or at least analyse COSMIC results, it was hoped that some form of information theory would allow some understanding of the raw output data. This section therefor gives a brief introduction to information theory and then other related measures [Fel98, Hay99].

Given a random variable X on a discrete p.d.f., the Shannon entropy is:

$$H[X] \equiv -\sum_{x \in X} \Pr(x) \log_2(\Pr(x)).$$

The logarithm base sets the units, in this case bits. $H[X]$ here means the value of entropy for the p.d.f. X over all x, not for the variable instance X. It gives a measure of the overall uncertainty of the distribution, 0 being totally certain, 1 being a coin toss, $H[X]$ tending to inf as the number of x increase. It does not say what will be next or how predictable the next is given the past, each x is independent. The text also defines joint and conditional entropy, which might be important were it applied.

H can also be calculated from a continuous distribution:

$$H_C[X] \equiv -\int f(x) \log[2] f(x) dx.$$

H also represents the most efficient coding of a given p.d.f., hence the famous Shannon source code theorem:

$$\frac{1}{N}(\text{Average Length of an Optimal Binary Code on X}) = H[X].$$

Entropy density is a measure of the uncertainty over a substring L from an infinite string S. If L was S then $H[X]$ will diverge, so the energy density is defined as $h_\mu = \frac{\lim}{L\to\inf}\frac{H(L)}{L}$. Which doesn't diverge despite appearing too.

Following on from this is computational mechanics, which bring together entropy density and Markov chains (effectively probabilistic finite state machines). A matrix of transition probabilities and a set of transition states is required, this then allows the calculation of the entropy density but is not given here.

Considering information density in an encoding, they all have the same basis - a known p.d.f. from which the basic measures can be calculated. Measuring probabilities in terms of states requires the states to be known, the mathematics seems to be able to stretch over infinite sets but this obviously is not practical. To find the states, we would need to artificially create state sets, each state containing many of the same simulation states. Without reducing the states there would never be a large enough sample to estimate the p.d.f., assuming there is a p.d.f. behind the interactions. This approach appears possible but is a topic in itself.

Haykin [Hay99] also gives some time to information theory, or information entropy. This is similar to above, importantly Haykin goes into more detail on mutual information. For application to COSMIC, this seems the most relevant.

Mutual information is a measure of how much extra information knowing r.v. Y gives to X, first there is the condition entropy:

$$H(X|Y) = H(X,Y) - H(Y).$$

with $0 \leq H(X|Y) \leq H(X)$ and:

$$H(X,Y) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} p(x,y)\log p(x,y).$$

where $p(x,y)$ is the joint probability function of the random variables X and Y and $\mathcal{X}$, $\mathcal{Y}$ are their alphabets. The difference between the entropy of X and the conditional entropy of X given Y leads the uncertainty of the system input that is removed by observing the system output. The is called the mutual

information between random variables X, and Y and is:

$$I(X;\ Y) = H(X) - H(X|Y).$$

or

$$I(X;\ Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log(\frac{p(,xy)}{p(x)p(y)})$$

or

$$I(X;\ Y) = H(Y) - H(Y|X).$$

as

$$I(X;\ Y) = I(Y;\ X).$$

The problem with using this appears to be the p.d.f., figures could be obtained the p.d.f. was known but it is not. In fact it is an over simplification to consider the distribution a p.d.f.. Also the alphabet is hard to ascertain, if it is a vector then the mathematics is beyond the scope of this work. Continuous variables might lead to a solution, but it is impossible to guess that approach would be easier in practice. A p.d.f. could possibly be built up from a large sample (which could extracted), but that still does not deal with the random variable problem and the fact that a p.d.f is simply not appropriate. At any instance in time we could ask what is the probability of gene product x existing when y exists. That is the simplest case but probability does not take into account all the effects that play a role. And what information we could hope to calculate is seemingly contained in the linkage matrix.

Also given are equations related to special circumstances of continuous mutual distributions, these rely on the marginal probability - probability when there is no independence between p.d.f.s. A result from this is helpful in blind source separation, and on the surface this is what COSMIC requires, but in others (complex maths, hard optimisation problem to solve, continuous p.d.f.s.) seems far removed what what would help COSMIC. Quite obvious is the use (and assumption) of Normally distributed input vectors in all examples, this cannot be realistic in all cases of a biological simulation. The conclusion from this approach is that it could provide some measures of evolution speed and

convergence but its application is a research topic in itself, the theory makes important assumptions that simply are not true in the COSMIC model.

3.4 Biologically Inspired Optimisation and Learning

This section brings together some of the wide variety of papers on biologically inspired optimisation. The inspiration has not just come from the genome, it comes from a wide range of scales. There is the ant metaphor by Dorigo and Gambardella; Dorigo, Maniezzo and Colorni. There is the species diversity metaphor by Marín and R.V. Solé; and there are the models of ecological process [DPb97]. There is also other, less direct models or algorithms, such as alternative GA replacement or the multi-level classifier [DC94,Dor95,CD98].

In an approach to identify measures of diversity for use in COSMIC, the work by Cao and Wu [CW98a] was considered. Here they demonstrate a GA that uses population diversity to control parameters, here it is incorporated into the selection function.

Firstly, the Hamming distance between two strings $v_i = (b_1^i, ..., b_l^i)$ and $v_j = (b_1^j, ..., b_l^j)$, where $b \in 0, 1$ and l is the length of the strings, is:

$$H(v_i, v_j) = \sum_{k=1}^{l} b_k^i \oplus b_k^j$$

where $\oplus$ is the binary exclusive-or operator. The normalised Hamming distance is then:

$$\overline{H}(v_i, v_j) = \frac{H(v_i, v_j)}{l}$$

which limits $\overline{H}(v_i, v_j)$ to the interval $[0, 1]$. Using this normalised figure the population diversity can be estimated with:

$$D = \frac{1}{m(m-1)/2} \sum_{i=1}^{m=1} \sum_{j=i+1}^{m} \overline{H}(v_i, v_j)$$

where $m(m-1)/2$ is the total number of H function evaluations. Using this measure, the replacement policy of replacing the most similar strings was

adopted as opposed to replaced one of the parents or replacing at random. The measure showed that this policy lead to the higher diversity.

Cao & Wu then goes on to make this selection via a cellular automata. The reasons look vague, what is actually happening seems to be choosing a string, sorting this string on fitness and calling it one dimension, and then sorting on hamming distance (but bounded by the fitness boundary of the chosen string). Scaling of fitness to fit the automata matrix is not discussed. Those other strings in squares next to the chosen string (neighbours) are said to be the most similar strings. The algorithm then says to pick another string outside the neighbourhood and do the selection again. All very vague, it talks about separating the fitness of strings in two groups more or less of the current strings, then sorting the two subsets on ascending order of hamming distance to the chosen string. Then says this creates a mapping onto the 2D cellular automata with which we can pick the neighbourhood; it not clear how.

In [CW98b] give two improvements to the basic Evolutionary Programming algorithm, firstly in initialisation and secondly in adaptive mutation rate. Initialisation seems to be ignored in most texts, there is a massive search space and 50 individuals and so the variance in the population is very high. The technique here uses quasi-random numbers to generate the initial population. This simple deterministic hashing function has potential use in COSMIC, but will not be talked about further until COSMIC makes use of it.

The numbers are generated based on the way numbers are represented in the normal Arabic numeric notation. Given a natural number q and radix p, q can be expression as series of coefficients such that $0 \leq a_i < p$ in the form $q = a_0 + a_1 p + a_2 p^2 + \cdots + a_m p^m$, $m = \lceil \log[p](q) \rceil$. The quasi-random number is then $\phi_p(q) = a_0 p^{-1} + a_1 p^{-2} + \cdots + a_m p^{-m}$. In decimal notation p is obviously 10, the Halton QRS method requires that p is prime. In this application, a different prime is used for each dimension of the EP problem, q is simply the individuals number, i.e. $1, 2, 3, ..., m$. For any values of q and p, $0 \leq \phi_p(q) < 1$ but obviously this can be linearly scaled. An example in the paper shows how much better the spread is, though is still random.

The next improvement uses the Euclidean space between solutions to calculate diversity. As with the previous Cao & Wu paper, the distance is normalised over the maximum distance and then a mean found be comparing all members of the the population to each other. This results is a diversity measure $D(P)$, of the string population P. In this application, the diversity measure was then used to control the mutation rate per iteration by using it as a coefficient of the variance. The more converged it becomes, the lower the mutation rate.

Some examples are given that show this algorithm was better than standard EP, not just in terms of fitness but also in number of iterations required. The most important finding was of the importance of initialisation and the evenly distributed but random initial population. This series of papers was also a demonstration of hamming distances between strings, something that turned out to be very important for COSMIC.

Using ants as the inspiration, Dorgio, Maniezzo, Colorni and others have created a group of algorithms initially aimed at solving the T.S.P. using a distributed co-operation scheme with agents modelled on ants [DMC96]; each agent leaves a trail along edges of a T.S.P. graph. This chemical marker called a pheromone signals that an ant has used an edge but it also evaporates over time. Ants given no other information will use choose the shortest path (or random walk behaviour if path length has been withheld), but the pheromone will guide it along some path. The effect of evaporating pheromone is to guide ants along shorter paths. This particular paper thoroughly describes the ant colony system and its variations. Ant-cycle, which updates pheromone on ending of tour; Ant-density, updating of pheromone on crossing each edge - pheromone level taking no account of edge length and Ant-quality, in which pheromone on each edge is updated as it is crossed. All three are compared in a T.S.P. with various parameters. Ant-cycle comes out slightly better but there is only 0.5% difference. This is explained by Ant-cycle using global information, though it isn't mentioned that the others use emergent global information in the form of the pheromone. Further tests are then carried out only on the ant-cycle algorithm, which is tested on the *Oliver30 problem* [WSF89].

A sub-conclusion is that a balance of α and β (evaporation versus persistence) is required to achieve the right mix of following previous ant trials and applying the ant's internal greedy algorithm. Given the right parameters the algorithm will converge but without it will either converge too quickly on a sub-optimal cycle or never converge on any reasonable cycle. This would appear to imply a lack of robustness but trials did show that these parameters are relatively constant for problem size. Another good point is that the search does not stagnate even when the optimal is found, assuming the parameters are well chosen.

Another comparison used elitism. The best tour was given increased phoneme by increasing the normal phoneme increase parameters for some constant number of ants. This lead to much quicker convergence, but it is hinted that premature convergence is more likely.

Compared to other results, it is said to produce just as good solutions as special purpose algorithms and does reasonably well on asymmetric T.S.P. (asymmetric being the harder variety) in which the special purpose algorithms find difficult. However, the computation time was always longer than the special purpose algorithms. More detailed studies and slightly different approaches are in [CDM92, DG97] and [DG96] respectively.

Changing scale in thinking, [MS99] took an unusual population approach to single function optimisation. The population was in fact a kind of food web, with interdependent species that depend on each other for food. The network is formed by a matrix, elements determining levels of reliance (randomly initialised) and the sum of each row indicating the level of survivability - 0 or less indicating that there are insufficient prey to feed that predator. A mapping to the Macroevolutionary algorithm makes a link between species dependent on fitness and Euclidean distance between species. As well as general proofs of performance, it is shown to perform better than a GA and it maintains more diversity and an average better solution. Even the time displayed in ticks (rather than generations) shows quicker convergence to good solutions.

Changing the scale again to something akin to a single agent, Autonmouse

by [DC94] was a trained robot using a multi-level classifier system. The authors go to great efforts to put this robot in the real world, as well as defining a notation system by which they classify types of behaviour, i.e. a sequence, a sub-task (a combination of tasks are required for some behaviour), multiple independent tasks and the suppression of one task in favour of another. The structure of the problem was predefined, and a multi-layer classifier was statically assigned to the problem with a controlling classifier to oversee the co-ordination. Among the conclusions drawn were that it showed initial design helped the learning task. It must be said that this was a real world trial and so there were other aspects to the design (such as the sensor input) which is not mentioned here. It is also true that real world simulation is much harder than computer simulation.

In [CD98] the above work is summarised. It summarises all their robots and approaches, and states their main belief - that AI is truly hard and yet despite this it is worth tackling *harder* problems using real robots. They also try to define what an agent is and how much it has in common with a real agent. It is similar in that it must survive to perform its goal, and similar in that there are degrees in which it does this.

3.5 Biological Metaphors and Simulations

This section briefly mentions some of the ecological simulations that could be applied to bacterial interaction. This includes work in [DPA97, DPa97, DPb97] on modelling a classifier controlled agent, [HJF97] on modelling agents in a food web and [MK97] who simulate large scale mutation and transposon action. Not mentioned is work more closely related to the current work in chapter 4, this being BacSim [KBW98], Gecko [Boo97], Swarm [Swa00], simulation logic [DHB00] and reaction chamber languages [OLG00].

The Echo model [HJF97] was used to test ideas about the essential simulation characteristics of an ecosystem. Discreteness and spatial heterogeneity [DL94] effect the predictive power of conventional models such as ordinary

differential equations and reaction-diffusion (Turing) systems in a classical biology problem and so this model was used to avoid these problems.

The model consists of agents with an internal genotype, an external appearance, a single resource to trade and a reservoir of internal resources. Agents can fight, trade (for other resources) or reproduce. Tests for which type of interaction to carry out are done in this fixed order. Reproduction is activated when a agent reaches some global resource level (fixed in the simulation), crossover is in the form of two point crossover but is done at the gene level. Genes are variable length as the mutation algorithm includes insertion and deletion. Trading is based on an exchange of spare resources, it includes a form of bluffing as either/both can give nothing if it has nothing to spare. Hraber says this should be a good thing, but as implemented it favoured the bluffers as those that lose out in the transaction will be selected against. Migration occurs when an agent wants nothing from the current site and gained nothing in the last round. The world itself is a grid, only agents at the same point in the grid can interact. Each position on the grid holds resources and requires resources, in the form of a probabilistic tax.

To test the diversity of the agents the Preston distribution was used. The log of the number in each species is plotted on the x axis and the number of species in that size range are plotted on the y axis. Diversity then appears as a log-normal distribution of the form $y = y_0 e^{-(aR)^2}$ where y is the number of species falling into the Rth octive, y_0 is the distribution mode and a is a constant related to the logarithmic standard deviation σ: $a = (2\sigma)^{\frac{1}{2}}$. Given that a is about 0.2 [Pre48] it was possible to estimate relative numbers of species given the number of individuals or the number of species.

A further prediction came from the species-area scaling relation [Pre62]. Given an isolated region with an area A, the total number of species S is given by $S = cA^z$, where c and z are regression constants and for empirical ecological communities (assuming it uses a log-normal distribution) z is around 1/4. According to [HJF97] both this relation and the value of z can be predicted using the Preston distribution. In the natural world this relation does have

exceptions, on a very large scale (continental) it does not hold true. Using the Preston distribution [HJF97] was able to determine if the evolution of the agents was due to evolution or just some effect of random processes.

When talking about neutral models there is no mention that the environment affects species diversity. Regardless, mating was made random to discover if this does create species diversity. The results show that the original (limited interaction) model favours new (and single) species. This is shown on a Preston curve for both models, but is then avoided because of the distributions problems.

The conclusion drawn is that the model and real ecosystems are far apart and are likely to stay that way for every model. Abstraction, scale and unnecessary discreteness all play their part in distorting any possible conclusions that could be drawn.

In [MK97] a GA algorithm with transposon action is compared to a traditional GA - traditional except that there is no crossover. It is shown empirically that local convergence is avoided on a multi-modal function but also included are some vague concepts of food and competition. Food per peak was limited, so each peak could support only a limited number of agents. Therefore, the measure of multi-modal converge was that the total population went up and this was observed.

Each agent carried with it a mutation rate for both the micro-mutation (normal bit mutation) and transposon mutation. The standard 'micro-mutation' emulates base substitutions, small deletions, insertions and rearrangements, expansion and contraction of triplet repeats, and others.

Replication was controlled using food, those that obtain food (based on distance to food carrying peaks) can create a daughter per food unit. The calculation of the distance to a peak appears over complicated; the difference between a peak (ideal genome) and the agents is taken and raised to some power ('pressure') giving the creatures proximity.

3.6 Summary

As this chapter demonstrates, there is a long history of simulating bacteria and using the mechanisms of bacterial as a source of inspiration in solving more abstract problems that are not related to biology. What can also be seen from, for example, [CW98b] is how simple and abstract evolution can be, and in presenting models in the this form, equally how far from reality they can be. In building a model there is always the obvious temptation to simplify. This simplification brings clarity but it must also remove many of the possible interactions that may be important. It was this view, that a holistic approach on the limit of computational power, was the best method of simulating such a complex system as bacteria.

Chapter 4

The COSMIC Model

4.1 Introduction

Using the biological information from chapter 2, this chapter builds a computational model of bacterial growth and evolution. It does this starting from the scale of genetics while also including the scale of bacterial populations based in an environment. This is all based on the idea of modelling the individual, be it individual cell or individual molecule, and so will be explained in terms of sets and relations between sets and members of sets. However, this chapter starts with section 4.2 and section 4.3 describing the main biological phenomena that COSMIC models. Section 4.4 then discusses more detail of how such a model could be implemented in such a way that computation is feasible. Section 4.5 then starts with the model proper by detailing the construction of genome, from the genes and their encoding, the types of genes, the construction of operons and finally to the genome of an individual cell. This section then goes on to specify the other constituent parts of a single cell, building to a population of these cells in a specified environment. The purpose of this section was to describe that static structure of the model, which the later sections then build on to include the dynamics.

Section 4.6 describes the dynamics within the context of chapter 2, this highlights the important points of transcription and gives an overview of the

most important dynamic in COSMIC, namely the interaction diagram of figure 4.4. Section 4.7 then discusses how this dynamic aspect is incorporated into the previous formal static representation. Section 4.8 and 4.9 describes the mathematical functions that implement the state transition dynamics which are applied to the structures of sections 4.5 to 4.7. Section 4.10 describes the specifics of the interactions in the context of the representation and the mathematical functions.

Section 4.11 moves to a different scale, that of the cell population, by discussing the details of the environment in which these cells live. Having now described all the structures and possible interaction pathways, section 4.13 describes the initialisation of the system as a whole, how the original genomes come about and how enzymes can exist when there are no enzymes to create them. Finally, so that evolution may occur, section 4.14 describes the mutation operator that is applied to the previously mentioned structures.

4.2 The Model - An Outline

It is possible to think of the genome as a large data bank of protein creation instructions, and instructions for all the other cell processes such as division/replication, formation, tactics for environmental stress, environmental input, nourishment and so forth. This DNA or RNA data both creates the processors of the data and supplies data to be processed. Comparing to a tradition formal structure, it would appear to be a kind of Boltzmann network, except that the state is much harder to define. The message passing connections would appear to be proteins, but then actuators of some action (be it DNA transcription, movement, etc.) are also proteins.

If it was to represented on a graph such as Figure 4.1 then could be modelled using a node for each transcription site, activated by some edges representing and connected to positive and negative activation factors, these activation factors are themselves coming from other transcription sites. In the diagram the functions (marked f()) in each node represent the DNA transcription de-

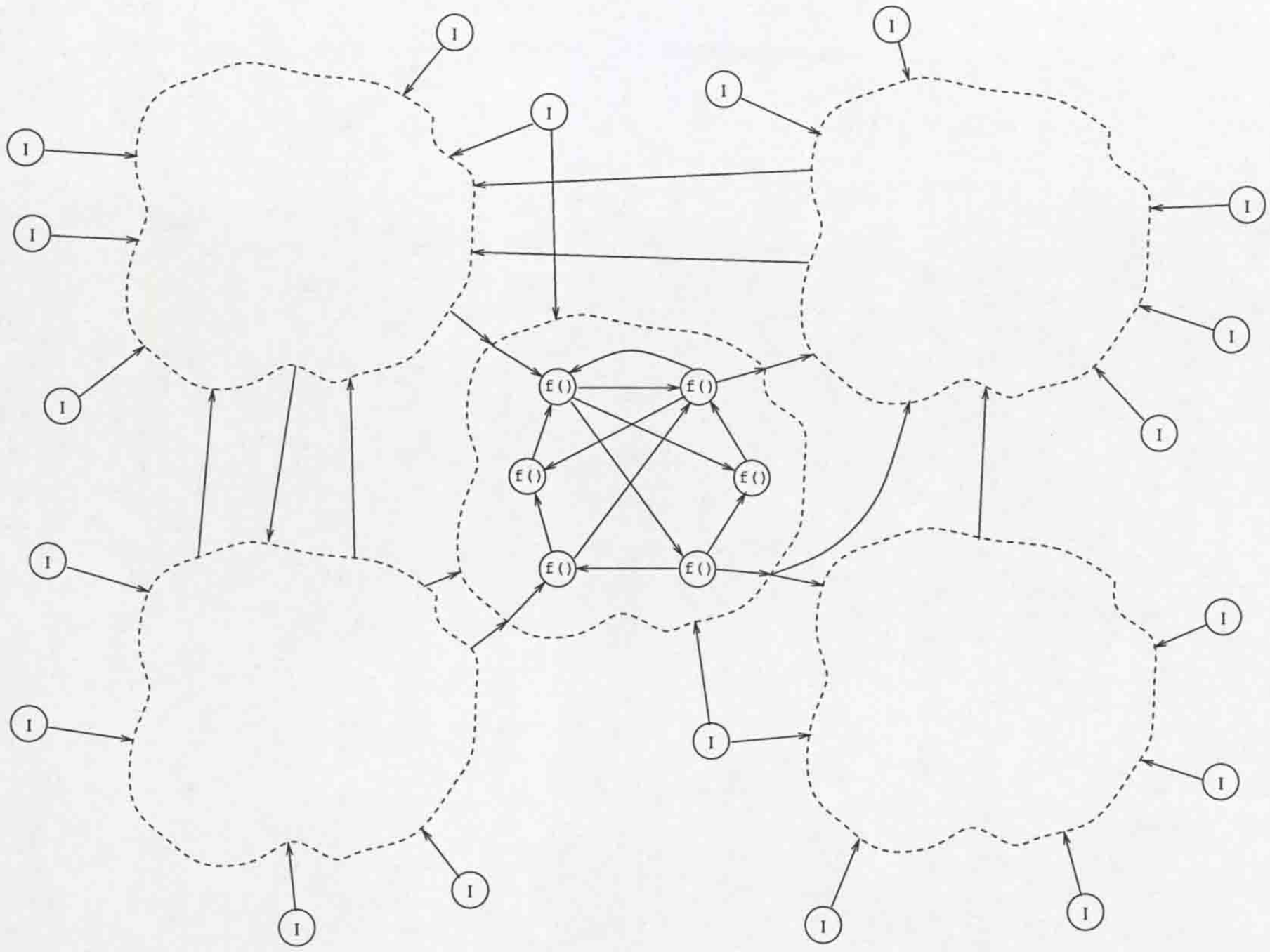


Figure 4.1: Conceptual outline of the network

pending on transcription activating proteins (the edges representing inputs). Nodes marked *I* are continuously transcribed and sensitive to environmental conditions. The outputs of each function are proteins which move inside the cytoplasm to activate other functions. System output is much harder to define, in an organism the output is the correct metabolism for the environment - this does not fit in the model.

The blocked nature of the diagram indicates a kind of context. Based on [YD02] it can be assumed that the hierarchical activation of transcription areas is massively layered, so allowing a kind of context to be activated; the all encompassing system looking like a hierarchically activated, loosely connected Boltzmann network that could could easily be well beyond any understanding. Still, all this then looks surprisingly like a programming language, with context maintained by the presence or absence of proteins.

As seen from the above references there is already work on computational models of biological systems, but there is a lot of variety when it comes to biological metaphor and it is questionable whether they have any value at all. It is hoped that COSMIC includes enough individual phenomena to be able to say it has many similarities - it will never be the same.

4.3 Model Realisation

In forming a model, yet more questions need to be answered. It is clear that biological processes can use massive parallelism and pattern matching capabilities without complex synchronisation. Including all the possible traits such as temporal delays, temporal dependencies, spatial positioning, varying enzyme concentration (generally implying a probabilistic activation function) and multiple inputs or outputs per gene all play their role in making a simulation intractable - so the question is which are the most important for an artificial simulation.

Implementing such a network also asks the question of how to go about training the network when a network of useful size will have a huge number of nodes and possible interaction modes - in other words there is the issue of survival goals and the concept of fitness.

The representation would seem to be a good place to start. Taking the example of *E.coli* which has a genome of 4000Kb. That is obviously a large number, but there is a redundancy from the genes being encoded loosely, there is no such thing as a gene template that can be overlaid onto the structure to read its meaning. As an example of this loose structure, it is known that intron/exon sequences in prokaryotes start with a repetition of TA bases around 30 bases before the sequence and an A rich area (though including some T bases) after. Note this suggests some sort of probabilistic identification, each base increasing the probability of that point being the start or end of an intron/exon sequence. Little information is available on the specifics of optional (i.e. repressor controlled) transcription, but this almost certainly works in a

similar way.

Needless to say, such a faithful representation will not be computable and yet there is definitely a need for a variable length representation if the size of the genome is to be allowed to grow a network of some (as yet not talked about) configuration. As a result, we suggest it is necessary to move away from the anonymous storage mechanism of DNA and move to a much more labelled representation. The labelling being chosen so that computability is conceivable but also the genome expressiveness is not constrained.

Genetic Programming might well look applicable at this point, its basis is the mutation of a program representation which appears similar to a genomic program. However, research would seem to indicate that GP relies on finding the optimal using a non-optimising Genetic Algorithm like algorithm, hence requiring huge amounts of power and a careful design to reduce the search space. This stems from the search for a correct program when it is obvious to any programmer that there is huge gulf between correct and nearly correct.

Bearing in mind that a typical Genetic Program tree is only a representation of some program, be it machine language or mathematical function, and that there is no framework for execution as this is done by the fitness function, then is clear that the problem with GP lies in the method of execution. The model proposed here does away with the sequential execution machine and instead executes a network of chemical interactions as specified by a GP program tree, the tree being very flat. An enzyme network at its most abstract level is an ideal machine for fitness evaluation, avoiding the halting problem by implicitly limiting execution tree depth. The only questions remaining are then which biological aspects to include in this machine, i.e. finding answers to the questions highlighted above. It is my belief that this machine could provide the power of an analogue computer, though to remain computable this would not be immediately apparent. Using a neural network like structure with some level of feedback, built from modules of primitive networks it must surely be possible to emulate a whole variety of behaviours found in all living things; or to be more tractable, bacteria and artificial systems.

As neural architectures contain a huge number of parameters, a suggestion taken from GP is to use Automatically Defined Functions (ADFs) to do the work of many smaller networks and so limit search space by removing a large number of the primitives. In the context of ANNs, this means first showing that it is possible to evolve simple ANNs, such as basic op-amp like configurations involving feedback or simple multiple input summation, difference, differentiation or integration; and then emulating their function by *Automatically Defined Neural Networks*. Although conceptually attractive from a computational view point, implementing ADFs poses problems equivalent to simulating a mixture of individual particles and particle masses; the particle mass is only an acceptable simplification if the mass takes into account its member particles but that then voids the computational benefits of treating as a whole. In short, it would have to be possible to switch between scales of view while maintaining coherence between the two views. Through implementation experience it has been found that maintaining coherence between views requires exponentially increasing implementation code for each common element.

4.4 Implementation Overview

The proceeding sections represented the rational for this work, such a model obviously needs to be implemented to find how feasible each facet actually is. The first stage would seem to be identifying the basic attributes of the network as mentioned above. Clearly the very most basic requirement is the use of transcription repressors which can be simulated using bit strings (or real valued strings) as tags that map the output of a gene to a repressor site that it restrains. The basic question here being how exact a match would be required and how many repressors are required. Standard options for DNA matching such as using the hamming or Euclidean distance being less than some ϵ are reasonably computable but it remains to be seen if they are accurate.

These simple questions only amount to deciding how well connected the graph or ANN actually needs to be, or in a biological sense, appears to need.

The matching tags acting as edges only form part of the basic outline, a secondary question is the choice of output function. As mentioned above, the transcription of a gene depends on many things, not necessarily just the repressors. Even the simplest choice of function would need to include the spatial and temporal aspects while using some simple exponential function. Intuitively this seems close to biological reality. It was hoped that a form of computational penalty for phenotype size would not be required, and that genotype size would instead be selected for or against.

So far the talk has been of a single bacterial cell, it's genome and the chemical network created from that genome i.e. the basic framework for a single bacterial cell. Any changes brought about by the genome network, (sequence insertion and deletion) have affected the single cell. Assuming the simplest case of a single celled prokaryotic bacteria then it could well be one of millions of individual cells, each trying to survive in some environment. The effects bacteria have on each other (aside from indirect effects of depleting the same energy source) have not really been looked at. The population of solutions approach is clearly the only way to create an evolved network, this is hardly an issue. The real issue is identifying just what the network is supposed to be evolving toward. As mentioned above, this could mean evolving analogue circuits consisting of op-amps, capacitors and the required control resisters; avoiding transistors and difficult to model components. In this case fitness is easy to define - though not necessarily easy to find.

The model could however be evolving realistic bacteria, in which case analogues of direct chemical processes are evolved to allow the whole to survive. Fitness in this evolutionary context is much harder to define, to simulate such a growth needs some additional environment as well as information relating to energy level and the ways in which it could be increased and decreased. Even for a first generation simulation this would be difficult, but this first realisation of COSMIC has achieved this in a reductionist way to leave only one survival goal.

There is also the question of just what are bacterial inputs and outputs.

They are known to take up and seek out calcium among other substances, but the only output would appear to be life. Metabolism of the surrounding nutrients is their fitness function but their abilities in achieving this through interaction with the environment are difficult to model. Glucose is a typical food source and yet goes through many stages of conversion, both to enter the cell and then to power it. The power glucose provides is very much chemical in nature and yet this simulation is trying to simulate evolution, the specifics of nutrient uptake are an unnecessary detail. The solution to this whole paragraph of difficult points was to identify what single high level feature could lead to life, and that was the uptake of nutrients leading to growth and finally division.

4.5 Genome Representation

The implementation of the genome and cell will be presented using a selection of set relations and functions representing reaction probabilities; this representation is in direct correspondence with the implementation. Also described is the environment and the input/output relations that link the individual cells to the environment. Simulations have shown that this model is tractable with current technology, a single machine simulates a single celled environment fifty times faster than real time.

Figure 4.2 shows a partial example of a model genome with a representative interpretation. The concatenation of string types is what is found in all living cells. To remain computable the boundaries are known and the tags seen in Figure 4.2 are set when a genome is created. Initial genomes are random, since initialisation must come from somewhere. Offspring genomes are derived from the parent, Figure 4.3 shows an example of type assignment to genes. Briefly put, the type is assigned by comparing the data sequence assigned to each of the fixed types of Figure 4.2. A strong anti-matching between gene and fixed type sets a genes type based on matched fixed type, as a result typing is dynamic and genes can have multiple types as shown in the example.

The strings used in the figures have a regular structure, the genome alpha-

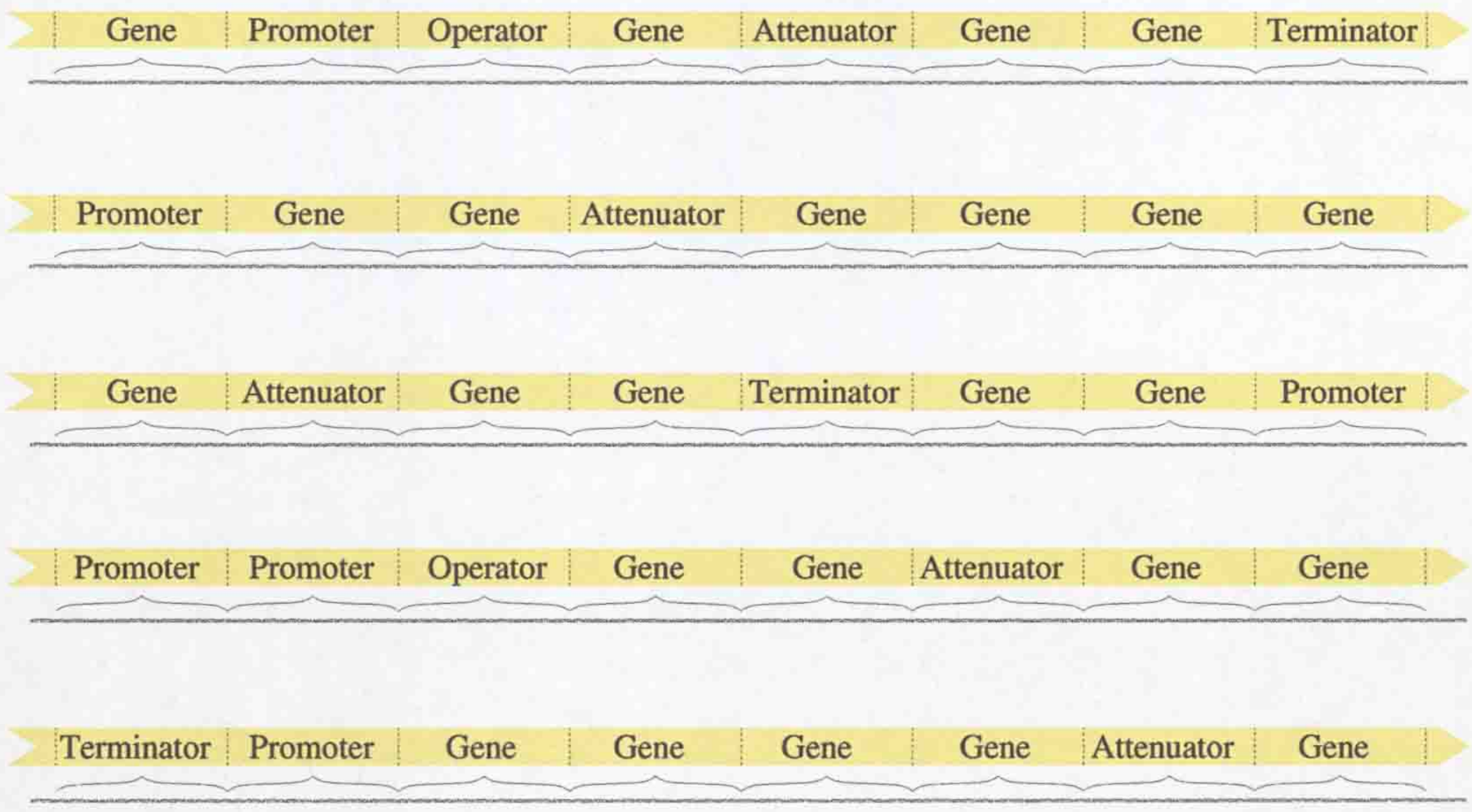


Figure 4.2: Flat genome structure, static representation

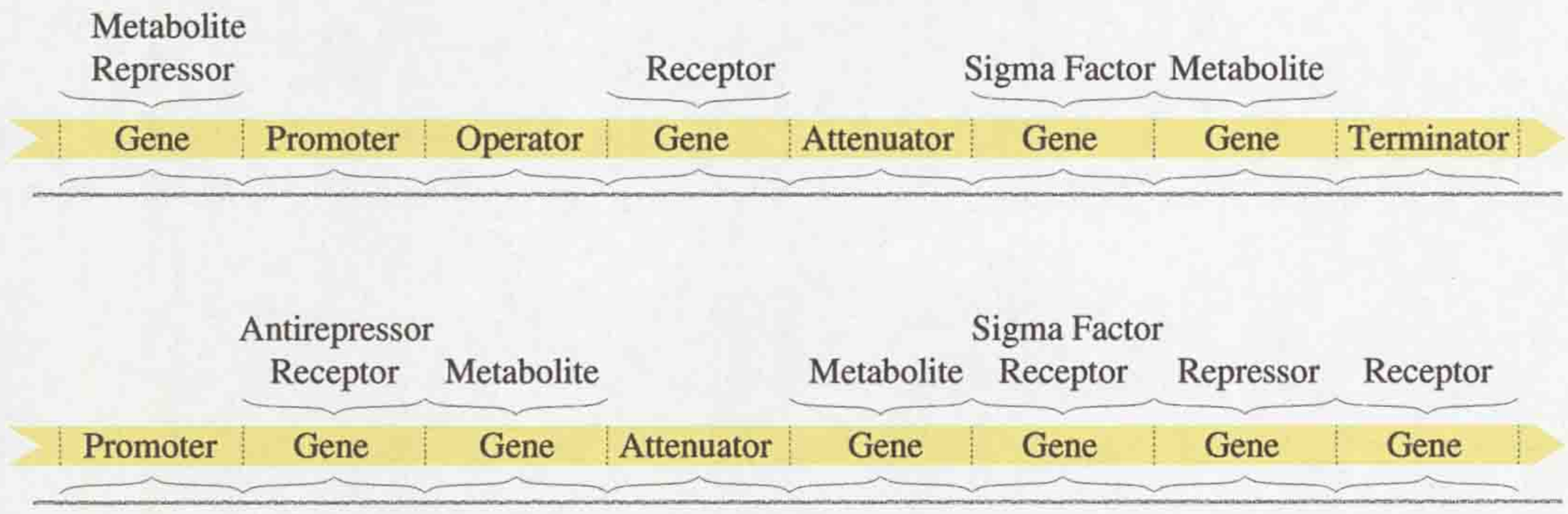


Figure 4.3: Genome structure, static representation with higher level meaning

bet is defined as $L = \{1, 2, ..., 4\}$. This is obviously based on the number of alleles per base of DNA. However $L = \{1, 2, ..., 20\}$ is just as valid because real DNA undergoes a two stage translation process leading to 20 different interpretations of the 64 combinations of 3 adjacent bases. This simulation ignores the RNA stage for reasons of computability and so the option is open to use either. The latter was chosen, mainly because it creates more diversity with shorter strings and can be more quickly calculated.

A single gene is defined as:

$$\Gamma = << L >^{\lambda}, \rho, \tau, t > \text{ where } 0 < \lambda < \infty$$

This includes a measure of distance ρ relative to the start of the genome and a type τ which records the attributes of this gene, attributes being the types of gene product that this gene transcribes. When transcribed Γ also is also used as a gene product, in this case the position ρ becomes the spatial position in the cytoplasm and t is an individual time since creation to allow for degradation using the half-life functions described below.

An operon is defined as:

$$U = << S >^{\sigma}, T >$$

Where:

$$0 \leq \sigma < \infty$$

$$S = \{P, O, \Gamma, A, T\}$$

$$\{P, O, A, T\} \in \Gamma$$

σ is the length of operon U. P is a promoter sequence having the same characteristics as a gene (i.e. a string representing translated RNA) but is never transcribed, only serving as a start point for RNA Polymerase. In this model the sigma factor plays the role of the complete polymerase complex.

O represents an operon sequence, A represents an attenuator sequence and T represents a terminator sequence. Again they all have the same characteristics as a gene sequence but are never transcribed, these strings only serves

as a site for binding repressors, or in the case of the terminator, stopping the polymerase. The specifics of how binding is simulated and what constraints are followed are given later, here only the data representation and an outline interaction is being defined.

The gene products of genes Γ are SF, Re, An, F and Γ itself. Strings have the following relationships:

$$SF \in \Gamma \text{ andanti} - \text{match } P$$

$$Re \in \Gamma \text{ andanti} - \text{match } O \text{ and } A$$

$$An \in \Gamma \text{ andanti} - \text{match } Re$$

$$F \in \Gamma \text{ andanti} - \text{match } \Omega$$

The degree of matching d is based on the individual string instances, the function defining this is given later. A genome is then defined as:

$$G =<< U >^{\kappa}> \text{ where } 0 \leq \kappa < \infty$$

To allow some path between the environment and the system there are genes (in representation) called input and output strings:

$$\Phi_{\phi} \text{ and } \Omega_{\omega}$$

Where:

$$0 < \phi, \omega < \infty$$

$$\Phi \in< \Gamma, a >^{\phi}$$

$$\Omega \in< \Gamma, a >^{\omega} .$$

a is a activation level associated with the input and a is a use level associated with the output. a is intended to allow for basic fitness testing by allowing the genomes cell to be placed in a competitive environment.

The possible interactions within the system are recorded by binary relations between strings, $Gi =< \Gamma_j, \Gamma_k, d >$, with d being a record of the anti-matching function value. A set of relations is then defined as:

$$Gn = \{Gi_1, Gi_2, ...\} \text{ where } 0 \leq |Gn| < |G|^2 - |G|$$

The total set of possible system interactions are divided into groups depending on the individual interaction type, giving the ordered set:

$$Ci =< PSF, ORe, RAn, ARe, \Phi\Gamma, \Omega F >$$

Where $PSF, ORe, RAn, ARe, \Phi\Gamma, \Omega F \in Gn$ represent promoter-sigma factor interactions, operator-repressor interactions, repressor-anti-repressor interactions, attenuator-repressor interactions, input-gene interactions and output-flagella activation protein interactions. Note that entries are not unique across Ci as genes have multiple types in their type mask τ_σ. Creation of the type mask τ_σ is explained later.

System state is a record of the interactions between enzymes that originated from the genome (transcriptional products), the promoter, operator and attenuator sites on the genome, and/or the enzyme that represent inputs and outputs. These are called the active interactions, the inactive interactions are the enzyme that at that instant are not involved with another protein.

Stated in a similar way to Gi, for active interactions this is defined as:

$$Sr =< \Gamma_\alpha, \Gamma_\beta, t, \rho >$$

Where t is the time since the interaction started and ρ is the mutual position in space. A set of interacting (bound) molecules is then defined as $Srn = Sr_1, Sr_2, ..., |Srn|$ being the current combined balance point between protein decay and transcription of proteins on both sides of the interaction. When a protein isn't currently involved in an interaction it is considered inactive and waits using a structure:

$$Su =< \Gamma_\alpha, t, \rho >$$

with t and ρ defined in the same way.

A set of unbound proteins is then:

$$Sun = \{Su_1, Su_2, ...\}$$

$|Sun|$ being the combined balance points of enzyme decay and transcription. Summing the occurrences of a particular enzyme in both the active Srn list

and the inactive *Sun* list gives a figure for the current balance between decay and transcription for that particular gene product. Again, it is worth pointing of that "enzyme" refers not just to actual molecules in the form of sigma factors, flagella activation proteins, etc., but of molecules binding sites such as the operator sites. This softening of the distinction between molecules and DNA allows for a more homogeneous definition of the interaction whose only downside is the unneeded time variables associated with each gene; in short it was a simplification for the purpose of specification.

The time and position fields in both the unbound molecules Su (t and ρ), bound molecules Sr (t and ρ) and the position and type fields in the genes Γ are discussed later, these values are part of the initialisation and runtime process.

Combining the above states gives a total cell state:

$$S =< \Gamma\Phi, F\Omega, AnR, SFP, ReO, ReA\Phi^S, \Omega^S, \Gamma^S >$$

Where $\Gamma\Phi, F\Omega, AnR, SFP, ReO, ReA \in Srn$ represent interactions of input - gene, output flagella activation protein, anti-repressor - repressor, sigma factor - promoter and repressor - operator. $\Phi^S, \Omega^S, \Gamma^S \in Sun$ and represent idle molecules of system inputs and system outputs, Γ^S being the gene products (flagella activation proteins (FAPs), repressors, anti-repressors, genes) and gene products/proteins promoters, operators and attenuators.

The cells state, interaction network and the genome are then contained in a cell that takes the form:

$$C =< G, Ci, S, \varepsilon, \mu, \vec{\rho}, i >$$

Where ε records organism energy level and μ recording matter level (cell mass). Movement and actions using or transforming the genome will change both the energy level and matter level. Movement by itself will change the position vector $\vec{\rho}$, the first two elements record the x and y position of the cell and the third θ representing an angle of orientation. $\vec{\rho}$ is a secondary concern and exists to enable placing the cell in an environment, bacterial tumble has been

simplified to this degree for this reason. i is the cells unique identifier, used to separately record cell heredity by the environment.

The environment is defined as:

$$\mathbf{E} =<< C >^{\eta}, E, t, \Delta t, t^r, P >$$

For testing purposed $|\eta|$ (cell population) is initialised to 20, environmental constraints then dictate the population size, which is dynamic. E is spatial lattice of the nutrients with an area of 1 but a resolution that allows the cell to feel a difference between receptors on different parts of its cell wall. t is the current time of the simulation, used as an absolute time from which all relative time events can be calculated. Δt is the fine grain iteration time step. t^r is the round time after which environmental state is synchronised between cells. P is the set of control parameters that allow for the effect but not the details of effects on the periphery of the model. It is:

$$P =< t^r, t^e, t^m, r^e, a^e, ar^e, f^e, em^r, c^r, s^e, s^m, me^r, |L|, |\Phi|, |\Omega|, \mu_{\max}, K_s, m >$$

where the values are given in table 4.1:

t^r would normally be 42 bases per second, but COSMIC enzymes are very short in length. In *E.coli* an average enzyme is coded from the range of 1500 bases long, taking 35 seconds. COSMIC genes are 10 to 15 codons in length but should still take 35 seconds. Because an average gene of 12.5 codons this leads to 2.8 simulation seconds per codon.

$Y_{\max}$ is already taken into account in $\mu_{\max}$. c^r is in fact 0.05 of the above figure, the above figure refers to free swimming in liquid. This adjustment also effectively changes the environment size to $2mm^2$ from 0.2 mm^2.

Initial genome distribution is of the ratios: Promoter:2, Operator:6, Terminator:6, Attenuator:3, Gene:48; this is one of the few distributions that is important but has no backup reference. There is no known quantifiable ratio of genes of given types.

t^e, t^m, r^e, a^e, ar^e, s^e and s^m are not currently used, ideally they should be but it remains to be seen how they can be used while maintaining a link with

Symbol	Value	Units	Comment
t^r	2.8	bp s^{-1}	Transcription rate (modified for alphabet)
t^e	0.0	gene^{-1}	Energy decrement per gene transcription
t^m	0.0	gene^{-1}	Matter decrement per gene transcription
r^e	0.0	event^{-1}	Energy decrement per repression event
a^e	0.0	event^{-1}	Energy decrement per attenuation event
ar^e	0.0	event^{-1}	Energy decrement per anti-repression event
f^e	0.0	event^{-1}	Energy decrement per flagellum use
em^r	$0.5\text{x}10^{-10}$	(m) (e)$^{-1}$	Energy to mass conversion increment
c^r	$3.125\text{x}10^{3}$	μm s^{-1}	Chemotaxis rate per flagella
s^e	0.5	s^{-1}	Energy decrement
s^m	0.99972	s^{-1}	Matter reduction coefficient
me^r	$1.5\text{x}10^{-5}$	s^{-1}	Glucose environment recovery rate
	$0.2 - 0.4$	fl	Initial cell volume with uniform distribution
	$10 - 15$	codons gene^{-1}	Initial uniform distribution
$\|L\|$	20	symbols codon^{-1}	Max loci per codon
$\|\Phi\|$	50	integers	Number of cell input receptors
$\|\Omega\|$	8	integers	Number of flagella
$\mu_{\max}$	0.046	minute^{-1}	Maximum cell growth rate
K_s	2.34m	fg fl^{-1}	Half-saturation constant
$Y_{\max}$	0.4444	fg(dm) fg^{-1}	Yield, dry cell mass per glucose
m	$6\text{x}10^{-4}$	fg(gl) (fg(dm)·min)$^{-1}$	Maintenance rate
	290	fg fl^{-1}	Cell dry mass density to volume coefficient

Table 4.1: Table of COSMIC parameters

reality. Following talks with Kreft these parameters are important but in the context of COSMIC are far too complex (being chemical in nature) to combine with the existing simulation, they would add an additional level of complexity that is considered simply unnecessary.

4.6 Genome Mechanics - Background

To explain the basic interaction mechanics that take place around the genome, the RNA polymerase, repressor and the DNA strings need to be explained in the context of the the more formal framework of the previous section.

The process needs an initiator that allows it to start running, one of the most important of these is the sigma factor, such as SF_σ from genome G_i transcribed from gene Γ_σ. In a real biological system the transcription complex that decodes the DNA is composed of two logical units, the sigma factor and the RNA polymerase (core enzyme). It is the sigma factor that makes the RNA polymerase specific to a kind of promoter, without the sigma factor the RNA polymerase would bind with low probability to any part of the genome. In a real cell, once transcription has started the sigma factor leaves the core enzyme and is free to move to another waiting core enzyme. Given that the RNA polymerase is the machinery and the sigma factor is the enabling element, the simulation does not have an RNA polymerase as an object, it is assumed to exist. As the RNA polymerase outnumbers the sigma factor by 3:1 [TMBW97] and as the sigma factor is free to continue once it has started transcription, it is reasonable to make this simplification.

So then, given a molecule called a sigma factor SF_σ that is a model of Su and was originally transcribed from a gene by a RNA polymerase. The sigma factor molecule will probabilistically match and stick to a promoter sequence P_k. In COSMIC this matching is based on anti-string sequence matching and the spatial distance between the loose sigma factor molecule and the static DNA string where the promoter site is found; matching functions and all possible

combinations of gene type matching are given later.

As the existence of the RNA polymerase is assumed, when the sigma factor has completed its role of transcription initiation it is released back into the cytoplasm. Starting at sequence P_k, the virtual RNA polymerase machinery will slowly move along the genome. As it moves it can come across five types of DNA sequence, they are: another promoter, an operator, an attenuator, a gene (or sequence of genes) and a terminator. Genomes are either random (when there is no parent, see 4.13) or based on the parent, in either case the ordering is not known.

If this promoter represented the start of a *lac* operon then it would be followed by a operator sequence O_{k+1}, and genes Γ_{k+2}, Γ_{k+3} and Γ_{k+4}. If the operator wasn't blocked by an attached repressor (which had been transcribed from a gene sequence elsewhere on the genome) then the RNA polymerase would continue along the genome and transcribe the three genes that follow the operator, these three genes would instantly be placed into the cytoplasm and be allowed to interact with any other enzymes that match in their reaction relations Gi_j, Gi_l, Gi_o, Gi_p, etc., any of these genes could be sigma factors for this promoter, for another promoter or for no promoter in the cells genome.

If the operator had an attached repressor then the RNA polymerase would cease in its transcription effort and unbind from the genome. Note that the *lac* operon has no immediate terminator. Also note that biological evidence suggests that polymerase moves along the genome rather than 'swimming' around it, transcription is too fast for it to be done any other way; references to blocking and removing are more metaphorical than actual but the simulation takes them literally as the true mechanism can only be guessed at.

If the promoter represented the start of a *trp* operon then it would be followed by a operator sequence O_{k+1}, an attenuator sequence A_{k+2} and five genes $\Gamma_{k+3}, ..., \Gamma_{k+7}$, all of which go together to encode for tryptophan at the RNA stage. As this model doesn't attempt to take account of the subtleties of the RNA stage it would be more accurate to use a single gene sequence Γ_{k+3}. This operon is quite different to the *lac* operon, not only does it have an

attenuator sequence but the operator is a co-repressor and the sole transcription product has a negative feedback effect on transcription of this operon. The attenuator sequence works like an operator except that it is more probabilistic in its action. A repressor bound to an attenuator has the effect of creating a 10:1 chance that the RNA polymerase will cease transcription, in the case of the *trp* operon the gene product is itself the repressor. COSMIC follows the 10:1 attenuator probability but obviously the negative self feedback is not specified and is left to evolution.

The operator in the real *trp* operon is a co-repressor that includes negative self feedback. The repressor comes from elsewhere on the genome, the co-repressor is tryptophan, the gene product of the *trp* operon. Without both binding together they cannot individually bind to the operator. COSMIC has no co-repressor type and its effect cannot be brought about directly. This alternative was chosen so as to reduce implementation complexity without penalising expressiveness.

Also included in the model is a repressor and anti-repressor interaction, neither example operon includes this latter type but it is known to exist and so can only enhance the expressiveness of the genome; one of the goals when identifying the interaction paths in the genome was to include a variety of inverting effects and allow evolution to select the interactions. The anti-repressor An_α is the sole means for removing a bound repressor Re_β from an operator O_{k+i}, obviously an important role for what should be bistable transcription.

Disregarding the *lac* and *trp* operons, in the real cell the transcribed gene sequences will (after conversion via RNA) become the RNA polymerases, sigma factors, repressors and anti-repressors necessary for continuous optional transcription, as well as creating the other necessary enzymes (complex proteins) to metabolise nutrients and so allow the survival of the genome contained in an individual bacterial cell. In COSMIC the same is true, the cell goes part of the way to surviving when transcription of the sigma factors, repressors and anti-repressors regenerates those lost to decay.

The other key to cell survival is a sustained correct response to environmen-

tal stimuli. In COSMIC this is achieved through a very much simplified model of receptors, cell mass and flagella culminating in a set of inputs Φ_1, Φ_2, ... and outputs Ω_1, Ω_2, ... Each input receptor Φ_i has a position on the cell wall which, to make the simulation fair and genomes transferable, are in identical positions on all cells. Based on the cell position and the receptor position the value of a for each Φ_i can be calculated. a is then used as a probability of matching input Φ_i to a receptive gene product Γ_j. Probability rather than the real case of existence or non-existence is a necessary simplification as many inputs would otherwise be needed. In the real biological case the receptors would be the start of a conversion process that makes energy and matter, the basic building blocks underlying transcription. The simulation cannot hope to take into account the chemical process and instead a bound transcription product Γ_j cannot be bound with anything else.

Outputs Ω_k have a position on the cell and are linked directly to flagella in the environment. If a flagella activation protein (FAP) Γ_l was to bind an output Ω_k then the environment simulates motion. At the moment motion is calculated by summing a vector of all flagella positions, ideally this should be perpendicular to the cell to simulate a tumble rather than a push.

The combination of input reward based on cell position and cell position based on flagellum output produces an indirect reward based system that is the basis on which the simulated *E.coli* evolve.

A summary of possible interactions, sources of transcription products and a map of indirect or direct attenuation is shown in figure 4.4. Each type is created from the same genes above and as mentioned before, the assignment of type of based on the anti-matching of genes to operators, promoters, attenuators, inputs or outputs rather than being specified at (for instance) initialisation.

This chart is discussed more fully in section 4.8 where the edges are tabulated with weights and depend on both molecule age, degree of anti-matching and distance in space between molecules.

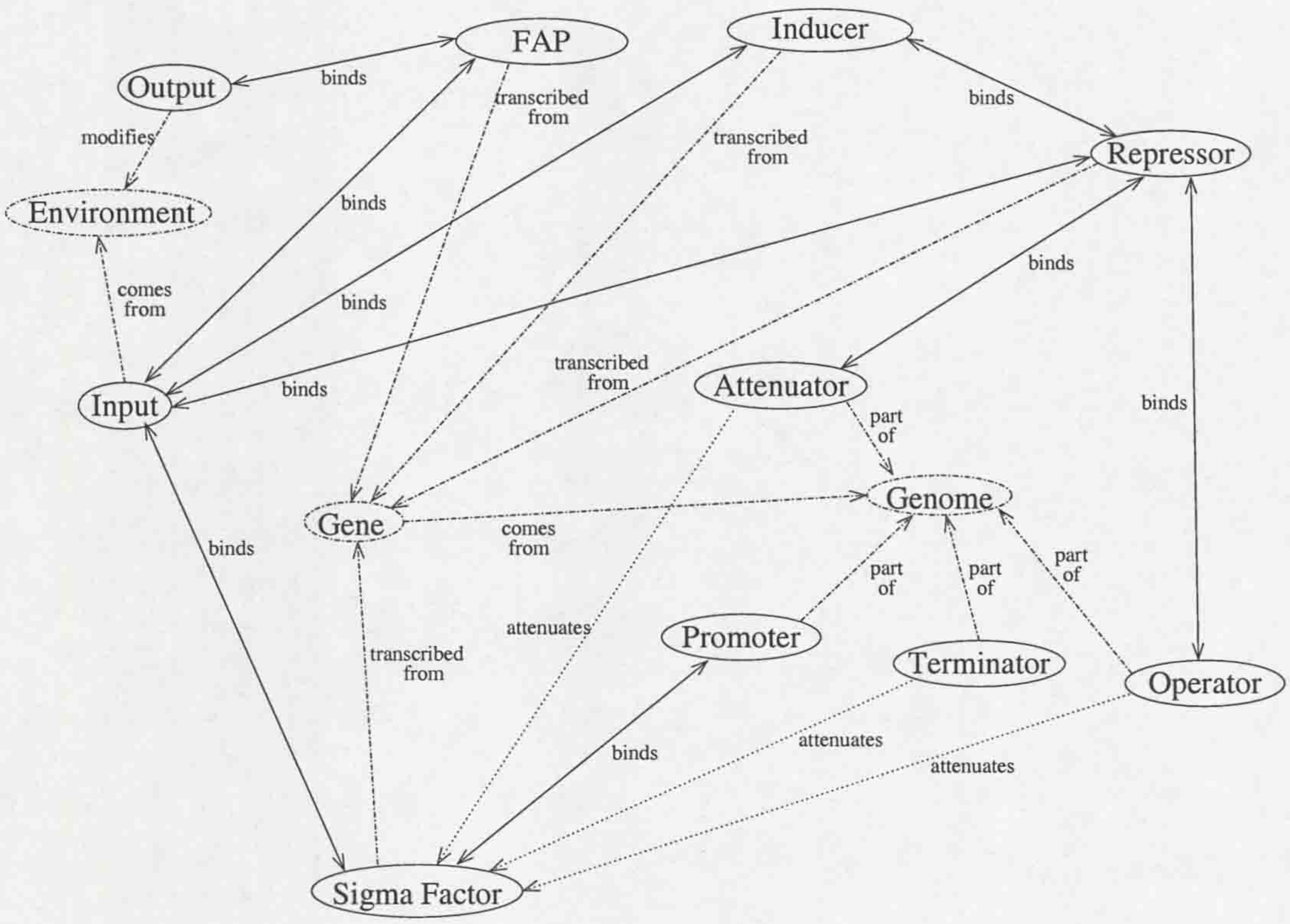


Figure 4.4: Possible "enzyme type" interactions

4.7 Network Creation - Assigning type

The relationships introduced above in figure 4.4 link transcription products, areas of genome and system input/output into a list of possible reactions, that list being occasionally referred to as a network; or more specifically a template for network creation. This is done on the type τ_σ associated with gene sequence Γ_σ, be it a real gene sequence in the case of the genome (P, O, A, T, Γ) or imitation gene sequence in the case of the input and outputs (Φ, Ω), all these relations define the contents of the set Ci in C.

As has already been mentioned, type assignment is dynamic and defined in terms of what each gene sequence can bind with. This does not however make the assignment arbitrary as some gene sequences already have type assignments; input and output gene sequences are marked as such during initialisation. Promoter, operator, attenuator and terminator sequences are all of a fixed type. All that remains is the assignment of type to the gene products themselves; possible types being sigma factors, repressors, anti-repressors, flagella activation proteins (FAPs), receptors and 'genes'. This latter type is a default for gene products that do not fit any other types. Note that some static type assignment is necessary, without it only a matching between Γ_i and Γ_k could be found, whether this represented an input/receptor, output/FAP, sigma factor/promoter etc. cannot be known, one side of the interactive pair must be given to resolve this ambiguity.

As promoters are known, their sigma factor counterparts are identified by being valid anti-matches to the promoters. This amounts to identifying the subset of gene sequences from the set of all sequences Γ_i in G, testing each possible pairing of promoter to gene sequence. This gives:

$$PSF = \{< P_\alpha, \Gamma_\beta >, ...\} \text{ where } \forall P_\alpha \in G, \forall \Gamma_\beta \in G, \quad \text{match}(P_\alpha, \Gamma_\beta) < \epsilon$$

match() being the anti-base matching function defined below and ϵ being the match tolerance, 85 for test purposes (giving a high interaction level) and around 15 for use during the simulation - an ideal value of 1 is far too low for computable evolution.

Repressors can also be easily identified by being a valid anti-match to operator sequences. Again this amounts to identifying the subset of gene sequences from all possible sequences, including those gene sequences that were found to be sigma factors. Testing each pair of gene sequence to operator sequence gives the set:

$$ORe = \{< O_\alpha, \Gamma_\beta >, ...\} \text{ where } \forall O_\alpha \in G, \forall \Gamma_\beta \in G, \quad \text{match}(O_\alpha, \Gamma_\beta) < \epsilon$$

match() and ϵ being the same as above.

Repressors that bind to attenuators can also easily be identified by being valid anti-matches to attenuator sequences. Again this amounts to identifying the subset of gene sequences from all possible sequences, including those gene sequences that were found to be sigma factors, repressors, etc. Testing each pair of gene sequence to attenuator sequence gives the set:

$$ARe = \{< A_\alpha, \Gamma_\beta >, ...\} \text{ where } \forall A_\alpha \in G, \forall \Gamma_\beta \in G, \quad \text{match}(A_\alpha, \Gamma_\beta) < \epsilon$$

match() and ϵ being the same as above.

System inputs bind to any gene that can become a transcription product (i.e. $\forall \Gamma \in G$) regardless of any type already assigned, so identifying gene sequences affected by input sequences involves using the anti-base matching function on the set of all possible sequences $\Phi x \Gamma \in G$. This results in the set:

$$\Phi\Gamma = \{< \Phi_\alpha, \Gamma_\beta >, ...\} \text{ where } \forall \Phi_\alpha, \forall \Gamma_\beta \in G, \text{ match}(\Phi_\alpha, \Gamma_\beta) < \epsilon$$

match() and ϵ being the same as above. Note that this relationship has been given the name input-gene, despite gene itself being a generic term.

System outputs also bind to any gene sequence that can become a transcription product regardless of any already assigned type, so again identifying gene sequences (FAPs) affected by output sequences uses the anti-base matching function on the set of all possible gene sequences. This gives the set

$$\Omega F =< \Omega_\alpha, \Gamma_\beta > \text{ where } \forall \Omega_\alpha, \forall \Gamma_\beta \in G, \text{ match}(\Omega_\alpha, \Gamma_\beta) < \epsilon$$

match() and ϵ being the same as before.

Anti-repressors might cause a problem, they represent the only interaction that has no fixed type on either side. One approach to this is to wait until all repressors have been identified using the above and then use the now familiar method of creating a working set - this remains the current method. This gives the interaction set:

$AnR = \{< \Gamma_\alpha, \Gamma_\beta >, ...\}$ where $\forall \Gamma_\alpha, \Gamma_\beta \in G$, $\text{match}(\Gamma_\alpha, \Gamma_\beta) < \epsilon$ and $\tau_{\Gamma_\alpha} = \text{Repressor}$

match() and ϵ being the same as before.

For the future there is the possibility that anti-repressors are identified by the sole condition of being an anti-base match to a gene sequence. That is, gene sequences are matched to gene sequences and if the anti-base matching is within tolerance then that is considered a valid relationship regardless of the possibility of there being no operator affected by the repressor. Of course, there is then the question of which is the repressor and which is the anti-repressor. If the repressor is not involved with an operator then it is of no concern and so this form of identification remains a practical possibility. It is plausible to use this mechanism to penalise genomes whose loose matching creates islands of activity; this activity can be set-up to use cell energy and so reduce the cell's fitness. This approach remains a possibility.

Terminators are not involved in any direct interactions, there was the possibility of having the the virtual RNA polymerase test sequences with the terminator but this is not biologically accurate; in reality the RNA polymerase machinery is stopped by the terminator by virtue of the terminators coding sequence snagging the machine, so the gene sequence reads as *stop* no matter how it compares to the polymerase.

4.8 Reaction Rates and Probabilities

This section discusses the construction of tables that specify the probabilities of transfer from an unreactive molecule state to a mutually reactive state, these functions simulate a half life as well as taking into account some other

details such as molecule position. In effect these probabilities reflect the use of edges from figure 4.4 at the level of the individual. Also included are functions for deciding molecule removal based on probability and degree of matching (or rather the degree of anti-matching). In the future it would seem more accurate to place less emphasis on a probability density function (p.d.f.) based on type and instead use a mapping function from gene encoding to p.d.f. that is also (to a lesser degree) based on type - otherwise identical enzymes will have vastly different life times, as is the case at the moment.

4.8.1 Potency matrix

This matrix is read as "y directly affects potency of x" with non-interaction shown as $-$. This is a time dependent molecule degeneration as a function of: $p(t) = e^{-t\alpha}$ Where t is the age of that molecule. A value of 0 indicates there is a relationship but has no effect in that direction. Some points to note are that the inputs, outputs and the gene based imitation enzymes (operator, promoter, attenuator and terminator) are timeless. The input and output enzymes represent the interface to the environment and so there is nothing to be gained from ageing and then replacing them - which would also have to be done using a mechanisms outside the transcriptional mechanism. The latter are part of the gene, which ages along with the cell's genome. In this model the genome is considered to be static in most time frames so the elements of the genome are static and therefore timeless. Evolution is implemented as an entirely independent stage and should not be considered here.

In Table 4.2 the first column shows degeneration of an idle molecule, other columns show the degeneration rate under that given relationship, with t being the ages of both individual enzymes. Here the gene Γ is used as a central point of interaction between the other string types, it could be said they are the centre of the system.

The individual potency coefficients in the *Self* field are used for ageing of individual enzymes, all other uses of potency are for the binding or unbinding between pairs and so uses the relevant pair to obtain the coefficient. The choice

$\alpha =$	Φ	Ω	Oper	Promo	Met	Rep	Poly	Anti	Atten	Prot	Recp	Γ
Φ	–	–	–	–	–	–	–	–	–	–	0.05	–
Ω	–	–	–	–	0	–	–	–	–	–	–	–
Operat	–	–	–	–	–	0.01	–	–	–	–	–	–
Promot	–	–	–	–	–	–	0.01	–	–	–	–	–
FAP	–	0.05	–	–	–	–	–	–	–	–	–	0
Rep	–	–	0.01	–	–	–	–	0.01	–	–	–	0
Poly	–	–	–	0.01	–	–	–	–	–	–	–	0
Anti	–	–	–	–	–	0.01	–	–	–	–	–	0
Atte	–	–	–	–	–	–	–	–	–	0.01	–	–
Prot	–	–	–	–	–	–	–	–	0	–	–	–
Recp	0	–	–	–	–	–	–	–	–	–	–	0
Γ	–	–	–	–	0.01	–	0.01	0.01	–	0.01	0.05	0.01

Table 4.2: Potency matrix, providing a coefficient of reaction rates

of row/column or column/row generally follows the same pattern; the row is the enzyme that ages, the column is the imitation enzyme that does not age. The only exception is the anti-repressor/repressor reaction, as both age the ordering is more arbitrary but is ultimately fixed - the repressor is the row.

4.8.2 Enzyme matching matrix - from non-reacting state

A probability (or rate) of matching another enzyme is given by:

$$b(d) = e^{-\beta d}$$

taken as a function of β and depending on d - the distance between the proteins in space ($|\rho_\alpha - \rho_\gamma|$). $-$ shows there is no chance of interaction. As gene types (i.e. what an enzyme can react with) are determined solely by their encoding, a single gene can be (for instance) both a FAP and a repressor. In this situation $s(d)$ is calculated for each case.

4.8.3 Enzyme matching matrix - from reacting state

When in an interacting(reactive) state the probability is given by:

$$b(t) = e^{-\gamma t}$$

and provides one coefficient of remaining in that state. t represents the age of the reaction. This is the same as the above matrix, though a brief mention

$\beta =$	Φ	Ω	Oper	Promo	Met	Rep	Poly	Anti	Atten	Prot	Recp
Φ	–	–	–	–	–	–	–	–	–	–	0.05
Ω	–	–	–	–	0	–	–	–	–	–	–
Operat	–	–	–	–	–	0.01	–	–	–	–	–
Promot	–	–	–	–	–	–	0.01	–	–	–	–
FAP	–	0.05	–	–	–	–	–	–	–	–	–
Rep	–	–	0	–	–	–	–	0.01	–	–	–
Poly	–	–	–	0	–	–	–	–	–	–	–
Anti	–	–	–	–	–	0.01	–	–	–	–	–
Atte	–	–	–	–	–	–	–	–	–	0.01	–
Prot	–	–	–	–	–	–	–	–	0	–	–
Recp	0	–	–	–	–	–	–	–	–	–	–

Table 4.3: Enzyme coefficients from a reacting state

$\gamma =$	Φ	Ω	Oper	Promo	Met	Rep	Poly	Anti	Atten	Prot	Recp
Φ	–	–	–	–	–	–	–	–	–	–	0.05
Ω	–	–	–	–	0	–	–	–	–	–	–
Operat	–	–	–	–	–	10^{-7}	–	–	–	–	–
Promot	–	–	–	–	–	–	0.01	–	–	–	–
FAP	–	0.05	–	–	–	–	–	–	–	–	–
Rep	–	–	0	–	–	–	–	0.001	–	–	–
Poly	–	–	–	0	–	–	–	–	–	–	–
Anti	–	–	–	–	–	0.001	–	–	–	–	–
Atte	–	–	–	–	–	–	–	–	–	0.01	–
Prot	–	–	–	–	–	–	–	–	0	–	–
Recp	0	–	–	–	–	–	–	–	–	–	–

Table 4.4: Enzyme coefficients from a non-reacting state

by Kauffman [Kau93] suggests that repressors remain attached to the genome until actively removed. This has been implemented but is beyond the scope of this matrix.

4.8.4 Protein matching matrix - direct compatibility

As well as the time and space dependent probability functions for state change, there is also a probability state change based on direct protein matching - that is direct anti-matching of gene sequences. As each gene Γ is potentially a different length, this must be taken into account as the expectation of the difference in length. A close match is the basic requirement for the above

interaction probabilities to take place. This matching defines the interaction paths (Gi) between potential individual enzymes in the system. Once calculated, the other interaction probabilities above can play their part until such a time as the network (Gi) becomes out of date. This is a necessary step as calculating the anti-matching degree for all possible genes is computationally expensive. As a result, this step is carried out when the genome is first created and then as any new additions are added to the genome, such as sequence insertion, or from future extensions to COSMIC such as plasmid mechanisms.

The anti-matching function is defined on the set of all gene pairs $\Gamma_\alpha, \Gamma_\beta$ with the function:

$$\text{match}(\Gamma_\alpha, \Gamma_\beta) = \frac{(\frac{b}{2})^2(|\Gamma_\alpha| - |\Gamma_\beta|)^2 + \min_{k=0}^{|\Gamma_\alpha - \Gamma_\beta|-1}\left[\sum_{i=1}^{|\Gamma_\beta|} bc(\neg\alpha_{(i+k)} XOR\ \beta_j, b)^2\right]}{|\Gamma_\alpha|}$$

Where $|\Gamma_\alpha| > |\Gamma_\beta|$, b is the number of alleles per locus and $bc(x, b)$ returns the normalised number of mismatched alleles (bits) per loci, i.e. normalised by $b = 4$ or $b = 20$, depending on chosen DNA sequence model. $(\frac{b}{2})^2(|\Gamma_\alpha| - |\Gamma_\beta|)^2$ is included to remove the bias for genes of different lengths. This gives the expected random error that is equivalent to extending the shorter gene to be as long as the longer gene. Without this the shorter sequences would obviously dominate by matching more easily.

Figure 4.5 shows an unnormalised cumulative distribution of 32 random genomes with an average of 1000 genes each. Frequency is on the vertical axis and distance on the horizontal axis, exact units are not meaningful. It is expected that this sort of graph will give an indication of convergence, as the average distance between genes will increase taking the mode to higher values. As can be seen, even increasing the genome sizes to 1000 still produces a large amount of sampling error, which is removed by the curve showing the overall trend.

Building on the text of section 4.8.4, the connections of the network are defined using the condition $d = \text{match}(\Gamma_\alpha, \Gamma_\beta) < \epsilon$. ϵ being the cut-off point deciding if there is an edge between gene sequence Γ_α and gene sequence Γ_β. It is hoped that ϵ can be small and results suggest that 0.15 is appropriate to

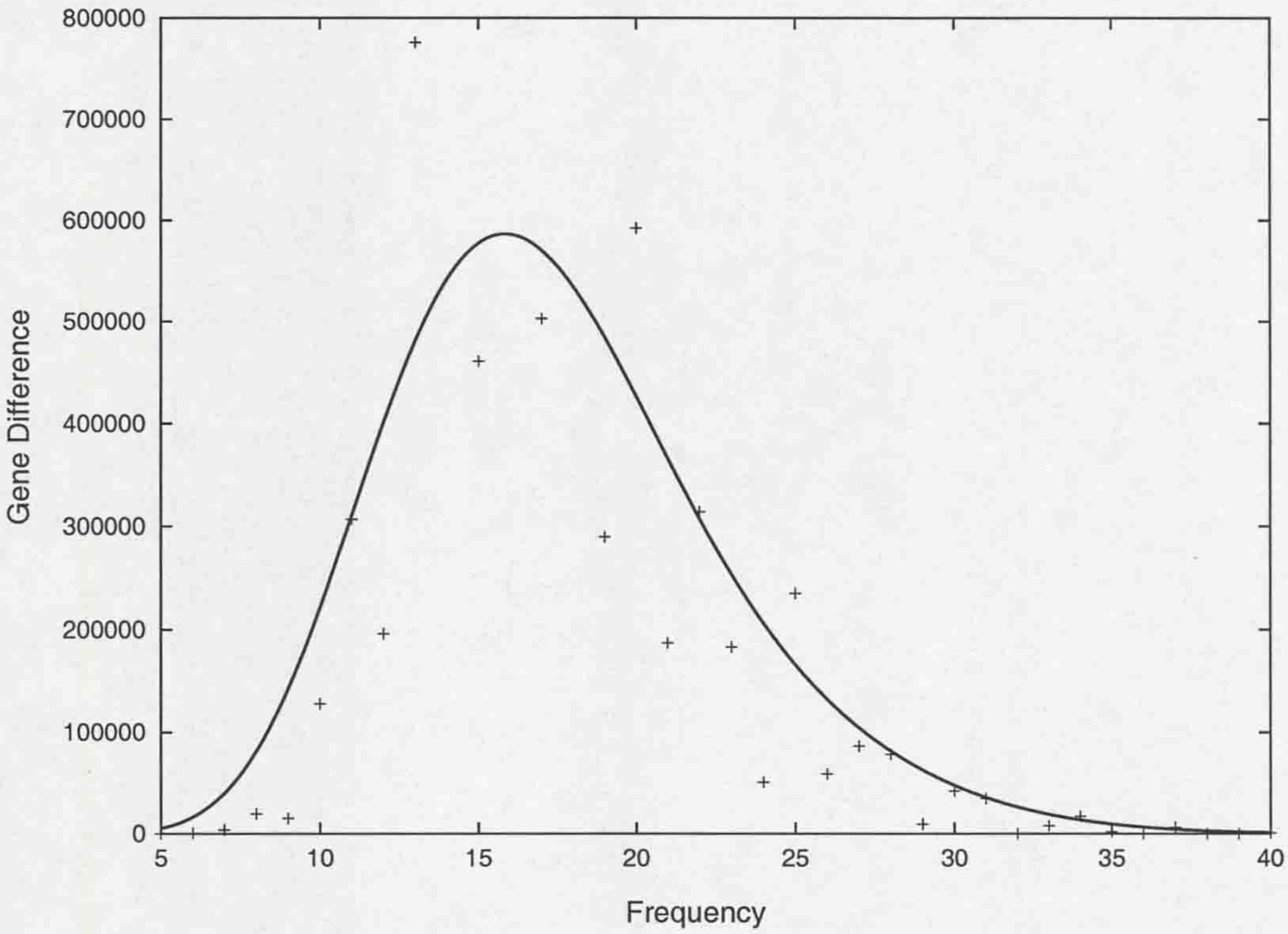


Figure 4.5: Cumulative distribution of all random genes, from a sample of 32 individual genomes.

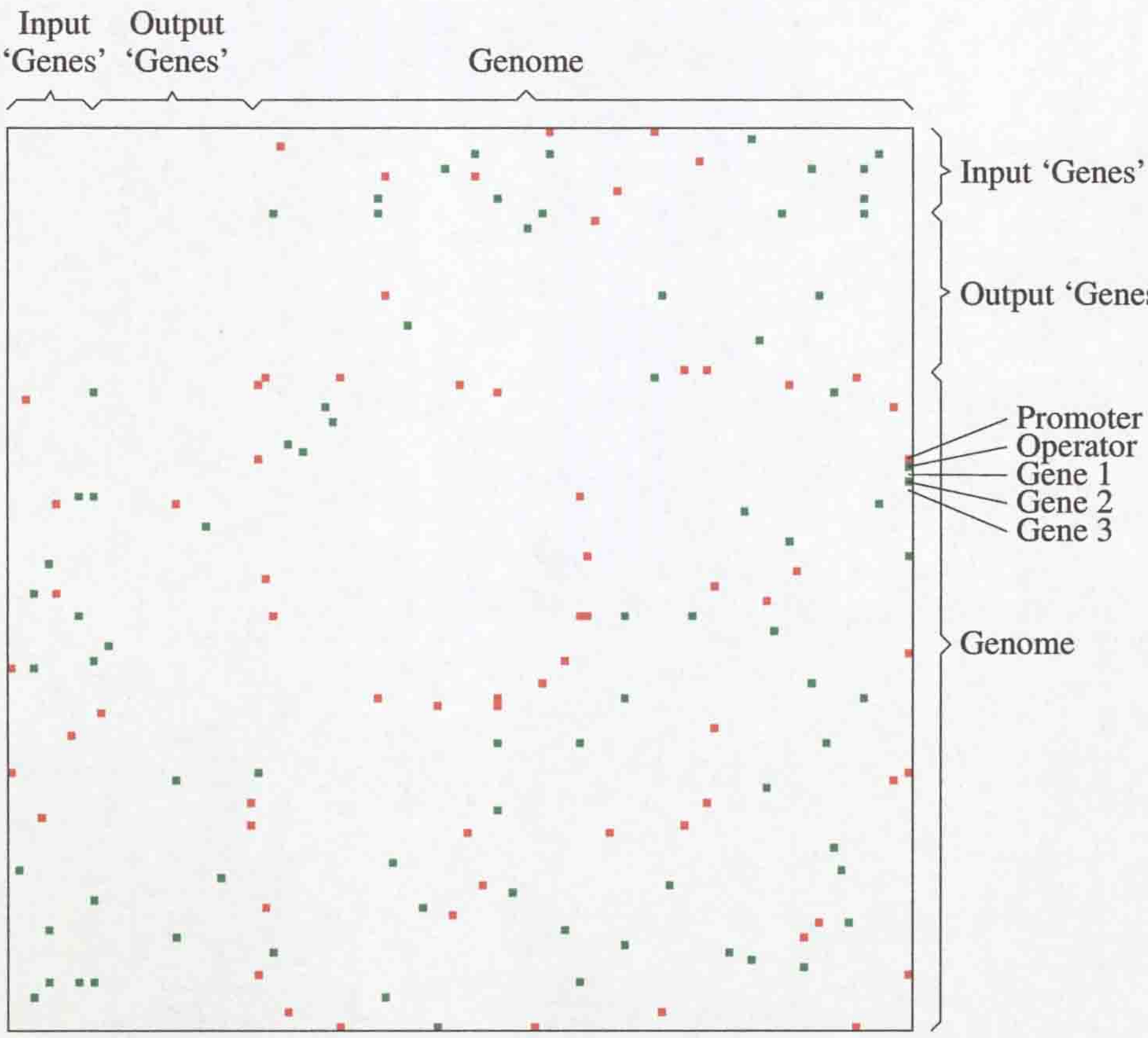


Figure 4.6: Gene interaction within a randomly initialised genome. Both vertical and horizontal axes represent the same input receptors, output receptors and the genome. Squares show there is a link (a relation) between the pairing of genes or receptors. Input and outputs do not interact direcly so the top left shows no interactions occuring.

ensure the network is loosely connected enough to be computable. Obviously, the network will be updated whenever some new sequence is added to the genome but this only requires n^3 steps per genome per iteration and n^4 during genome initialisation rather than a quick initialisation and n^4 steps per genome per iteration. A quantitative figure on the connectivity of the *E.coli* genome is not available, but it is considered to be low.

To enable visual inspection of running simulations, the simulation software can display the gene connection network. An example using random (i.e. non-converged) genes is given in figure 4.6. This shows 10 inputs Φ (first 10

columns), 5 outputs Ω and a genome of 90 genes Γ. The term gene refers to a sequence rather than a gene that leads to a transcription product. As all interactions are reciprocal, this adjacency matrix is symmetric - it is shown in it's full form to allow for debugging of what would otherwise be a large amount of data.

The colours are solely to help readability on what would otherwise be a mass of black squares. Each adjacent square is a different shade ensuring that no two adjacent genes are the same shade, again to be readable. In cases where there is much more gene interaction, and so more filled in squares, the difference would be stark.

A small operon is annotated, possible interaction paths are shown between the operon towards the end and an operon around a third from the start of the genome. Without obtaining more detail, this example shows the operons promotor interacts in some way with a gene at the end of the genome, so we can assume that gene at the end of the genome as a sigma factor. As is the gene toward the start of the genome, on the same horizontal position as the promoter.

4.8.5 Combining reactivity functions

During the simulation run, the above functions (p(t), b(d) (bind enzymes), b(t) (release enzymes) and match($\Gamma_\alpha, \Gamma_\beta$)) need to be combined to give an overall probability of state change. The probabilistic steps are there to add the stochastic effects of existence inside the cytoplasm.

The probabilities potency(t), bind(d) and release(t) are then used throughout the lifetime of the network Ci as the weights in the probabilistic reaction, or more specifically as probabilities for enzyme state changes. Each enzyme instance has two basic states, it is either reacting with some other enzyme (including being attached to the genome) or it is idle. In the latter case, the probability of it reacting is:

$$p^u(\Gamma_\alpha, \Gamma_\beta) = f^u(\text{potency}(t_\alpha), \text{bind}(d), \text{potency}(t_\beta), \text{bind}(d))$$

and the probability of it ceasing to react is:

$$p^r(\Gamma_\alpha, \Gamma_\beta) = f^r(\text{potency}(t_\alpha), \text{release}(t_\alpha), \text{potency}(t_\beta), \text{release}(t_\alpha))$$

The simplest function for both $f^r()$ and $f^i()$ is the product of pairs, and this has been used. Other options are $\min(a, b)$, $\max(a, b)$ and $a + b - ab$ - each calculated for each pair of the gene and then for the combination of the genes.

4.9 Reaction P.D.F.s

The implementation uses the following formulas to calculate both probabilities used for stochastic decision making and in stochastic forcasting of future events, this being an effort to remove the overhead that comes with continuously generating random numbers and testing against a probability function. The simulation supports a variable sized time iteration, this detail is not shown in the functions that follow but is taken into account so that probabilities remain effectively constant. These following sections build on the previous sections in that they state the formula used in each reaction type, whereas the above sections have defined the same formula in terms of other formula.

It should be noted that the choice of the exponential function as used throughout is based on ease of computation rather than any biological reality. This function was chosen because its inverse can easily be found and computed in either direction. The exact function for simulating these reactions in this scenario is unknown, reassuringly what can be seen from biology is that any and all functions are used at some point. As a result, some function had to be used and linear seems inappropriate, exponential was the only choice.

The same should be noted of the reaction parameters. There is no absolute correct parameter set for these reactions, the can never be one in such an abstract scenario. The best solution was to combine the known quantities in a way that was fair and so give all the enzymes the same chance. As all enzymes and cells operate by the same rules there is no strong bias, all reaction parameters are combined without any one dominating.

4.9.1 Input regions, receptors and enzymes

The stochastic unbinding of input region-receptive gene products is calculated using the formula:

$$t_r = \frac{-\log p}{\alpha_r + \gamma_r}$$

Receptive gene products being repressors, anti-repressors, FAPs and/or sigma factors. p is a random variable in the range $(0, 1]$ and α_r, γ_r are rate coefficients from the potency matrix of table 4.2 and γ_r from the unbinding matrix of table 4.8.3. The result t_r is then the time to unbind relative to the gene's time of creation; not the current simulation time. This is to avoid digitising effects. If $t_r <$ current_time then the enzyme will never be bound in the first place. The above equation comes from the integral of the unbinding probability:

$$p = e^{-t_r \gamma_r} e^{-t_g \alpha_r}$$

The stochastic binding of input region-receptive gene products can be calculated using the formula:

$$t_r = \frac{\log(p) + d\beta_r}{-\alpha_r}$$

Where p is a random variable in the range $(0, 1]$ and α_r is from the potency table 4.2 and β_r is from the binding table 4.8.2. This formula isn't currently used, instead the following formula is used to make a probabilistic decision at any given instant.

$$p = e^{-d\beta_r} e^{-t_r \alpha_r}$$

The problem with this approach is the continuous stochastic checks of all possible pairs of enzymes. It would be much better to calculate t_r but that then creates a conflict for when t_r expires it could be the case that it has already expired with another input enzyme. In the implementation there are more options to avoid the continuous testing of probabilities, however it would be difficult to show that randomising artefacts did not perturb the model.

4.9.2 Repressor proteins and operator regions

The stochastic binding of operator region with repressor gene product is decided on the following probability:

$$p = e^{-d\beta_r} e^{-t_r \alpha_r}$$

Where d is the physical distance between operator and gene product, t_r is the age of the repressor in question (operators have no age), α_r is the operator-repressor coefficient in the potency matrix and β_r is the operator-repressor coefficient in the binding matrix.

The stochastic unbinding of operator with repressor is calculated with the following formula:

$$t_r = \frac{-\log(p)}{\gamma_r + \alpha_r}$$

Where t_r is the time the binding will end, relative to the repressor's time of creation. p is a random variable in the range $(0, 1]$, γ_r is the operator-repressor coefficient taken from the unbinding matrix and α_r is the operator-repressor coefficient taken from the potency matrix. This formula is based on the integral of the following:

$$p = e^{-t_r \gamma_r} e^{-t_r \alpha_r}$$

Where t_r is the current age of the repressor.

Note that natural repressors do not willingly break their hold on the DNA, repressor binding is stable. The repressor must be actively removed by an anti-repressor; as a result γ_r is very small to ensure t_r is very long, longer than the cell is reasonably expected to live.

4.9.3 Promoter regions and sigma factors

The stochastic binding of promoter and sigma factors is essentially the same as the previous interaction, note that the presence of RNA polymerase can be reasonably assumed. Only the sigma factor ages, the promoter is timeless. Binding and hence transcription initiation is based on the following probabilistic decision:

$$p = e^{-d\beta_{sf}} e^{-t_{sf} \alpha_{sf}}$$

Where d is the physical distance between gene and sigma factor inside the cytoplasm, t_{sf} is the age of the sigma factor instance, α_{sf} is the promoter/sigma factor coefficient in the potency matrix and β_{sf} is the promoter/sigma factor coefficient in the binding matrix.

Stochastic unbinding is generally not used, a transcription rate defines the number of bases per unit time that can be transcribed. i.e. $|\Gamma| \cdot tr$. Except for the operator and attenuator effects it is guaranteed that the RNA polymerase will reach the end of the operon and so transcribe all genes. Once the sigma factor has bound, the initiation stage has been reached and the sigma factor is almost instantly ready to be reused, taking $|P| \cdot tr$ simulation seconds to return to the unbound set.

The case where the operator has an active repressor serves as the exception. An initiated polymerase will wait for the repressor to be removed but it waits a finite time, this p.d.f. provides that finite time and takes on a similar form to that of the operator/repressor using the formula:

$$t_{sf} = \frac{-\log(p)}{\gamma_{sf} + \alpha_{sf}}$$

Where t_{sf} is the time the binding will end, relative to the time of transcription initiation. p is a random variable in the range $(0, 1]$, γ_{sf} is the promoter/sigma factor coefficient taken from the unbinding matrix and α_{sf} is the promoter/sigma factor coefficient taken from the potency matrix. This formula is based on the integral of:

$$p = e^{-t_{sf}\gamma_{sf}} e^{-t_{sf}\alpha_{sf}}$$

Where t_{sf} is the current age of the sigma factor. Note that $t_{sf} > |P| \cdot tr$ will be taken as $t_{sf} = |P| \cdot tr$ and so early termination before initiation will not occur. Failure during transcription of a single gene is likely to create more problems than diversity. In effect it could be argued that this possibility of failure is similar to the use of attenuation sequences, but this implementation route would be so restrictive as to obscure the case without it and not be strong enough to support the case for it; there does not seem to be any value

in entirely probabilistic termination. There is also the problem of supporting partial strings which are assumed to be exactly based on the given instance gene type, in this model any form of termination occurs on the gene boundaries so partial gene products are never produced.

4.9.4 Output regions and flagella activation proteins

Binding of the output region and flagella activation gene product is a probabilistic decision based on:

$$p = e^{-d\beta_m} e^{-t_m \alpha_m}$$

Where d is the physical distance between protein inside the cytoplasm, t_m is the age of the protein instance, α_m is the output region-FAP coefficient in the potency matrix and β_m is the output region-FAP coefficient in the binding matrix.

Stochastic unbinding takes the same form as the operator/repressor interaction probabilities. The formula is:

$$t_m = \frac{-\log(p)}{\gamma_m + \alpha_m}$$

Where t_m is the time the binding will end, relative to the FAP's time of creation. p is a random variable in the range $(0, 1]$, γ_m is the output/FAP coefficient taken from the unbinding matrix and α_m is the output/FAP coefficient taken from the potency matrix. This formula is based on the integral of:

$$p = e^{-t_m \gamma_m} e^{-t_m \alpha_m}$$

Where t_m is the current age of the FAP instance.

4.9.5 Anti-repressor/repressor interaction

Binding of the anti-repressor with repressors is a probabilistic decision based on:

$$p = e^{-d\beta_r} e^{-t_a \alpha_a} e^{-t_r \alpha_r}$$

Where d is the physical distance between enzymes inside the cytoplasm, t_a is the age of the anti-repressor instance, t_r is the age of the repressor instance, α_a is the anti-repressor/repressor coefficient in the potency matrix and β_r is the anti-repressor/repressor coefficient in the binding matrix. α_r is the same as the α_a coefficient but from the point of view of the repressor.

Stochastic unbinding uses the formula:

$$t_a = \frac{-\log(p) + k(\gamma_r + \alpha_r)}{-(\gamma_a + \alpha_a + \gamma_r + \alpha_r)}$$

Where t_a is the time the binding will end, relative to the anti-repressors' time of creation. p is a random variable in the range $(0, 1]$. γ_r is the repressor/ anti-repressor coefficient taken from the unbinding matrix, α_a is the anti-repressor/ repressor coefficient taken from the potency matrix. α_r and γ_r are the same coefficients but taken from repressor's point of view. k is the age difference between the anti-repressor and the repressor. Should t_a indicate a bind time less than the current time, the binding never occurs. This formula is based on the integral of:

$$p = e^{-t_a \gamma_a} e^{-t_r \alpha_r} e^{-t_a \alpha_a}$$

Where t_a is the current age of the repressor and t_r is the current age of the anti-repressor.

4.9.6 Attenuator regions and repressor interactions

Binding of the attenuator region and repressor gene product is a probabilistic decision based on:

$$p = e^{-d\beta_r} e^{-t_r \alpha_r}$$

Where d is the physical distance between enzyme inside the cytoplasm, t_r is the age of the repressor instance, α_r is the attenuator region/repressor gene product coefficient in the potency matrix and β_r is the attenuator region/repressor gene product coefficient in the binding matrix.

Note that an attenuator repressor is known as a enzyme, this makes it unambiguous in the table but confusing if used anywhere else.

Stochastic unbinding takes the same form as the output/FAP interaction probabilities. The formula is:

$$t_r = \frac{-\log(p)}{\gamma_r + \alpha_r}$$

Where t_r is the time the binding will end, relative to the repressor's time of creation. p is a random variable in the range $(0, 1]$, γ_r is the attenuator/repressor coefficient taken from the unbinding matrix and α_r is the attenuator/repressor coefficient taken from the potency matrix. This formula is based on the integral of:

$$p = e^{-t_r \gamma_r} e^{-t_r \alpha_r}$$

Where t_r is the current age of the repressor.

4.9.7 Terminator regions and RNA polymerase

Transcription terminators have no other effect than stopping the RNA polymerase from continuing with transcription. There are no probabilities with this as the natural terminator mechanism is taken to be reliable. Natural termination is either *rho* dependent or *rho* independent, termination in the model is always rho independent. Rho dependence could be simulated using another gene product type or by some external level parameter. Both of these option were discounted, the later because it is far too artificial and the former because terminators would then be another form of attenuator and the distinction between the two would be too small for terminators to be worth considering as a separate type. Therefore, attenuators can be considered as essentially rho dependent terminators but with biased termination probability.

4.9.8 Individual enzyme ageing

When all enzymes are created each enzyme is assigned a time to live based on the p.d.f.:

$$p = e^{-t\lambda}$$

Where λ is the coefficient taken from the potency table 4.2 and t is the time to live relative to the current time. Since an enzyme has multiple types λ is calculated as the mean of the type coefficients. As with the others, this is integrated to form:

$$t = -\log(p)/\lambda$$

giving the correct relative value for a random variable p in the range $(0, 1]$. As with the above integrated forms, this avoids the need for stochastic checking of enzyme removal events.

The figures in the potency matrix give an average half life of 99 seconds, this is slightly shorter than can be found in *E.coli* but not much shorter. As mentioned in [TMBW97], half lives vary considerably and regulatory enzymes have the shortest half lives of 2-30 minutes. For COSMIC a short half life is necessary simply to speed up the response of input to output and reduce unnecessary computation.

4.10 Genome Mechanics - Run time interactions

This section brings together ideas from the last few sections to create the main simulation algorithm. As has been noted before, there are many scales to this simulation, this scale is that of the genome to the transcription engine. Later we will move onto the interface to the cell wall that represents another scale.

During a typical simulation step, the algorithm goes through the phases of enzyme binding, timed event handling and removal of defunct enzymes. Enzyme binding being initiation of any of the enzyme interactions mentioned at length previously. Nothing happens during binding so the only states COSMIC considers are binding and unbinding, unbinding being implemented as a timed event that has a known time to activate, thus allowing unbinding to be implemented without polling a set of all bound enzyme pairs. A Nassi-Schneiderman based diagram of this is shown in figure 4.7, the fine level detail

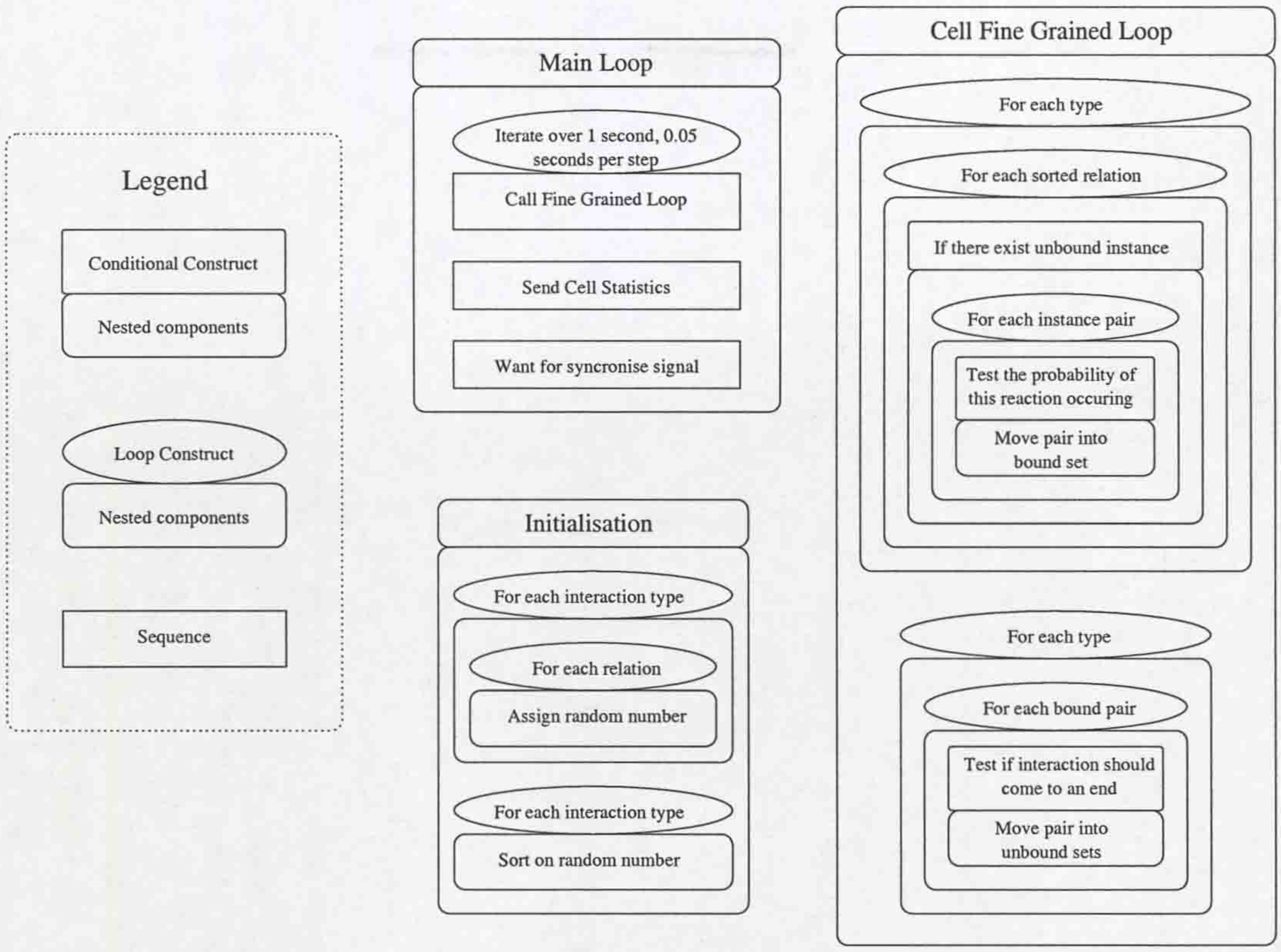


Figure 4.7: Nassi-Schneiderman based diagrams depicting the overall structure handling each of the gene and enzyme process, namely binding and unbinding enzymes and genes, with high level sychronisation between cells.

is then discussed in the following sections. The main point to note here is that all relations dealing with interaction between enzymes and genes are sorted on a random number to remove biases introduced by the order of execution.

4.10.1 Enzyme binding

The first phase is binding, this is carried out on a per type basis for each of the possible interaction paths (as defined by the interaction network set Ci). All Gn lists have a random ordering so as not to bias the interaction, this is fixed for the life time of the cell.

Given a promoter P and sigma factor SF pair, the number of available enzymes are tested as zero sigma factors makes the reaction impossible. For each pair of unbound (available) enzymes, the combined binding probability is

calculated using the above reaction matrices and then tested against a random variable. Should a binding occur then that pair of enzymes are moved from the two Sus (both of which are in Ω in S) to SFP (also in S). Apart from SFP recording a bound pair Γ_α and Γ_β, the Su and Sr types are distinct because most interaction paths deal with unbound enzymes, so the wasted effort of searching through lists of bound enzymes is avoided. At the point of binding the time at which the sigma factor and promoter site will part is calculated as $|P| \cdot tr$ simulation seconds, where tr is the transcription rate in bases per second - suitably adjusted to take account of the COSMICs shorter than natural gene length. Once this time has passed the sigma factor will unbind from the genome and return to Su and the sigma factor entry Γ_β in Sr is set to $\emptyset$. The position of the sigma factor instance ρ_{SF} is set to the promoter position ρ_P. RNA polymerase doesn't explicitly exist in the model but it is assumed to exist, this will start transcribing genes; the timing generally determined by the gene length and in this case is based on calculating with $|P| \cdot tr$ and later $|\Gamma| \cdot tr$ where Γ is an operator, attenuator, terminator, a simple gene or another promoter. In order to put a cost on every interaction, $|\Gamma| \cdot te$ is subtracted from the cell's energy ε in C_i during binding. This being an attempt to put a price on every interaction.

Following P on the genome G_i of C_i, there could be an operator site O. If the operator O exists and if it can be found in ORe then the operator has an attached repressor and so pauses transcription - assuming there is an RNA polymerase on the adjacent promoter to pause. When paused by a repressor the RNA polymerase stays on the promoter site and either unbinds through timing out via stochastic unbinding or continues transcription after the repressor has been removed by an anti-repressor. When the RNA polymerase has moved from the promoter site, the promoter is moved from Sr in S and placed back into Γ^S to await another transcription initiation, Γ_α in Sr is set to $\emptyset$ to indicate that the promoter instance has moved back into Γ^S. This results in Sr having no references to enzyme instances, it remains as a state variable showing the position ρ on the genome and so therefore the current position of the RNA

polymerase during transcription.

The case of the operator and repressor is less involved as there is no movement of the mobile RNA polymerase, once the repressor has finished it simply detaches, Sr is removed. The overall approach is the same as the promoter/sigma factor interaction, the possible interaction network ORe is scanned and for each pair of enzymes O and Re, the combined probability is calculated and then compared to a random variable. If this pair is chosen then the operator O and the repressor Re enzymes are moved from Su to Sr. These two enzymes are represented by two Γ^S entries of type Su, binding brings them together in a single Sr. The interaction time is calculated using the reverse probability function given in 4.9 and this event then waits for that time to occur - a time that is quite long to account for the correct behaviour of anti-repressors (inducers). In the meantime the given operator site is blocked from other repressors and the adjacent promoter site is forced to pause any polymerase trying to transcribe genes. In the same way as for promoters, an operator which is already in use stops another repressor from binding until the current repressor is removed when its probabilistically determined time is reached. One effect of the binding is to reduce the cell's energy according to the function $tr \cdot |O|$, this being an attempt to put cost on all interactions.

The repressor unbinding time is long because the biological case shows that repressor attachment to the genome is stable and so only an anti-repressor enzyme can remove the repressor. This is therefore another interaction path and unlike the other paths, both the list of bound and unbound repressors is read through to enable binding events with repressors that are currently repressing. Compatible reaction paths are read from the interaction network ARe in Ci, individual enzymes in the cytoplasm are read from both Γ^S (the unbound list) and ReO (the bound repressors list). Successful comparison of the binding function given in 4.9 with a random variable leads to the enzymes being moved from which ever state they were in and bound together in an Sr, which is then placed in ReA in S. Unbinding time is calculated using the unbinding p.d.f. function given in 4.9 together with a random variable. When

unbinding occurs, the two enzyme instances are moved from Sr to individual Su entries in Γ^S, to allow rebinding and the repressor position ρ_{Re} is set equal to the operator position ρ_O.

The attenuator interaction path is unusual in that it is potentially destructive. When binding unbound attenuators, the list of potential interactions is read from ARe in Ci as usual and enzyme instances of these types are located in Γ^S in S. Each possible pair from Γ^S (Su) is given the chance for binding using the p.d.f. function in section 4.9. A bound pair is then moved from the Γ^S unbound list into Sr and the unbinding time calculated from a random variable based on the inverse p.d.f. unbinding function. This is the same process as for the operator/repressor interaction. The destructive aspect comes from the RNA polymerase that will come across the attenuator site. When a RNA polymerase reaches the attenuator site it is stochastically stopped or continues transcription after the attenuator, in either case the repressor bound to the attenuator is removed and destroyed. The attenuator in Sr is put back in Γ^S and the repressor instance in Sr is removed altogether. In the case of normal timed unbinding, both instances are moved from Sr and placed in Γ^S, ready for another chance to rebind. The attenuator reaction does not affect the cell energy level, but should the repressor survive, the repressor location ρ_{Re} is set to the attenuator location ρ_A.

Interactions involving the input receptors (modelled as static genes Γ) involve the same steps as those for the operator. The interaction network $\Phi\Gamma$ is stepped through in a random but fixed order for each iteration of the simulation, each possible enzyme pair that fits the type specification in Γ^S and Φ^S has their mutual probability determined and is then tested against a random variable. If the pair is chosen then the input instance Su from Φ^S in S and gene instance Su from Γ^S is moved to a common Sr in $\Gamma\Phi$. The reaction time for this pair is calculated using an inverse p.d.f. and random variable. When unbinding occurs, the enzymes in Sr are put into their respective unbound lists and the position of the enzyme ρ_Γ which bound to the receptor is set equal to receptor $\rho\Phi$. The inputs represent the environments input to the system and

so need to reflect this input. This is achieved by combining the normal binding probability with a probability value that represents environmental input. Details and justification are in the environment section 4.11.

The output is the only interaction that leads to movement of the cell and so positive matter level increase for the cell as a whole. As with the others, the interaction network ΩF is cycled through in a random but fixed order. A compatible pair of idle enzymes in the form of the output receptor Ω and the FAP F are found using the combined probability of their characteristics, this probability being tested against a random variable. A positive outcome moves the two enzymes from their respective lists $\Omega \in S$ and $F \in S$ into the mutual list $F\Omega$. When unbinding occurs, the enzyme in Sr are put into their respective unbound lists and the position of the receptive enzyme ρ_Γ is set equal to ρ_Ω. The reaction time is also calculated so that the binding can be stopped, as with the others this time is stochastic, based on the inverse of the combined probability functions.

The sixth interaction path is that of the repressors and anti-repressors. Again the interaction network RAn is cycled through in a random but fixed order and enzyme pairs repressors Re and anti-repressors An are tested using a combined probability function against a random variable. Should the comparison succeed, both enzymes are moved from their sets and placed together in AnR which is a record of when the reaction will end and the two enzymes will no longer be bound. As both enzymes involved in this reaction poses a time and therefore decay, the end time of the reaction is based on $\min(t_{An}, t_{Re})$. In other words, it is possible for a selected pair to not actually interact at all, the moment they are identified they can be dropped because one of the pair is too old. When unbinding does occur, both enzymes are positioned at a point half way between both of them, that is repressor enzyme ρ_{An} and anti-repressor ρ_{Re} is set equal to $\frac{1}{2}(\rho_{An} + \rho_{Re})$.

4.10.2 Timed events - interaction state changes

The above section lists all the possible interaction paths that are initiated by the simulation, assuming an initial state of no interacting enzymes. This section reiterates what was lead onto in the binding section by covering all instances when an interaction ends or changes state into another interacting pair of enzymes. All timed events are sorted in earliest first order, allowing them to be checked with the smallest amount of effort; each event set from S (that is $\Gamma\Phi, F\Omega, AnR, SFP, ReO, ReA$) is checked in a round robin order until no events remain to be processed (ie. Their unbinding action is carried out), for this current time frame. This is a fairer scheme to that used above, the round robin approach assures that no one interaction type (such as the promoter-ribosome interaction) dominates by having the first chance to try all of its possibilities. It is assumed that the low probability of binding ensures no binding artefacts are present, so despite not being strictly fair this scheme does not show in the results.

The sole multi-state event is the sigma factor-promoter state, the sigma factor is placed back into the cytoplasm soon after transcription initiation and is replaced by an RNA polymerase. As the RNA polymerase can move, its state is defined by the gene type it is currently over. Each timed event signals the move from one gene to the next. This new event is placed in the time sorted event queue $SFP \in S$; this event is made up of the null sigma factor $\emptyset$ and the gene Γ_1 that followed the promoter P; the promoter P and sigma factor SF is placed back into $\Gamma \in S$ for the next usage. The new time of this event is taken from the first gene Γ_1 and is simply $tr \cdot |\Gamma_1|$ - for efficiency the operator state is assumed to have occurred, it is essentially taken to be part of the promoter's state.

When a post promoter event in SFP times out, the gene Γ_i is transcribed and a new enzyme based on the template Γ_i is created and placed in the unbound queue of $\Gamma \in S$. The next event time is calculated as $tr \cdot |\Gamma_{i+1}|$, and Γ_{i+1} is placed in SFP instead of Γ_i. If Γ_i is a terminator then the Sr entry is removed from $SFP \in S$. If Γ_i is an attenuator and that attenuator is in ReA

then transcription is stochastically terminated in the ratio 1:10 (as per the *trp* operon example).

When an event in ReO times out, the repression operation has done its job of blocking a promoter and so has no further work to do. The repressor Re and operator O are placed back in $\Gamma \in S$ and the ReO entry removed. This contradicts the biological case and so the bound time is very long.

When an event in AnR times out, the interaction has completed its task of blocking a repressor from binding to an operator. The repressor Re and the anti-repressor An are placed back in Γ, and the event in AnR is removed.

When an event in $\Gamma\Phi$ times out, the interaction has completed its task of both blocking enzyme Γ - by using it elsewhere. Regardless, as with the other simple interaction types the input Φ is placed back into $\Phi \in S$, the gene Γ is placed back into $\Gamma \in S$ and the event in $\Gamma\Phi$ is removed. In the future the number of inputs is expected to be used as a source of inputting external information into the transcription simulation, the number having a negative effect on the enzymes available for other uses - such as transcription, but also for repression, anti-repression and acting as a FAP. When an event in $F\Omega$ times out, the effect in terms of set movement is identical to that of the input.

4.10.3 Iteration final steps

The above two sections cover most of the time spent by the algorithm, binding enzymes and unbinding enzymes with occasional enzyme creation. All enzymes have a time to live that starts when each is first created, when an enzyme reaches the end of its life it is removed from the cytoplasm represented by $\Gamma^S \in S$. The time of death is determined stochastically at the time of birth and though the exact time can occur while an enzyme is bound to another, the enzyme is only removed when it is not bound. The exact time is determined by the enzyme type, which is an average of the applicable types in the potency matrix 4.2. As inputs and outputs do not age and only unbound enzymes are moved, only the set $\Gamma^S inS$ can contain enzymes to be removed.

Once a single iteration is complete, the cardinality of each set in S is

recorded. Also recorded is the combined (bound and unbound) cardinality of each transcription product from G that are found across all of S and the bound inputs/outputs from Φ and Ω found in the sets $\Gamma\Phi$ and $M\Omega$.

The final step is to increment the simulation time internal to the next value. Tests have revealed that a value of $\Delta t = 0.02$ is a small enough for all the above events to be discrete. Depending on the probability coefficients in the reaction tables, this figure could well be larger without granularity artefacts.

4.11 Environment

4.11.1 Introduction

The simulation as defined so far consists of a collection of cells, each cell C having an individual state and dynamically defined state changes. In order to evolve the cells towards some goal, an environment is necessary to provide open ended evolution. The environment can never be completely open ended (Kreft, personal comm.); evolution towards something must surely mean there is some global optimum being sought, even though that optimum is implicit and defined by the interaction between the environment and the cells themselves. The work of Kreft *et al.* represents a good basis for both open ended evolution and further extension to the simulation framework. Kreft *et al.* [KBW98] describe an environment in which *E.coli* grow on a glucose enriched medium; the main purpose of the work is to demonstrate cell growth patterns formed by the growing population. The growth patterns largely depend on accurately modelling the growth rate of individual cells as they deplete their localised glucose source. For use in our simulation, the important points are i) this is an environment in which to build on (as already stated) and ii) the modelling is quantitative and based on a small set of differential equations and parameters, making them sufficiently accurate but also highly computable.

The environment as modelled by Kreft *et al.* [KBW98] is used as an arena in which COSMIC simulated *E.coli* cells must evolve to co-ordinate their flagella in response to the local environment under each cell. The implicit fitness

function is therefore to seek out glucose rich areas either by path following or through risk taking.

The environment therefore creates a longer time scale in which multiple cells exist in a shared space. Cells are created and die, in between they interact and try to live as best as their genomes allow. Each genome is computationally intensive but relatively isolated over the short term and so genome simulation is implemented as a distributed system. The cells are periodically considered as a real population in order to pass on combined environmental effects. This synchronisation time is t^r in $\mathbf{E}$, comparing this to the per cell fine grained time of Δt shows synchronisation time to be around two orders of magnitude greater, hence synchronisation with the environment and peers is a long term effect that allows for the genome to respond to its environment.

4.11.2 Cell input

The cell interaction comes about through the depletion of environmental energy E in $\mathbf{E}$ during the environmental synchronisation phase. All cells are rewarded by growth proportional to glucose level under each cell. This depends on the condition of the environment immediately under the cell and the current mass of the cell, mass leading to diameter by the formula:

$$2\Pi(\sqrt[3]{\mu * 0.75/\Pi}\,)^2$$

The diameter can then be shown in the visualisations and be used as a size attribute with which to decrease the substrate concentration in the environment. When the cell is given a reward, the reward comes from the cells current position in the environment and so the same location will give less reward in the next cell synchronisation phase, obviously all parameters are automatically adjusted to take account of changes in the synchronisation window size and the resolution of the glucose matrix E. Over the long term the environment recovers at a rate set by global parameter me^r in P. A static cell will soon deplete all the locally available resources; thus static cells are selected against in favour of moving cells, and randomly moving or fixed trajectory cells are

selected against in favour of intelligently moving cells.

Qualitative information and accurate parameters for glucose were taken from accurate biology literature as cited by Kreft *et al.* [KBW98], these being [BD96, TN84, KW82, NTT96]. The main effect to be modelled is the link between cell growth and glucose concentration, this can be approximated with the following relationship:

$$v = \frac{x \cdot V_{\max} \cdot S}{K_m + S}$$

This is a Monad [KW82] based equation and takes the values [KW82] $V_{\max} = 0.342ms^{-1}$ (maximum growth velocity) and $K_m = 2.3$mg glucose (half-saturating constant). S is the glucose concentration relative to the individual and starts at 4.5 mg glucose. x is the current cell volume in grams, converted ([SLD79]) from cell mass in litres. v is the volume increase in g s^{-1}. Glucose use is then found from the average efficiency ([NTT96]) of 0.245 g cell volume per g glucose. In the future this may instead use an equation based on [KBW98] which accounts for the non-linearity of glucose take-up efficiency, this effect is currently considered too small to have any impact on COSMIC.

v and $v/0.245$ can then be scaled according to t^r and the environment updated along with the cell, the cell also uses energy for non-growth which is lumped into the maintenance term q_m.

4.11.3 Cell output

Cell growth comes from the environment and can be considered an *input* to the cell; cell *output* makes use the each cells set of flagella. The activation probability for each input receptor in $\Phi \in \Gamma\Phi + \Phi^S$ is set according to the glucose level under each receptor, this is a more direct form of input to the cell but has no effect on the well being of the cell. The position and orientation of each cell combined with the position of each receptor on the cell wall gives a position in the glucose matrix E, the level at that position is then directly used as the value for a in each Φ. Note that in order to enforce some phenotype regularity, positions of each receptor Φ and of each flagella Ω are identical on all cells.

During the normal simulation of the cell, receptive proteins will bind with the inputs Φ, the actual number bound is not important as COSMIC makes a departure from reality and instead makes use of those proteins which haven't bound. In effect, the inputs form a sink and those proteins that remain free are available to continue as before; to put it another way, the input is inverted. However the genome responds, the output will be in the form of bound flagella activation proteins. The average number of binds to Ω_o over the time t^r indicates the activation level of the each flagella. The cell position is changed by using the activation level of each flagella, cell movement is the vector sum of each of the activation levels multiplied by that flagella's position vector. This total movement vector is then reoriented according to the cell's orientation and then added to the cell's current position. Orientation of the cell is not changed, the movement vector is sufficient. The movement this achieves is not biologically realistic, the reality of bacterial chemotaxis is mechanically stochastic reorientation and swarming of flagella into one identically rotating tail. An accurate model of this is outside the scope of the simulation, environmental feedback is essential to include a form of fitness measure but considered a secondary part of the model.

The maximum speed of travel in a liquid solution ([Mac96]) is around $25\mu ms^{-1}$, this amounts to around 10 body length per second. COSMIC considers the environment to be an agar plate to allow for 2D resources in the implementation and visualisation, as a result the maximum speed is somewhat lower and is set by parameter c^r. Note that energy for chemotaxis is assumed to exist, simulating the exact biochemistry is beyond the scope of COSMIC.

4.11.4 Cell death

Once the environment has updated its substrate map, all cells are informed of their new cell mass and position and are then free to continue with the next $t^r/\Delta t$ iterations. The algorithm cycles around again; looping until $0 = |\Gamma^S| \in S$, $\mu \leq 0$ or $\epsilon \leq 0$, at this point it is known for certain that the feedback mechanism has failed at the scale of the genome ($0 = |\Gamma^S|$) or the genomes

interaction with the environment (μ and ϵ) has failed. As a result the entire cell is removed from the simulation, whatever nutrients it contained is not released into the environment.

4.11.5 An example environment

COSMIC provides a view of the population as in Figure 4.8, this shows the presences of cells through their effect on the environment. This is a typical early example with 20 initial random genomes, the darker the area the lower the glucose level. A time series of pictures like this show the cells moving. Here the different sizes show relate to the differences in cell mass, it must be noted that this figure is artificial and the difference has been exagerated. In most cases the non-converged genomes all have the same effect on the environment, the cell is motionless and eventually runs out enzymes or mass falls below a critical level. In either case, the simulation removes the cell from the environment leaving the black circle of substrate use. Chapter 7 gives a series of pictures taken directly from COSMIC, unlike Figure 4.8 these show the cells with realistic dimentions and the state of the environment.

4.12 Cell Division

In comparison to the cell itself, the simulation of cell division is a simple process that aims to fairly distribute the contents of the parent cell between the two daughter cells. The most technically challenging aspect is the division of the gene products. Gene products exist as many instances of each gene and at division each daughter cell should have a fair share of the same types of gene products. For COSMIC, this is achieved by considering each individual gene product in turn and metaphorically flipping a coin to decide which parent that gene product will transfer to. Given many trials of cell division, the result follows the Normal distribution with a mean of half the gene products. Thus the process is generally fair, and rarely harsh enough to kill the most brittle cell.

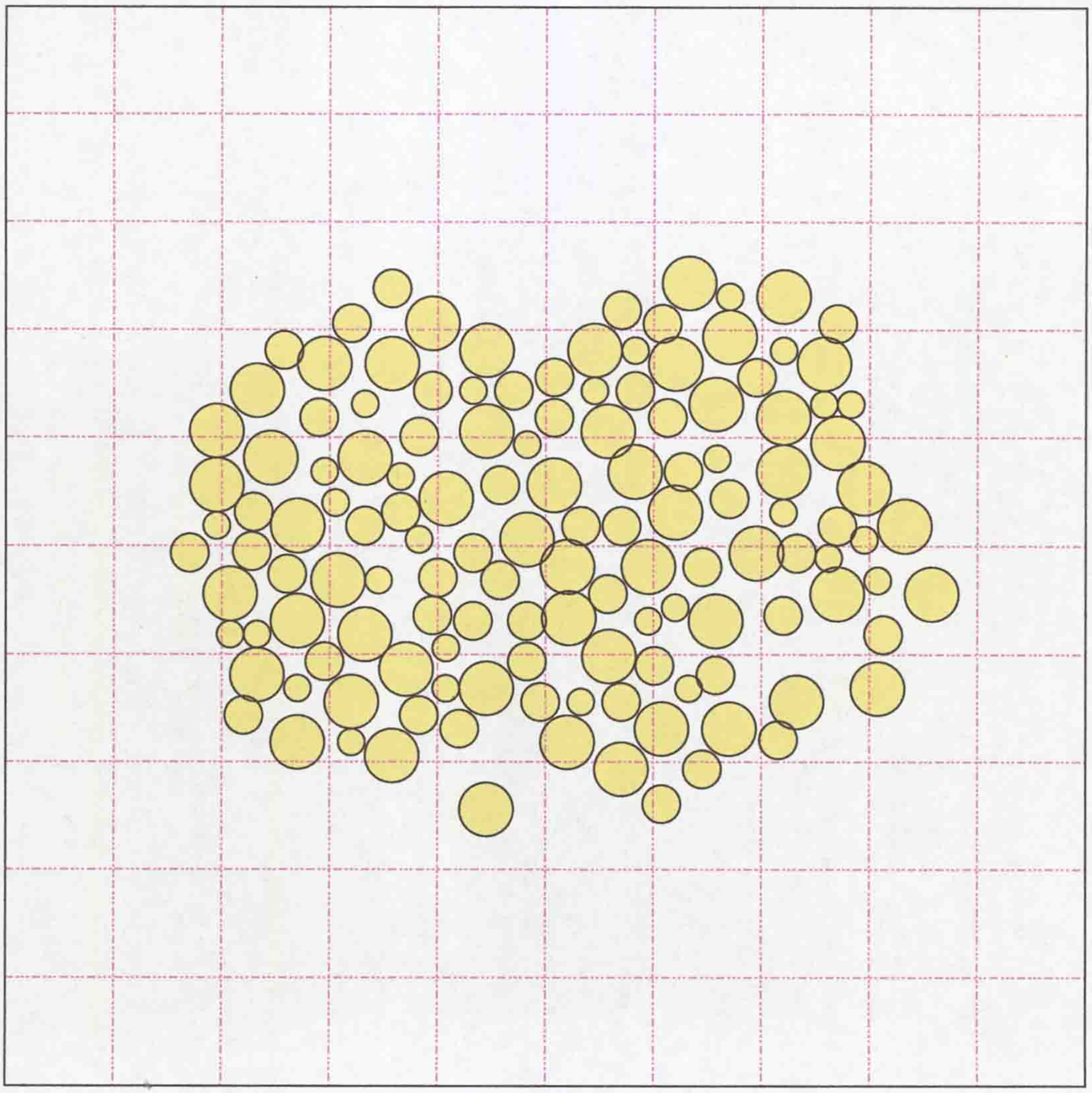


Figure 4.8: Example of population distribution with underlay of nutrient availability. Differing sizes relate directly to cell volume and so relative success, as the bigger the cell, the closer to the onset of division and the creation of a genetically identical cell.

For example, given a single gene type with 10 gene products of the same type, each of those products has a 50:50 chance of being with one or the other daughter. The probability of all 10 gene products going to the same cell is then very low at 0.2%, which then means cell death through division should be a rare event. Most gene types have many more gene products and so we would expect only the most brittle cells to suffer death from division. This is not a bad result in itself as we are seeking robust transcription networks.

The final change to the daughter cells is a halving of mass. This is deterministic and set to exactly half that of the parent cell. Nothing could be gained from making this step noisy.

In every other respect, the two daughter cells are identical in content. Both have the same genome and both have the same input receptors and output responses. In the environment the daughter cells are moved a small distance away from the parent, so that they do not compete for resources and to be visible in the environment snapshots.

4.13 Individual Initialisation

Each individual cell in the population contains a genome and derived gene products, COSMIC initialises each genome uniquely and so the population is effectively multiple species but sharing the same physiology. If a cell is successful then its genome will be passed on to more offspring and so create populations of a single closely related species. The first stage is important, as artificially populating the simulation with related cells first means identifying good genomes. As has been said before, the definition of good genome has never been clear and so it was always better to have the simulation decide. The results later show that a good-enough solution will take over the population, and so choosing the good genome is effectively a race to be the first good enough genome.

The input and the output set of proteins are shared by all cells and so are created first. All enzyme templates share the same gene length distribution,

it is a uniform distribution over the range 10-15 that gives each gene's λ, this figure is loosely proportional to the expected size of the network, though changing the binding tolerance ϵ would also account for small changes. The strings that make up Φ and Ω are mutually exclusive ($\Phi \cap \Omega = \emptyset$) but are otherwise random.

$|\Phi|$ is set so as to give a reasonable number of receptors given the genome size and enzyme population size. As the number of simulation enzymes is much smaller than the real number, so is the number of receptors. A figure of 50 is currently used, this is quite arbitrary but is obviously many orders of magnitude away from the real figure.

$|\Omega|$ follows the average number of flagella. In *E.coli* this figure is in the range 0-15, the simulation uses the average of 8 that was found on wild strain normals [Mac96].

The lengths of the genes (Γ) are set using the same uniform distribution as the length of the input/output sets. Any difference is obviously penalised by the matching function and so a difference in lengths of input/output sets requires that the gene lengths be distributed inside that length range. The initial content of the genes is again randomly distributed over the set L.

The size of the genome (in units of genes) is a complicated issue with no clear answer. Genome size is intended to be flexible enough to allow COSMIC to converge on a reasonable number, factors that come into play are the genome duplication time, and energy usage for duplication and transcription. As the enzyme interactions work by detaining metabolites, the genome size must also take into account input and output size, as these are necessarily fixed. A reasonably starting point would seem to be the square of the minimum cardinality of the sets Φ and Ω. The main point here is to identify the relationship between the figures so they don't become yet more tuning parameters, this point could be said for all the ill defined parameters. Regardless, a uniform distribution of $70 - 120$ seems a reasonable starting point and was not changed for all the simulation runs.

4.14 Genome Mutation

None of the text so far has described genome length change or any other form of mutation. COSMIC implements one form of mutation, sequence deletion and insertion. Both sequence insertion and deletion modify the length of the genome G, and update any reactions that may have been involved on the chosen strand of genome. Insertion is based on a uniform distribution of genome length marking the start point, the same distribution marking the end point, and the same distribution marking the insertion point. Deletions use the same distribution to find start and end points on the genome G. That section is then immediately removed and complications resolved based on what was bound with that segment of genome. The solution to this is type specific but ultimately means preserving the enzymes involved with the genome. These large mutations could well be deleterious but considering how reasonable solutions dominate the population (as shown in later chapters) the loss of cells to these large charges seems balanced when compared to the possibility of increasing the population fitness at a faster rate. Note the original goal of COSMIC was the simulation of evolution with all the varied forms of evolution, yet this version of COSMIC includes this single operator. This was due to time constraints, the above framework took so long to implement that a broad set of mutation operators had to be left for future work. With that in mind, the above model has scope for these operators.

4.15 Summary

This chapter has described a formal mathematical model *E.coli* that includes the scales of genetics and a multicellular population of individual cells. The model is described using a combination of set theory, relations and probability density functions, realising an entirely individual-based model. This model then forms an open-ended framework onto which mutation operators can be applied.

Using supporting evidence from chapter 2 we state that optional transcrip-

tion is the most important aspect of *E.coli* when considering the modelling of evolution and so COSMIC should be based on this mechanism, with supporting mechanisms where necessary.

The model is described using a representation based on a hierarchy of sets, at the highest level there is the environment set containing the cells, the environment and high level parameters. The cell is then an ordered set, each containing a genome set, a gene product set and a multiple sets storing interaction relationships between elements of the gene product set and the genome set. Relations between elements of sets is then determined by functions whose inputs are parameters of the individual elements. An optional transcription mechanism for producing gene products (considered by COSMIC to immediately become enzymes) from genes then completes the self supporting cycle of enzyme creation, allowing the cell to be considered alive if the cycle continues.

The cells are placed in an environment in which they compete with each other on a continuous basis for available substrate - used directly by cells for growth, but importantly the substrate is reduced by a cells presence. Cells divide when they reach a critical size, in doing so pass on their individual genome and half their gene products. This in total provides an implicit fitness function by which cells compete against each other, a cell must be motile to remain in high substrate areas and this motility depends on the genome. The more controlled the motility the higher average substrate and so the quicker the growth, leading to the quicker reproduction of the genome that brought about quicker cell division in the first place. Note the term quicker, as cell division speed is relative.

The genome of each cell is subject to mutation on a continuous basis, sequence insertion and deletion over the whole genome is possible with a small probability. This is the source of evolution by which cells are to evolve better motility.

Finally we note that the COSMIC model presented in this form is computationally feasible despite the large number of component parts. This is largely the result of careful consideration during the implementation of the set struc-

tures and the model being executed in parallel due to the vast differences in time scales being modelled.

Chapter 5

Parallelisation

5.1 Introduction

Large simulations of bacterial colonies require vast amounts of computational time. The only way to achieve the necessary level of performance is with parallel computers and a suitably designed implementation that maps the problem onto the hardware. For real problems this mapping can be non-trivial requiring careful consideration of the constraints in both the system being modelled and the hardware that executes the model.

This chapter describes the parallel implementation of COSMIC and shows that it is possible to map a dynamic problem such as this onto fixed resources, for the most part by making use of implicit multiplexing of resources and showing the importance of knowing where to partition the problem between server and clients. Through this an efficient simulation has been created, making maximal use of the available hardware without constraining the model to require excessively specific resources.

An individual based modelling approach has large populations of enzymes multiplied by the large populations of cells that combine to give a system of some 9 million individual entities[1]. This simulation needs to run on a possibly hetrogeneous system, involving machines of different speeds and varying

[1]Summed over approximately 280 cells as found in a recent simulation.

communication latencies.

To make the system more extensible and realistic the genome and the proteins exists inside a cell wall, demarcating what belongs to that cell and so making a clear cell boundary. This has further advantages in terms of extensibility, following the same concepts of increasing scale allows the inclusion of other concepts later on, the cell becoming a container for all things related to the cell. This helped enforce an ideal boundary of cells and environment, making the division at a clearly defined boundary in the model improves the structure of the overall implementation.

The computational architecture takes these factors into account by following the client server model combined with coarse grained synchronisation of processes. Each process varies in execution time but there are vastly more processes than processors. The environment containing the cells is considered the server and cells inside the environment are the clients, or individual processes. This ensures efficient parallelisation as cell intercommunication is rare in the current COSMIC revision and the environment has minimal processing needs, the overall result being a linear growth in computational nodes allowing a linear growth in the product of simulation speed and total simultaneous number of cells.

5.2 Simulation System

The simulation was first compiled using Gnu GCC 3.0.4, compiled for Linux 2.2.21 running on a 640 megabyte Pentium3 666 Mhz. No Linux specific instructions were used and so the simulation will run on any GCC platform (i.e. almost any architecture in existence) or any C++ compiler once compiler idiosyncrasies have been dealt with.

Memory requirements vary dramatically. As the simulation is essentially a particle simulation, each particle being an enzyme, the connectivity of the network defines the memory usage and simulation speed. A massively connected network will use around 320K bytes per bacterial cell, and so in a population

of many this amounts to a lot of memory and time. Fortunately the networks convergence will bring with it an exponential speed increase and exponential decrease in memory usage.

The entire COSMIC system has been implemented using the Parallel Virtual Machine system [GBDJMS94], allowing the distribution of computation to other machines. In theory this allows COSMIC to can scale up to run hundreds or thousands of cells, this being a more realistic figure for dependable results. The practice of this remains to be seen, PVM has interface limitations which constrain performance and C++ can make inter-process control and communication very verbose. The alternative parallelising system, MPI has problems of its own and was rejected early in development.

Execution currently consists of merely running the application which represents the environment, this application then creates cells as separate UNIX processes via the PVM system. Execution continues up to a (currently) fixed global time though the possibility of restarting a stopped simulation has been implemented. Saving object state is an excellent idea but C++ does not do this automatically.

5.3 The Process Tree

The division between server and client is obvious as cells don't directly interfere with each other, the cells can then be clients and have relative autonomy for the fine grained process of computing the contents of the cell and its response to the environment. With the current COSMIC implementation intercell communication only occurs when cells divide, this is a rare event compared to the fine grained computation of enzymes. This strategy also limits data duplication by clearly separating which object has access to data and which object needs only an average, time scale difference being the important metric. Message passing is also kept to a minimum because synchronisation across all cells is coarse grained, the client-server synchronisation time scale is much longer than the internal cell time scale used for enzyme interactions.

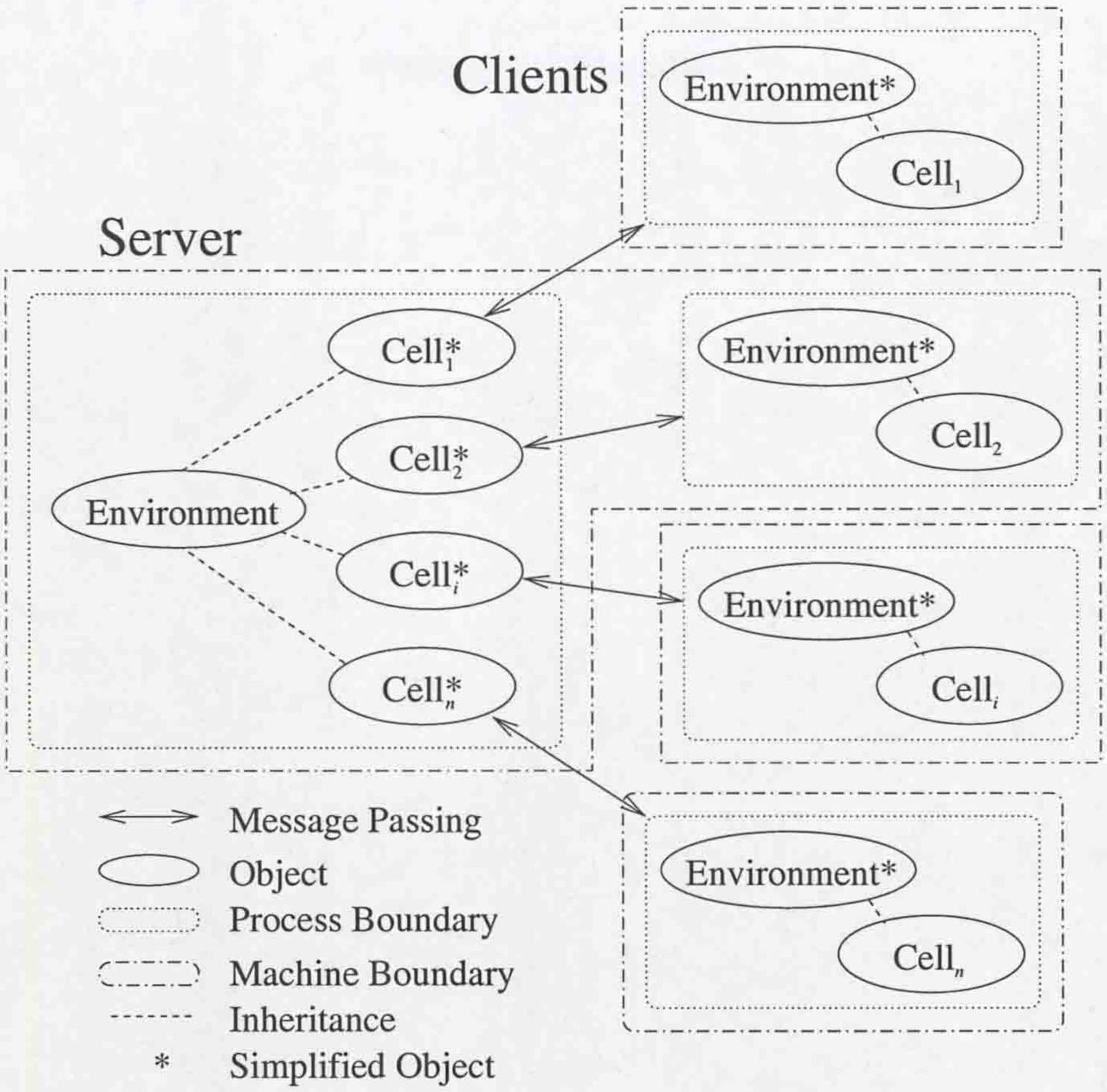


Figure 5.1: Process synchronisation

Figure 5.1 shows the logical structure and the physical mapping onto machines. The environment and all the cells are individual UNIX processes, the environment process is started and then this (via PVM[1]) starts the client cells. It is a limitation of PVM (i.e. hidden load balancing) that each cell is a process even if residing on the same physical machine. Considering the time scales, the environment and cells can share the same machine as the environment's processor usage is always low when compared to the sum of cell processes.

As occasional statistics and values are passed between server and clients, the main processes all have lightweight objects (Figure 5.1 marked with a *) to store these miscellaneous values without resorting to duplicate but cut down objects.

[1]Parallel Virtual Machine is a message passing library for parallelising applications, supporting data sharing and remote execution primitives.

COSMIC requires strict synchronisation to ensure no cell runs through more iterations than any other cell. As a result there is a potential for wasted computation time as all cells must pause at some point and come to a common mark point while slower cells complete that same time frame.

The following steps help avoid this by keeping the synchronisation time to an absolute minimum.

1. The environment process waits for each cell in the population to return a result, the environment process iterates steps 2-3 until all cells have reported in their results.

2. The environment process receives data values indicating what the cell did in the environment during the fine grained execution of the enzyme processing. Also passed are more general statistics about internal cell activity.

3. Values indicating how far the cell has moved and new glucose concentrations from around the cell are then passed to the cell so that it can respond to the new environmental conditions.

4. With all cells waiting on the environment process, the environment process then makes a life or death decision based on the received per cell statistics. Each cell can either carry on living or be told to die (i.e. remove itself from the system) or divide into two cells. A message is sent to each cell individually, if the message allows the cell to continue living then that cell continues executing by running another set of fine grained enzymatic iterations.

5. COSMIC has provision to periodically save the entire state of the simulation, as the environment decides when this is to happen each cell has to be informed. As saving of state is predictable, this message can be pre-empted to shorten the idle time of cell processes that would otherwise still be waiting on the environment process.

6. Using per cell statistics and cell chemotaxis rate obtained from the cell process, a specific location in the environment is distorted to show the glucose use and general presence of the cell. Figure 5.2 is an example of this.

7. The environment then properly modifies it's glucose map, as the cells are now already running autonomously this comparatively lengthy operation has time to complete before the cycle starts again with cells returning their next round of results.

This execution strategy allows for a hetrogeneous execution environment, automatic load balancing tending to place more cells on comparatively faster machines. The ideal load balancing does however have an unknown component, the execution time of a cell varies over time and so ideally process migration is also required for 100% efficiency.

This tight synchronisation comes at a cost, all cells will at some point in time come to a halt and wait for a very short duration for the environment process to respond. Considering network latencies this is actually very low because a 100 Mbit network can send a 10 byte message to 500 cells in around 77 msec. Although this a long time in computational terms there is little waste when 500 cells are spread across much fewer machines, the message to continue executing will be heard by some cells well ahead of the majority. These cells then have a few milliseconds to run before having to share the processor with the many more cells that will start running again in this 77 msec period. This combined with the long resynchronisation window of several seconds makes the actual overhead very small.

The main problem lies in general reliability. Each cell is required to return its result data before the environment can proceed, so cells on slow machines or slow networks will slow down the whole simulation. In the average case, where all machines are used equally, this is not an issue, but experience has shown that processor or network load caused by other jobs varies across machine and also varies with time over many days. Even worse, the loss of a machine will

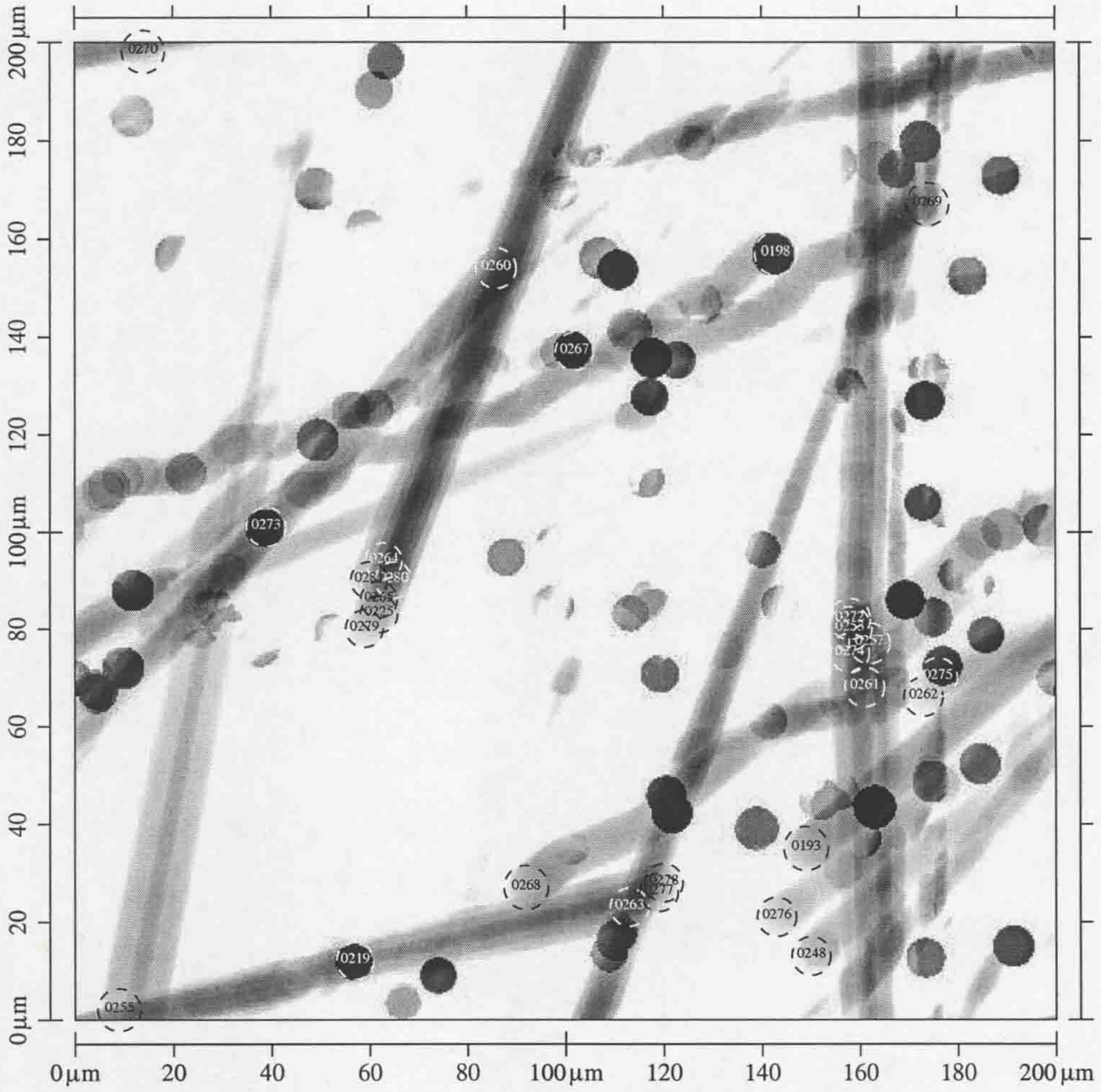


Figure 5.2: Environmental change after 212 minutes.

kill all the cells that occupied that machine and force COSMIC to hang forever waiting for the results that will never arrive. It is for this reason that COSMIC supports state saving, if a problem terminates the simulation it can simply be restarted from a point in the near past with no loss of information. This avoids total simulation failure but the dynamics of the network are a much more pervasive problem.

Ideally this is solved with more dynamic load balancing. The role of load balancing in the present implementation is limited to making a best guess to which machine a new cell should be created and spend its lifetime. As a result, external changes in machine or network load cannot be accounted for and the simulation is then held back. Parallelism is implemented using PVM, which is a message passing interface with an at-execution load balancer. This provides a good basic framework but is limited when executing in a real world dynamic environment, leading to the non-optimal distribution of computation. For clusters of PCs the solution to this is process migration based on automatic dynamic load balancing algorithms such as provided by the MOSIX patches available for Linux or dynamic load balancing provided with some commercial UNIX machines. Migration of processes between processors should improve load balancing to a near optimal point because processes are no longer fixed to where they started but can instead be exported to another co-operating processor. The only limitation is the time and resources required to stop, move and restart a process.

Process migration is expected to work for uneven processor load, network load balancing is however a much more difficult problem. The transient nature of network traffic and the effort in identifying bottlenecks make this an optimisation problem in itself. Fortunately the course grained synchronisation in COSMIC avoids the worst effects.

5.4 Cell Division

The most involved process of a running COSMIC system is the division of a cell once it has matched the division criteria. Typical communication patterns involve only the environment and the cells, all of which are contacted individually, cell division is different in that three parties are involved and cells must synchronise with each other.

Typically a running cell will receive a `continue` message from the environment after sending its updated attributes. These attributes include the cell mass, based on this mass the environment decides (solely on a threshold level of 0.4fl) that the cell is large enough to divide, it instead sends a `divide` message, this is figure 5.3, edge 1. This signal will eventually lead to the cell running through another iteration, first it will go through the necessary steps to entirely divide itself in half:

1. The cell contents are marked by a 50:50 probability. These contents include all the currently transcribed gene products, all active reactions and any interactions between the genome and the gene products - such as sigma factors. The genome is not halved but identically duplicated.

2. Using the marking, outbound gene products and all their interactions are then saved to file store, figure 5.3, edge 2. This small group of around 15 files completely describe the properties of the new cell. Aside from the individual molecules, cell mass is halved and an adjacent environmental position is computed. In the parent (donor?) cell the same marked components are removed and the cell mass is halved.

3. The environment (always acting as overseer) initiates a new cell process via PVM (figure 5.3, edge 5) and instructs the daughter cell to await a signal from the parent (figure 5.3, edge 3), this signal ensures that the file data is complete before full cell initialisation.

4. Once informed by the parent, the daughter then reads the file store (figure 5.3, edge 4), initialises its data structures and runs as part of the

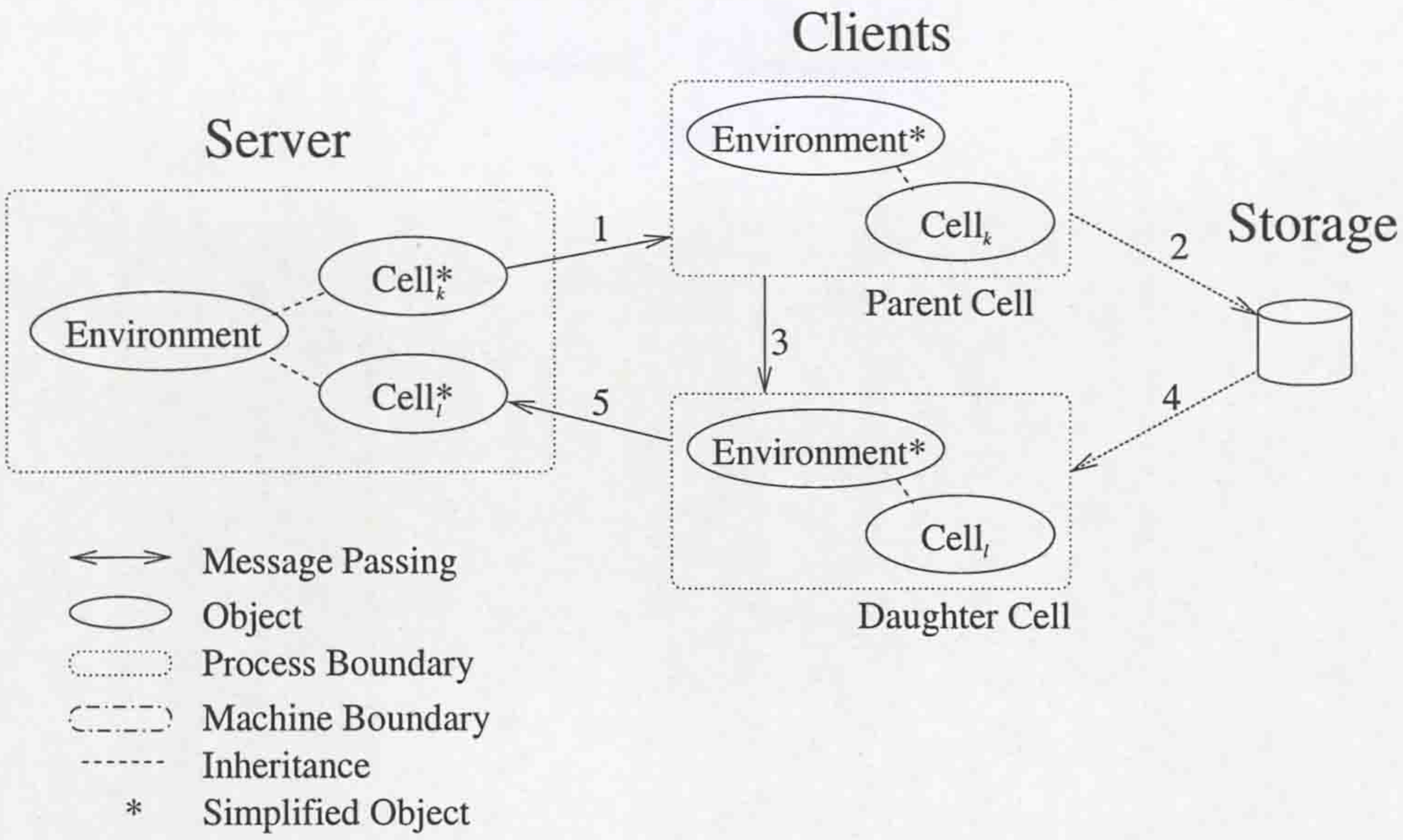


Figure 5.3: Process synchronisation at cell division

whole population. The halved parent completes an iteration as usual and the daughter is left to complete its first iteration.

This whole process is much more involved in terms of recording cell state rather than inter-cell communication per se. Communication is limited to processes synchronisation required through the use of file store. It is a limitation of C++ that code reuse is limited when saving objects, the lack of serialisation capabilities forces custom procedures for each object and each form of output stream.

The biggest challenge here was to ensure the division scheme fitted efficiently with the normal synchronisation of all cells with the environment. To this end cell division causes little bottleneck when used with a high population, running the cell process and reading the files takes much less time than an iteration of the fine grained internal cellular simulation.

5.5 Dynamic Effects

The previous sections gave the parallel algorithms as currently used by COSMIC. These algorithms seemed reasonable in terms of efficiency of execution and robustness to adverse conditions. In running the simulation it became clear there were three stages of development in the execution patterns. This cast a new light on what appeared to be good solutions, the practice of investigating how load could better be balanced and then implementing a solution all took time to be developed as the significance of dynamic effects became apparent.

The execution of the simulation can then be regarded as three phases. The first is a search phase where the simulation maintains a minimum of 20 cells and regularly needs to add new cells as the current population fails to achieve the basic evolutionary aims. At this point the computational system is very lightly loaded with at most one cell per processor. The simulation then runs at many iterations per second and finds at least one valid cell by around 200 simulation minutes, 2 hours and 8 minutes minutes of wall clock time. This marks the beginning of the next phase, that of growth.

During the growth stage the cell population rapidly expands at a rate of 3 cells every 5 simulated minutes. There are quickly many more cells than processors and so the load balancer becomes very important in maintaining cluster efficiency.

The final stage is then saturation, there are so many cells that random placement of cells during load balancing would have no effect. The system moves extremely slowly, using all computational resources regardless of `nice`[2] setting and taking 30+ seconds per iteration. At this time any load balancing with other processes is useless, when running other processes are starved, when not running there is the realisation that the pausing for 50% of the time run twice as slowly, and unimpeded is already too slow. This stage develops rather

[2]UNIX prioritises running processes according to their level of niceness. A value of 0 is the default and requests that as much CPU time as possible be given to this process. A value of 19 requests that only spare CPU time is used by that process.

than being arrived at, but once present the cluster can then be regarded as over loaded and that number of cells is too many. Fortunately this only occurred on the student access cluster, and shortly afterwards a private cluster of much higher specification was obtained. Although the problem must surely still be present, when using this new cluster it has not been witnessed.

5.6 Non-private Access Clusters

Despite the benefits of near optimal load balancing, practical situations can also bring a requirement for load limiting. When used on a cluster not dedicated for use by COSMIC but in fact used by users for real-time day to day tasks, COSMIC must then take second place. Initial attempts obviously made use of system priorities to put COSMIC processes at the end of the process schedulers queue. Surprisingly this doesn't actually work, if the number of low priority processes outnumber the normal priority processes the user will still feel the difference, giving each low priority process 0.1% of system time has significant impact when there are 1000 low priority processes on the same processor. The end result was insufficient load control, the real time uses were still slowed down by a large margin whenever the simulation grew beyond the initial search phase.

To address these problems a second solution attempted to implement a course grained load balancer where by all processes were paused for x minutes and then left to run for y minutes. This was implemented by using the UNIX signals SIGSTOP(pause) and SIGCONT(continue). However, it was considered too intensive an operation to signal all processes (including gzip and ssh), or even the environment and the cells. Exploiting the requirement of strict synchronisation only the environment process need be paused, all other processes will then pause when they return their results. This minimised overhead and ensured that the minority of cells which take up most of the time could have chance to finish their iteration without being unduly paused.

In effect then, this gave more time to the longer running cells while stop-

ping the mass of quickly executing cells that defeated the `nice` flag given to the scheduler and seemed to be an ideal solution; except that COSMIC is modelling evolution and so the scenario changed. The cells which take a long time to execute tend to be the most successful, and so over the course of days their number grows to the point where this scheme becomes ineffective [3]. Ideally this scheme could have been more fine grained and signalled the cells directly, the added network overhead is small compared to the need for greater control. It would have required a daemon process to send the signals, as the cells themselves only respond to PVM messages between iterations but this could well have been worth the end result.

The previous method focused on time slices based on the current time of day, it could be said this only worked for the middle phase of execution where something could be gained by other users. The static table that converted time of day to execution/pause ratios was held back when in the first phase of simulation and irrelevant to the latter stage. A more dynamic control was needed and for this it seemed appropriate to compute total computation time across all cells and then use this figure to ensure less than 100

Except that it in practice it did not work. The jiffies computation is always distorted by the strong synchronisation requirement and by other user processes delaying a `niced` cell. Total jiffies was always much lower in an active simulation than expected, worse still the figure bore little resemblance to cluster wide system load. During the early phase jiffie total was 50% of maximum, during the second stage it was also 50% of maximum.

Ultimately all these schemes failed to work through either lack of suitable control mechanisms, or at least control mechanisms that scale up, or by an interplay of strong interprocess synchronisation and the random demands of other system users. The arrival of an effectively dedicated cluster made all this a moot point. The take home message is that load balancing is much harder than it seems, the information to make decisions on is always out of date, the

[3] With hindsight, this technique was only ineffective because the demand for computational power greatly exceeded supply. It would take many minutes for a cell to complete its iteration and so pausing the environment process had no effect.

course of action based on the information takes time to take effect, and in many scenarios of high load the mechanisms offered by the OS have no effect anyway.

5.7 Supporting Scale

In using a cluster the restrictions and shear awkwardness come to bare, single machines have a large set of tools supporting process monitoring and control, with a few clicks or key presses a process can be killed or stopped, associated open files located or just determine memory usage. The introduction of a cluster then suddenly makes these tasks much more difficult, although available for each machine the lack of unification adds another level of complexity.

In attempting to deal with this problem, a set of scripts were created that ran common tools on all nodes of the cluster and then collated the results. At this time the primary cluster was the Computer Science Linux farm that served students and staff, computational capabilities varied between machines and over the day and over the year. To make matters worse, some machines are taken out of service without notice.

The most general tool that developed over time was `tstcluster.sh`, this script passes a given set of bash commands onto all machines in turn, first checking if that machine was up and accepting connections in a reasonable time. For the check the script opened a timed `ssh` connection to a machine, if the connection succeeded within 10 seconds the machine was considered up and unloaded, if not then the machine is considered down or otherwise uncommunicative regardless of what `ping` would indicate.

Using this script some useful functions can be implemented:

```
./tstcluster.sh ps -A -o pcpu "|" sort -gr "|" head -n 1
```

will find the most active process on each machine, indicating if a machine is available but not used by COSMIC. Requesting just the load average is

misleading as it does not take page swapping into account. A machine low on memory should be avoided just as much as a machine with many users.

The line:

```
./tstcluster.sh "find /proc/ -maxdepth 1 -user greg \
   -printf \"find %p/fd/ -not -type d\n\" | sh 2>/dev/null | wc"
```

counts the open file descriptors for each machine in the cluster. Compiled into Linux is a limit of 1024, above this value files cannot be opened and processes cannot be started.

Another useful tool is `pvmtouch.sh` which updates on all machines the time stamp for all PVM files owned by `greg` (this author), thereby avoiding `tmpwatch`[4] deleting temporary files that are still in use by long running simulations. The temporary file in question is a regular file containing the process ID of the PVM daemon and the path to a UNIX socket by which the PVM console based client communicates with the daemon. Without the file the client assumes there is no daemon and so starts another daemon, effectively loosing contact with the COSMIC simulation that continues to execute.

Other useful commands developed out of necessity are `pvmadd.sh` which adds a working set of machines to the PVM machine list. More machines can be added over time, but not deleted. Unfortunately PVM has no mechanism that allows a machine to be removed from the available list but not delete any running PVM processes on that machine. `pvmadd.sh` is edited by hand using the results of the above availability check.

`netlog.sh` logs network load between the server machine (which also acts as a client) and the majority of clients. It was expected this would show moments of network saturation or some pattern of usage. In practice neither occurred, the processes exchange little data and what data there is is surprisingly stochastic, removing any pattern.

`mkdirs.sh` makes a new set of result directories, based on a given initial path. Normally `current/` and then post simulation is renamed to the format:

[4] **tmpwatch** is a system utility that is run periodically to remove temporary files that appear unused.

`run[Year][Month][Day]/` Which is used for indexing and archiving past simulations. The results are partitioned into type to reduce the number of files per directory, although generally unlimited `ext2`(the standard Linux filesystem) incurs a performance penalty for every file in a directory, on the order of $O(n^2)$. This imposes a practical limit of 10,000 files, fewer if read performance is paramount. Partitioning into directories avoids this limit for all simulations so far executed. Further partitioning is done on the environmental state pictures as 10,000 pictures amounts to 100,000 iterations, or over 27 hours of simulated time. In this case case the data is partitioned into sets of 1000 files - the last four digits of time, recording every tenth.

During development some cell processes crashed and although all forms of process IO were recorded on an individual cell basis, finding that cell amongst the hundreds or thousands of other cells was a tedious task. `scanssytems.sh` solved this problem by first asking PVM for a list of running cells and then using that list to confirm the process actually existed on the given machine. The lack of confirmation pointed out the failed process and so the failed cell number, the number then leading a logged `stderr` of the cell and to clues of failure.

All of these small utilities help to greatly reduce the development time, by removing errors from otherwise tedious tasks or by finding what would otherwise be a needle in a haystack of machines. It is only when one works on a cluster that the problems of organisation and communication become apparent. Nothing is in the one place, that requires a mental paradigm shift and the above selection of programs to make the transition easier.

5.8 Other Limitations

Linux uses the concept of UNIX file descriptors to refer to files and link processes via pipes. There is a limit of 1024 file descriptors compiled into the Computer Science Linux farm and although this might seem enough when for example 400 cells runs across 20 machines, there are more factors to take into

account. Firstly, cell population per machine can vary anywhere from 0 to the some future maximum population size and although these extremes are obviously unlikely it does increase the chances of reaching the file descriptor limit if an extreme is approached. Secondly, each cell process starts 10 other processes (gzip and ssh) to compress and return results. Each of these processes uses at least 3 file descriptors. The numbers then speak for themselves, 400 cells across 20 machines implies around 20 cells per machine, each machine needing 660 file descriptors. This assumes even distribution, which won't occur and so the limit become probabilistic at around 27 cells per machine, assuming negligible use by other users. To find if this limit is being approached:

```
./tstcluster.sh "find /proc/ -maxdepth 1 -user greg -printf \
   \"find %p/fd/ -not -type d\\\n\" | sh 2>/dev/null | wc -l"
```

Asks each machine how many `greg` owned file descriptors are used, and so how close to the limit the simulation is. It would have been quicker to check `/proc/sys/fs/file-nr` but the correlation between these figures and the manually obtained figure was not clear.

5.9 Saving State

With the recognition that COSMIC requires considerable amounts of computational time and the unreliable state of resources available at the time, check-pointing was deemed vital. This is the saving of the entire simulation state to file store in such a way that the entire simulation can be restarted from this state image. Strong parallels exist with the cell division process, as a newly created simulation is no different from a daughter cell from a parent, but for the 50:50 enzyme division. The saving itself was then all but the same, the real challenge came from synchronising all the processes to save at the same time while not troubling the client processes with unnecessary synchronisation messages between iterations, namely to not save state. Client cells do not know when this is as only the environment decides if state should be

saved. The solution was pre-emptive messages that took into account the overall client-server synchronisation and never forced client cells to synchronise a second time for each iteration, the message was passed immediately after the main synchronisation message. When state was to be saved, the message was delayed to better fit with division logic but as state saving is a time consuming process this extra small delay is immaterial.

5.10 Storing Data

The shear amount of data generated by COSMIC is enough to fill available storage within very little time. Of the data sets generated some have high information content when measured per kilobyte whereas others have very little information content. All the information could be useful and knowing what is important should always be something for later examination. Clearly too much resolution is possible, but what resolution is required can only be guessed at. There are several problems here, getting the data from the client cell to the possible remote disk, issues of compression capability and choice of date format.

Since resolution is not known, some output streams use the maximum resolution of one sample per fine grained iteration. This then records all information possible but is largely redundant. The solution to this was multiple stages of compression that reduced the output to 1.6% of its original size. This is partly attributed to the use of ASCII text string and mainly attributed to the high redundancy of the data.

Later the COSMIC system also started to use a more ideal format in which tags representing events and parameters concerned were recorded in temporal order. This retained the resolution but was more compact as only changes are recorded and not system wide state[5]. Tags and associated data are all output as ASCII text strings and so are compressible to a ratio of 10

Using either format, once compressed the data stream must be moved to

[5]This is in the sense of a set of all possible measures in a given domain, typically the individual enzyme totals for each expressible gene.

main host system for information extraction. In the ideal case where all machines in a cluster share the same file store via some networked file system, this simple involves witting out the stream to a file name. Setting this up requires system privileges that were not available when using the shared access cluster[6]. For this `ssh` was used to open a connection back the main file store host (which wasn't under the same administrative control as the shared access cluster) and write the data stream.

All this compression and transport was achieved through running external processes in a pipe line, with lines such as:

```
gzip | gzip | ssh babbage cat ''>'' datafile
```

where `babbage` is the file store machine and the file name `datafile` was based on the type of data stream and the cell number. Starting external processes and attaching to `printf()` style formated output functions is very simple in a UNIX environment, the `popen()` system call runs the given string as a command to be executed and associates a standard buffered file descriptor with this executed command, attaching say `fprintf()` to the `stdin` of (for example) `gzip`.

5.11 Modifying Parameters

Simulation control is done via parameter files which record all the global variables and p.d.f.s. During simulation execution they are read periodically, the exact period coming from each file's place in the program hierarchy. Monitoring is done by log files and log pictures. Viewing the output is a batch process, a suite of conversion programs read the event logs and convert them to pictures and graphs as found in Section 5.12 and much more fully in Chapters 7 and 6.

[6] A loosely coupled set of machine made available for students to use as X servers and generally access their UNIX accounts via remote access.

5.12 Parallel Efficiency

The execution environment of COSMIC has changed over the years of its development, in the beginning COSMIC ran on a single machine with a view to running of the farm of loosely coupled machines. The experiences in the last few sections have been largely came from using the farm. COSMIC currently runs on a Grid-enabled cluster of 12 node dual processor Athlon XP 2000+ machines. This leads to a rapid simulation but is still slower than real time, on the order of 7:1. In the space of 9 days COSMIC had evaluated 2132 bacterial cells (this is the sum of cells created with random genomes and cells resulting from the cell division process), with 298 cells still living at the point the simulation ended. The final environment had turned into a bacterially challenging patchwork of nutrients, average final genomes were in the range 42 to 1023 genes long (185 mean), with 10 to 107542 enzymes per cell (13473 mean). CPU utilisation varied in the range 1-100% creating around 3 gigabytes of data per day.

The output from the simulation clearly has two distant scales, the environment state and the state of each cell. The architecture ultimately allows a never-ending simulation in which state can be recorded and reloaded while changing the global and local parameters. From the recorded data a variety of visualisations can be constructed, the common feature is the abstract nature of the labels in that real proteins are not simulated and so data can be isomorphic but not easily comprehensible. These being per cell gene expression charts and network graphs representing interacting genes. Also generated are per cell graphed averages of major parameters and snapshots of the environment at the population level as well as charts showing the lineage of all cells. All these chart types will be described in Chapter 6 and again in Chapter 7.

The labelling system of cells is based on an ever increasing unique identifier that a cell obtains upon creation as either a daughter or a newly initialised cell. When a cell divides the parent retains the original identifier and the daughter receives the next unused identifier. At that instant both cells are the same in terms of genetics but differ in the share of enzymes as each enzyme has a

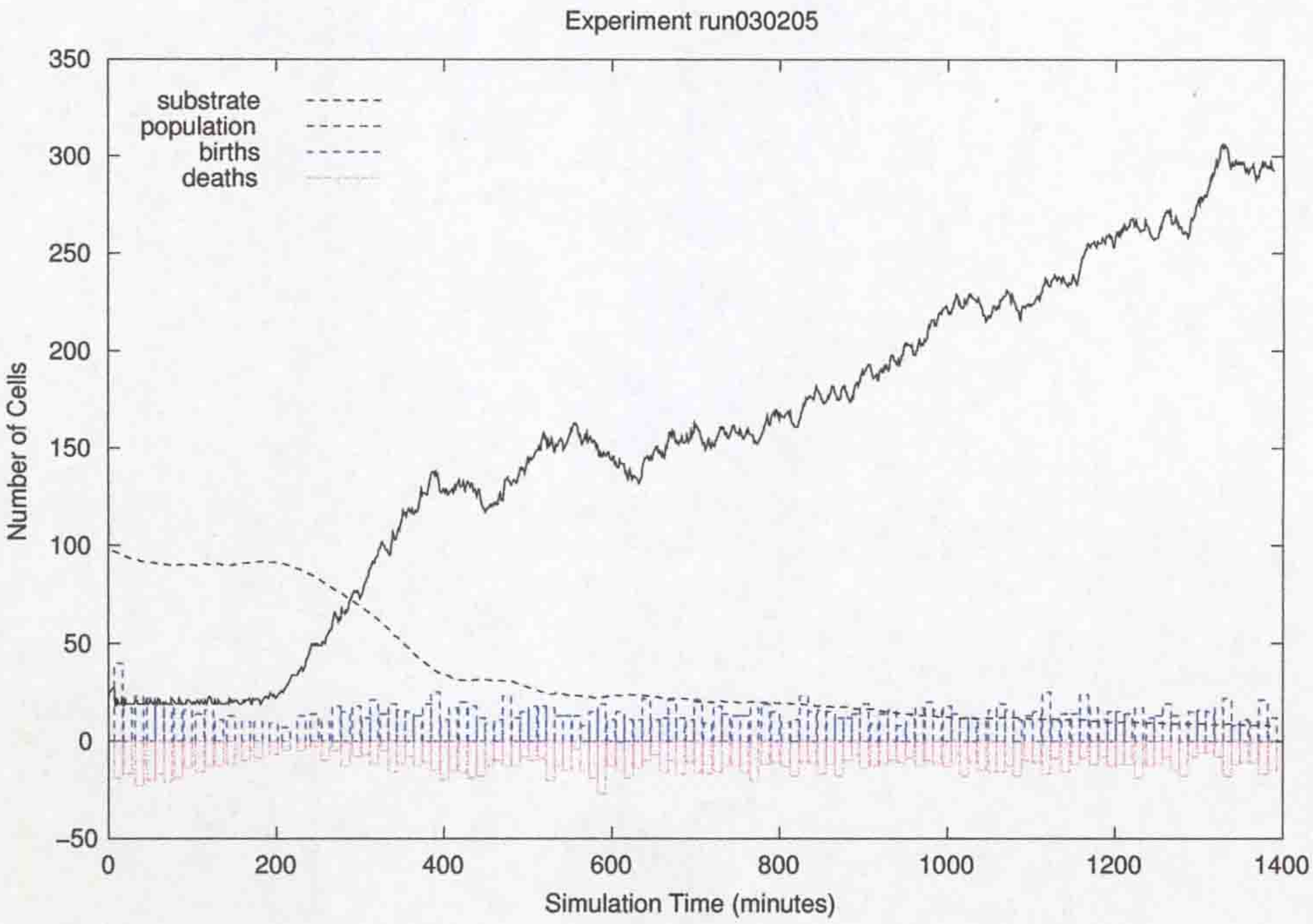


Figure 5.4: Population growth during a typical simulation.

50:50 chance of staying with the parent. It is an implementation decision that parents retain their identifier rather than obtain another identifier.

Simulation load is not constant, the ever changing number of processes per machine is constantly changing. Figure 5.4 shows the rise of the cell population and hence the rise in process numbers. Each cell equating to around 9 processes, the main cell and output data compression processes. There needs to be many more cells than processors before efficiency can be attempted. Notice also the stable number of births, the system is forced to synchronise to these new cells and we would expect to have an impact on efficiency, but as shown later efficiency is too variable for this to be a cause.

Figure 5.5 shows the efficiency of the whole simulation system over the course of a long run lasting around 2 weeks of wall clock time. This graph shows several measures, the upper line is the maximum available computation time. Computed from the maximum processor usage across all processes at that instant. The noisy nature of this plot implies more could be executing

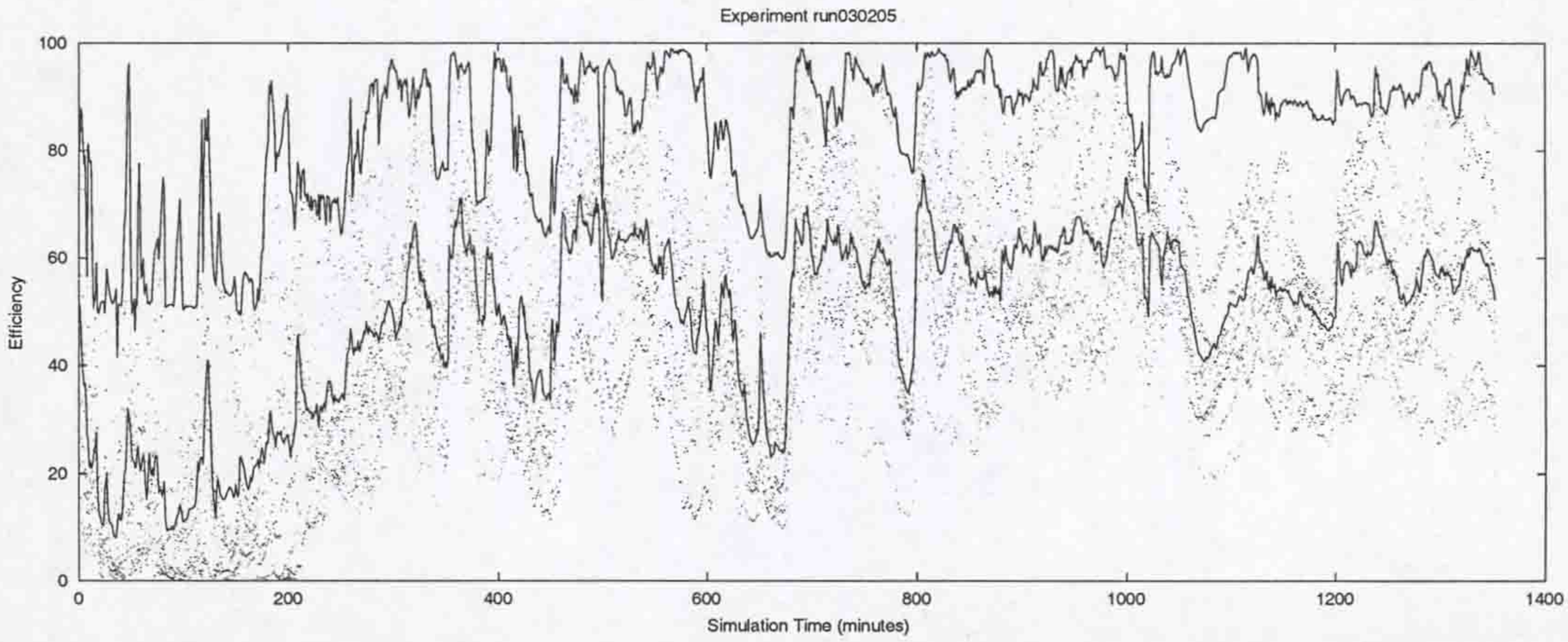


Figure 5.5: Overall efficiency of a long simulation run, the upper line being the maximum available computation time. The lower line being the actual usage at that time. The difference between 100% and the upper line comes from other processes started by COSMIC but outside the auditing process, most likely during cell division. The difference between the two lines comes from process balancing errors inherit in PVM and its blind allocation of processes.

than is accounted for, each machine in the cluster (except for the main node) is dedicated to COSMIC and so should never themselves hold back the simulation. However, the main node is shared with other people and processes, this could easily account for some of the drops in available resources. The other possibility is simply unaccounted processes during cell division, the synchronisation bottleneck of cell division and starting another process forces the simulation to momentarily pause. This pause was deemed insignificant during design but here it seems this and the presence of other users can have a big impact, much bigger than it was thought.

The lower line of the same figure depicts the actual processor usage of COSMIC, as can be seen this closely follows the available level. The difference between both lines comes from process balancing errors inherit in PVM and its blind allocation of processes combined with the dynamic needs of the processes and lack of process migration. Shown as small dots amongst figure 5.5 are the data sets that make up the maximum and actual efficiency values. Each represents the processor usage at that instant, and so if processor usage was the bottleneck we would expect at least one of these to be 100%, with the

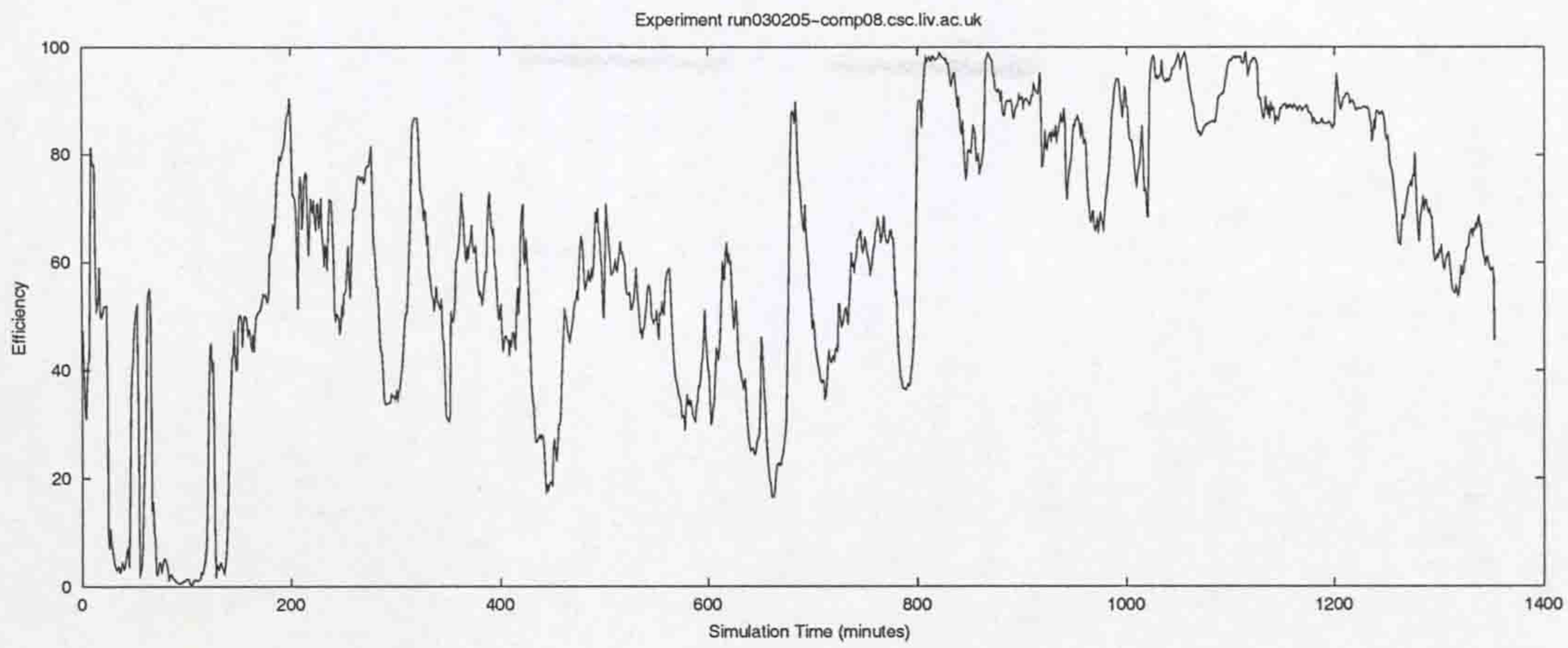


Figure 5.6: Efficiency of a single randomly chosen machine.

lowest being the most idle processor. Idleness is caused by cells on a machine being quick to execute in comparison to other cells, quick execution followed by the environment-cell synchronisation requirement requires that a cell waits.

Picking out a machine at random from the 10 available, figure 5.6 shows how variable utilisation is, varying from nearly 0% to nearly 100%. Comparing against figure 5.5 we can see load balancing had a large part to play. While this machine was idle there was still a significant level of load, often around the 50% boundary. At the early stage this is telling, comparing to the cell population of figure 5.4 the population was rather small and each processor could have approximately been used by only one cell. The increase in peak efficiency then comes with the increase in the number of cells, allowing a better global balancing of load though still far from optimal.

5.13 Summary

The COSMIC model is a vast tool for modelling evolution at the genetic scale, the parallelisation of this model has been completed and been shown here to perform adequately on a small cluster. Output data suggests several causes and solutions for the lower than expected efficiency. The most compelling is the load balancing of cells with a dynamic execution time, this has several broad

solutions. Firstly, the load balancing of PVM could be improved to better take account of the expected load brought about by each cell, this requires writing a load balancing function for PVM and finding a reasonable measure of cell complexity. Another option is to use MOSIX or another Single System Image software solution[7]. Other options lie in either better parallel API implementations or perhaps a more standardised MOSIX like system. This latter option would be an ideal feature of future GRID middleware that has yet to arrive in GRID. Regardless of the efficiency issues, the parallelisation of COSMIC has made the individual based modelling approach feasible and practical while still covering several scales. This is no panacea, in a few years time what is practical only on a cluster will be practical on a desktop machine, the cluster only raises the bar by around 20 times, a bad algorithm will always be slow regardless of the machines used.

[7]Generally a Linux kernel patch which supports the migration of running processes between machines. This completely solves the load balancing problem but creates another problem through being highly OS dependent

Chapter 6

Visualisation

6.1 Introduction

This chapter introduces the most common visualisations used to represent the raw data generated by COSMIC. It would be expected that many of these ideas come directly from their biological counterpart, but as was hinted during the chapter on biological background, biology can be severely limited by what can be measured. Available data dictates what visualisations can be constructed with any degree of usefulness and so the legacy of practical biology leaves COSMIC waiting for a distant future of biology; there is so much more detail available from COSMIC at every level of measurement than there could be from years of wet lab research. This then makes full validation near impossible but for this chapter it also means there is the opportunity to create something new that allows access to manageable data sets at a time when the real world is some way behind.

The two scales of COSMIC give the initial division of the visualisations, the top level environment where the cells play out their struggle for survival and ultimately demonstrate their evolution. These are covered in sections 6.2 and 6.3. At the other scale there is the internals of each cell, which contains the richest data but also the hardest to view in any one way that captures all the changes. This is covered in sections 6.4 to 6.6.

6.2 Environment - Glucose concentration

As the environment is what cells should be differentially responding to (i.e. the selector), it is important to ensure there is sufficient information in the environment to make evolution possible. The COSMIC system saves snapshots of the environment in terms of glucose concentration every 10 seconds of simulated time. From this we can identify cells, trends in motility and trends in substrate use/replenishment. For each of these snapshots, white equates to a glucose level of 4.5 mg, black equates to absence of glucose. The dimensions of the square are 0.2 mm. Filled black circles represent bacteria that have not moved through lack of connection with their flagella, grey streaks show moving bacteria. (Note: Bacteria in this system cannot move without leaving some visible trail because they always consume a visible level of substrate). Per cell glucose use has been exaggerated to better motivate evolutionary change, real *E.coli* use such a small proportion of substrate per cell that one hundred times more cells would be needed to sufficiently modify the environment. Clearly there are insufficient resources for that number of cells and so substrate usage was increased one hundred times. The environment is made up of a 500x500 array of `floats`, giving cells enough resolution to have a reasonable chance of sensing gradients while also not consuming vast amounts of space when saving snapshots.

Using these snapshots it is also possible to generate an overview of the rise in population and the fall in substrate, as shown in Figure 6.1. This shows quite clearly the different phases of COSMIC simulation; the search phase that looks for a solution genome which can at the very least move its host cell, even if there is no control; once that cell is found it then exponentially takes over the population; finally that cell linage has consumed most of the substrate in the environment and the population is restrained to slow growth. Each of these phases are discussed below, with an example of the environment at the time.

The environmental views can be shown as a time series by stepping through in either direction as a movie showing the deforming environment, or looked at more closely by annotating the image with legible cell numbers. Both types are

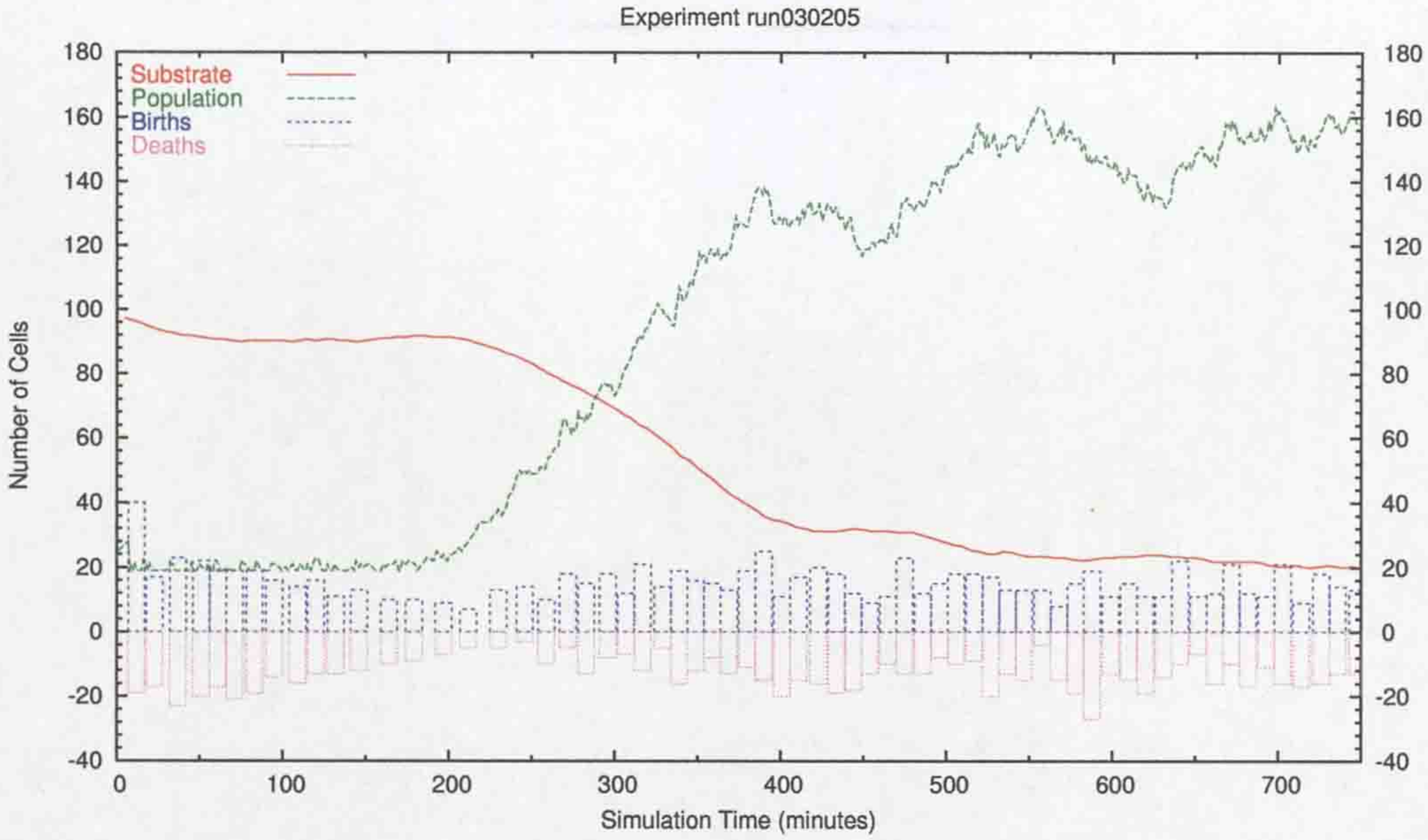


Figure 6.1: Population size and substrate concentration, 4.5mg glucose is here shown normalised to 100.

presented here. Figures 6.2, 6.3, 6.4, 6.5 show four views of the environment over a widely spaced period of time. These show the searching phase, the knee of the exponential growth, during exponential growth and some time afterwards when the environment has become nutrient restricted in most areas.

Figure 6.2 show a typical view of the environment during the initial searching phase. Cells sense substrate level on a linear scale but the view presented here has been adjusted to emphasise the grey[1]. COSMIC maintains a cell population of at least 20 cells, this is important during the search phase because the random genomes lead to frequent cell deaths. Without this maintenance the cell population would normally reach 0 in a matter of minutes. There are around 20 cells shown here, living cells can be seen circled and uniquely numbered(Section 5.12), allowing a cross reference to the other data sets. Filled black circles are dead cells, only their previous effect on the environment remains. In time the black diminishes as the environment is replenished. Moving cells leave fading trails for the same reason. Overall then we see cells that are

[1]Through experience it was found that light grey very easily looks pure white when there is no known pure white with which to compare.

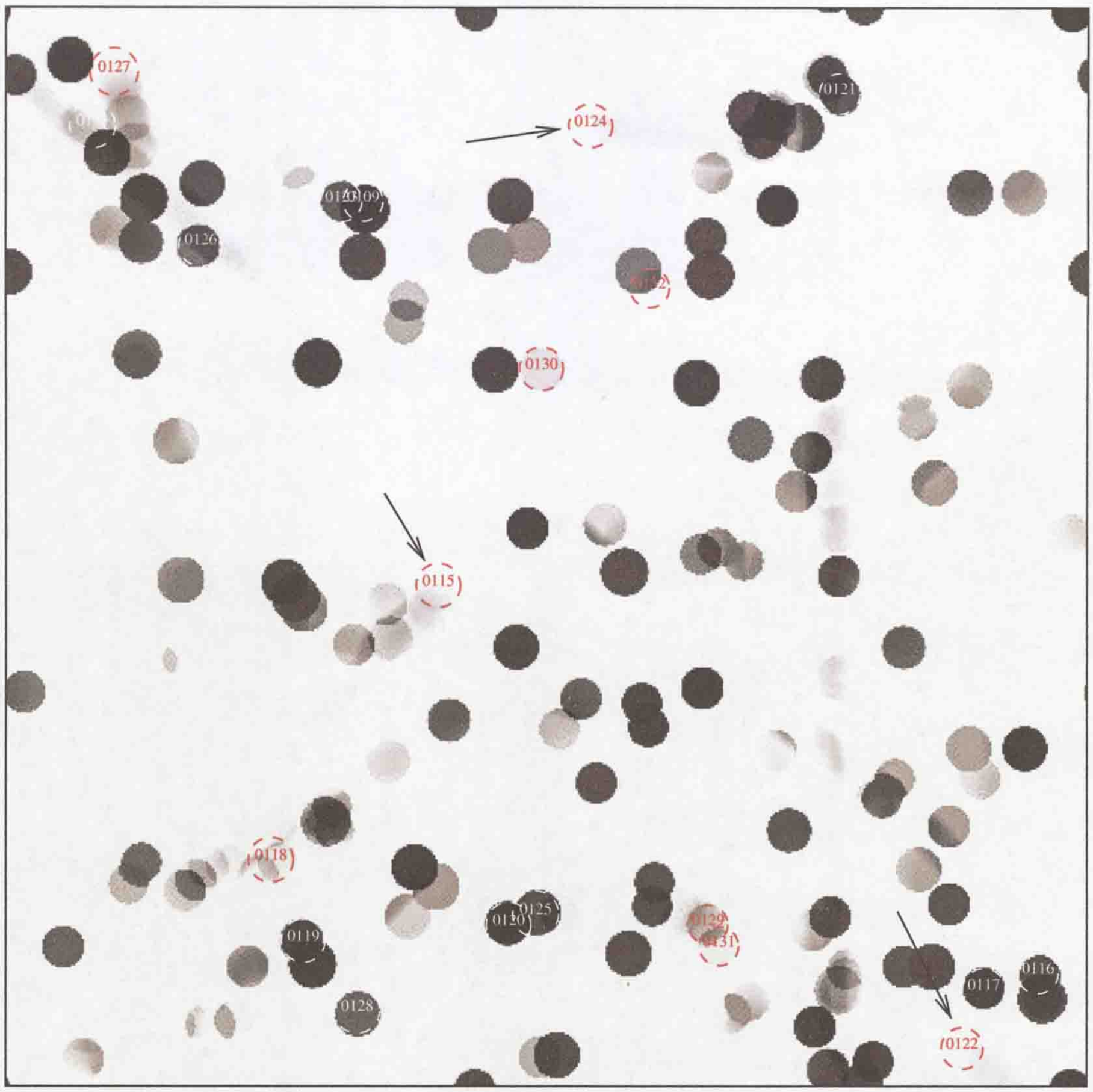


Figure 6.2: Environment at t+4000 (t+66.6 minutes), covering 0.2 mm square. Living cells can be seen circled and uniquely numbered (Section 5.12), allowing a cross reference to the other data sets. Filled black circles are dead cells, only their previous effect on the environment remains. In time the black diminishes as the environment is replenished. Moving cells leave fading trails for the same as their effect on the environment is slowly reduced. Arrows indicate cells mentioned in the main text.

either alive or the effects of cells that were recently alive - without this replenishment the environment would quickly become useless as a scenario for evolving chemotaxis.

Some cells in Figure 6.2 are apparently doing well, they are moving unlike the majority that don't move at all. Cell 0115 (note all these cells have been clearly marked on the figure) near the middle moving generally up and right would seem to be a good candidate, as would cell 0122 on the bottom right and cell 0124. Looking at the mass of black from where 0124 came, it seems reasonable that this cell was the result of a recent division and the parent died in the process. Unfortunately the data confirming this has been lost, this is however a common effect.

Figure 6.3 then shows the environment at around the start of the exponential growth with the finding of one cell which has the ability to move, regardless of any control. This shows several dark stripes left by cells resulting from a divided pair moving in the same direction but at different speeds, cells 0225, 0265 on the top middle and cells 0266, 0249 (bottom middle) and possibly 0273 and 0257 are related, though the latter two move in different directions. Another related pair can be seen on the bottom left of middle, moving down; 0255 and 0271 (all 8 are marked with a plain arrow). At this point the daughters of cell 0143 (marked with a starred arrow) are already starting to take over the population, 10 of the cells here belong to that lineage, they are cells 0193, 0225, 0248, 0249, 0253, 0263, 0265, 0266, 0268 and 0269.

Figure 6.4 shows the environment 66 minutes later, with the daughters of cell 0143 taking over the total population. There are now dark streaks created by groups of cells moving in the same direction, in this case the darkest streak is caused by the same group of cells wrapping around the environment. Later on this effect will dominate as more cells have motility. There are now 40 cells belonging to the 0143 linage, to many to list. There are, however, some other lineages here amongst the growing number of 0143 related cells, these are left over from when the cell population was around 20 and continually seeded with new cells. In this struggle, the growth rate of the 0143 related population must

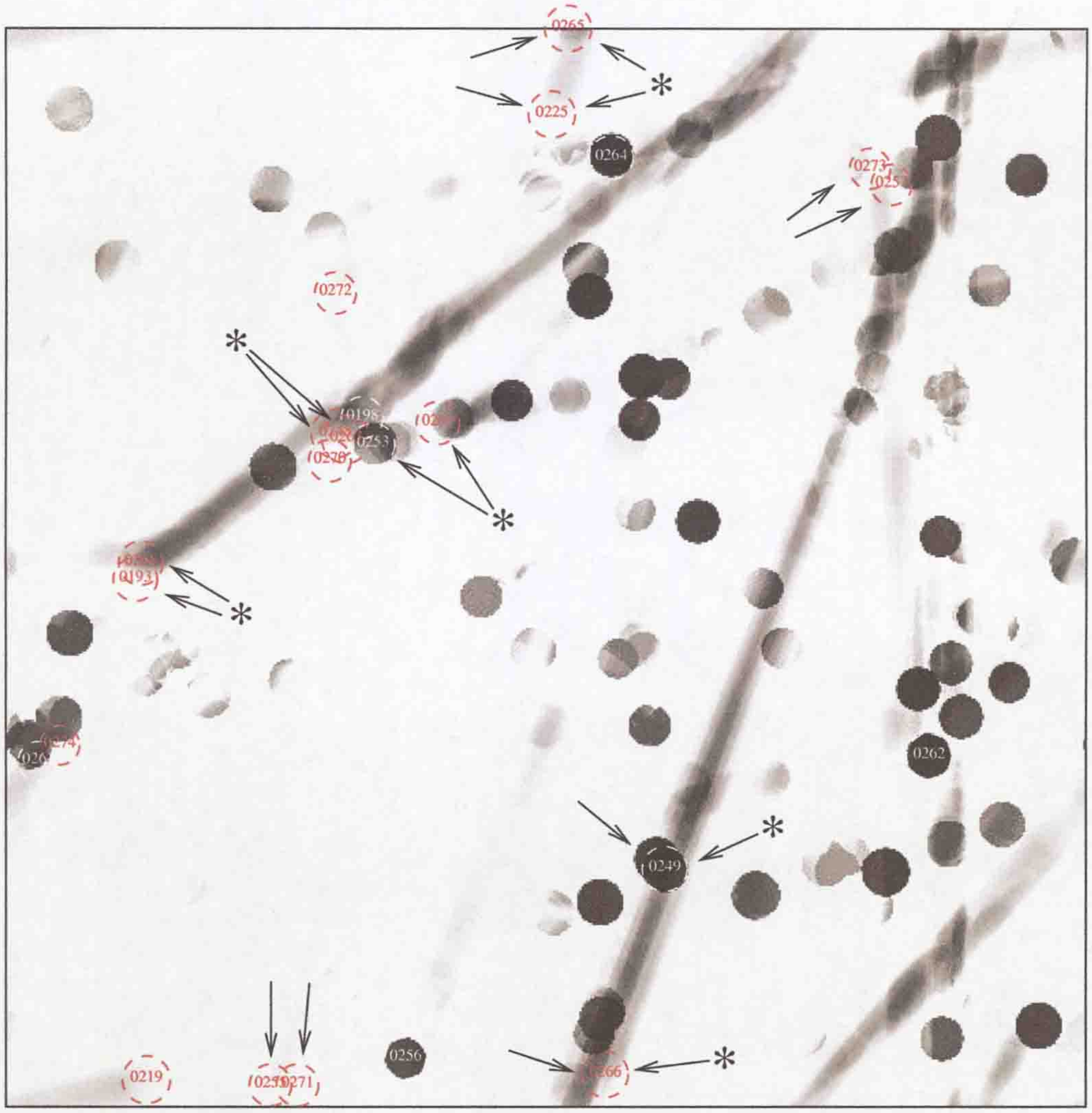


Figure 6.3: Environment at t+12000 (t+200 minutes), showing the proliferation of cells.

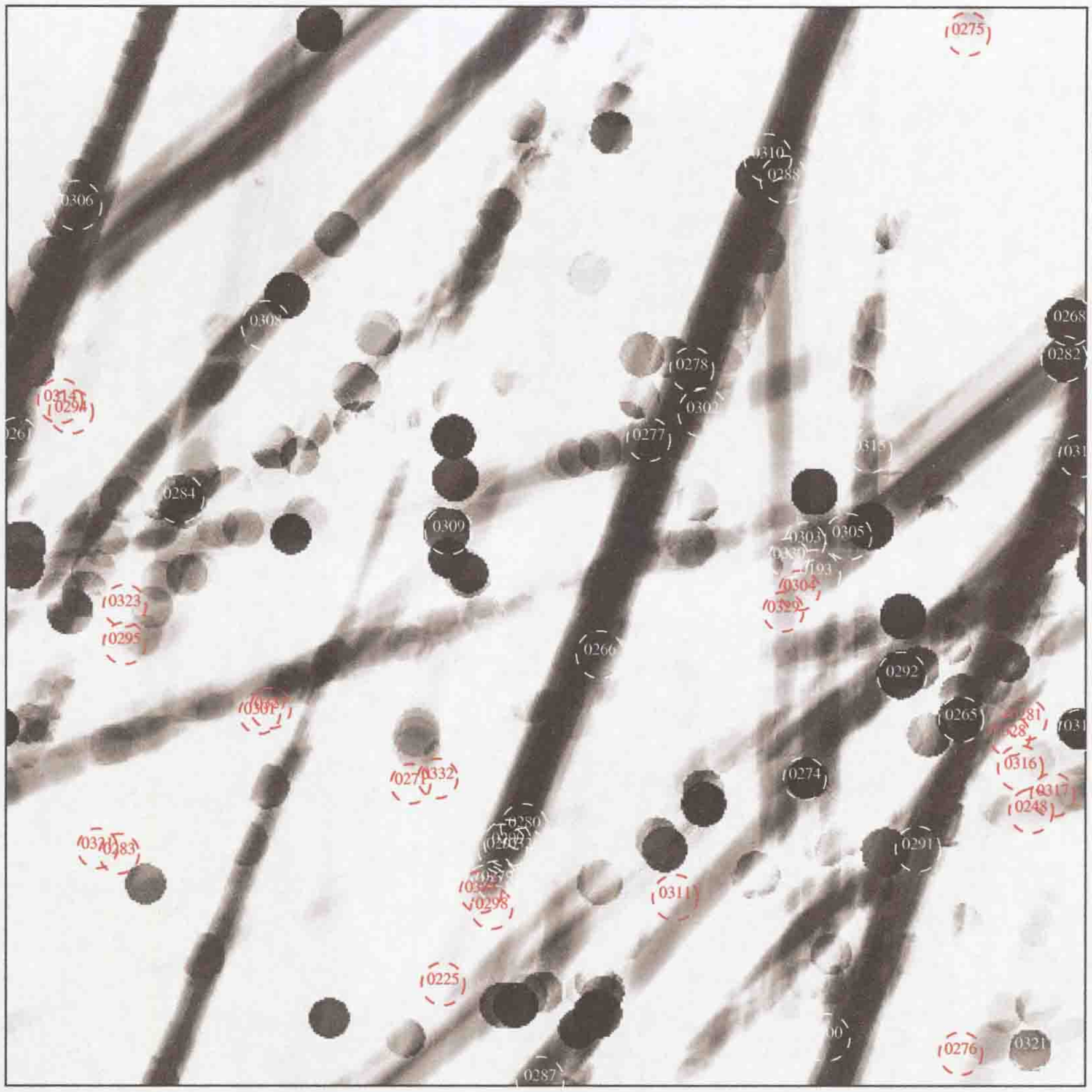


Figure 6.4: Environment at t+16000 (t+266.6 minutes), showing the continued proliferation of cells.

eventually wipe them out short of a major change in the genetics of those other lineages.

The final detailed view of the environment is shown in figure 6.5, here we can see immediately the environment has become depleted to the point of all cell indicators showing as white, meaning the average concentration in those locations has fallen to less than 50% of the start condition. Unlike the other images, this image is presented without brightness distortion(linear brightness). This point marks the end of the exponential rise and large drop in average environmental nutrients, together with the first drop in population. The population has dropped to 126 cells at time 28000s, from 137 at 23500s, actually dropping to 117 cells at 26800s. Of the 137 cells, 116 belong to the 0143 lineage. So, even at this point the smaller lineages still manage to linger.

Multiple views of the environment are shown in figure 6.6 and continuing in figure 6.7. These images show the environment over successive time steps. Time steps of 10, 200, 2000 and 4000 seconds are shown, from then on time steps are in 4000 seconds. Time 12000 marks the start of the population explosion, we would then expect to see some signs of more intelligent behaviour. Intelligent behaviour is too difficult to see when there are many cells, instead cells need to be considered on an individual basis.

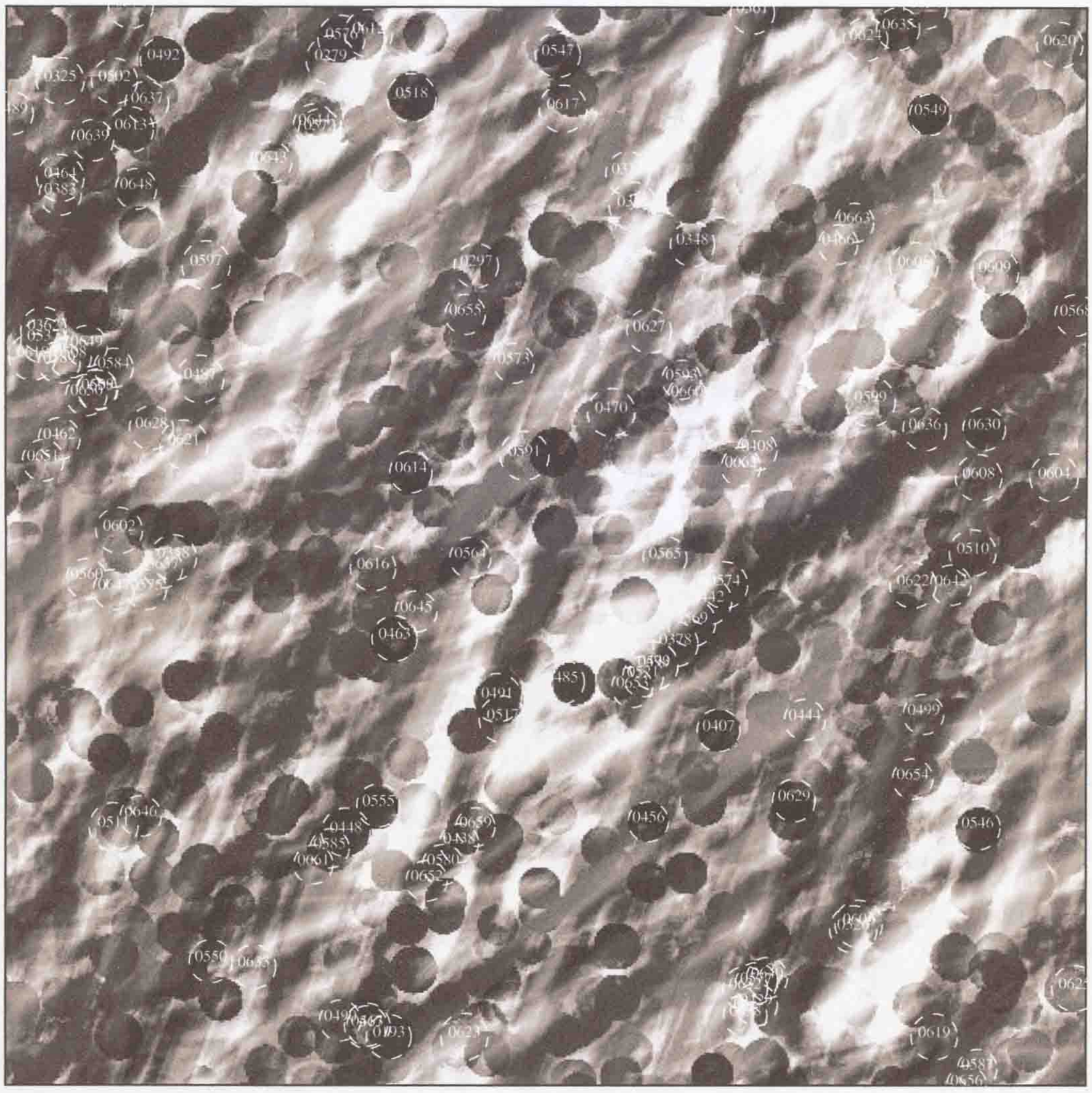
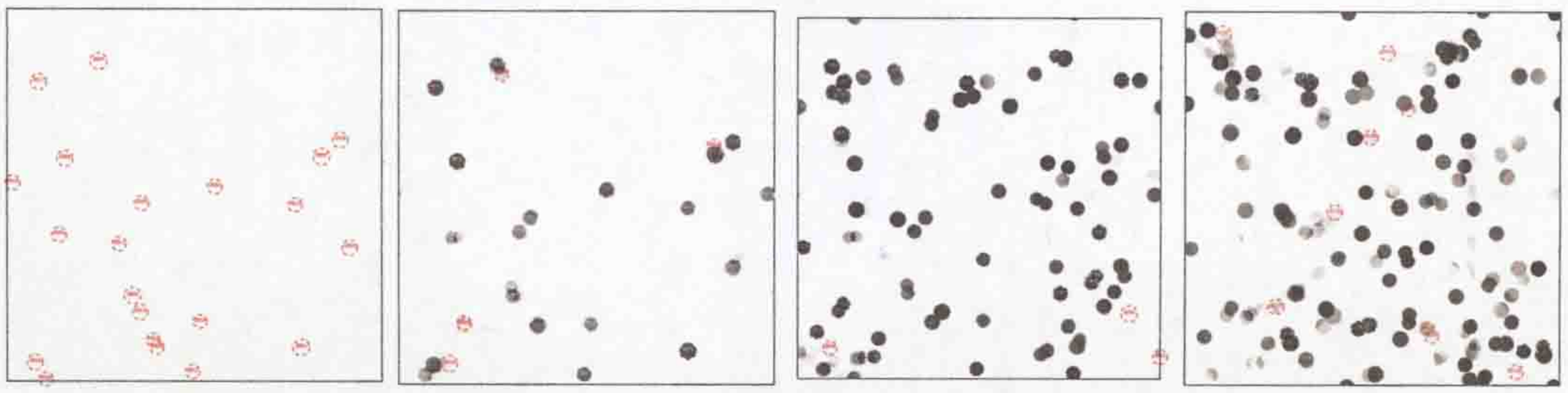


Figure 6.5: Environment at t+28000 (t+466.6 minutes). Shown with linear brightness as black is now the dominant brightness.



t+10 t+200 t+2000 t+4000

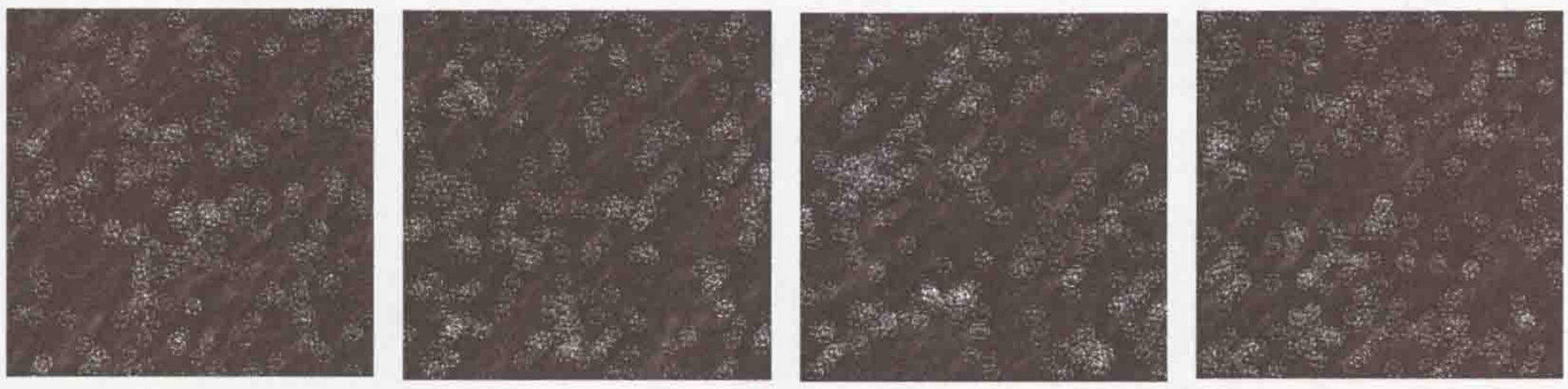
t+8000 t+12000 t+16000 t+20000

t+24000 t+28000 t+32000 t+36000

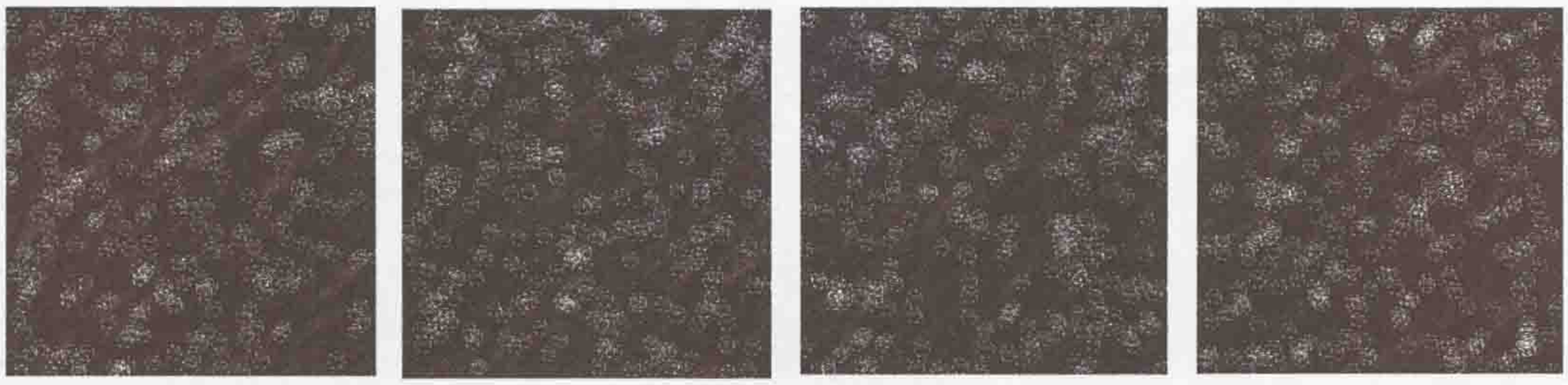
t+40000 t+44000 t+48000 t+52000

Figure 6.6: Environmental time slices, part 1.

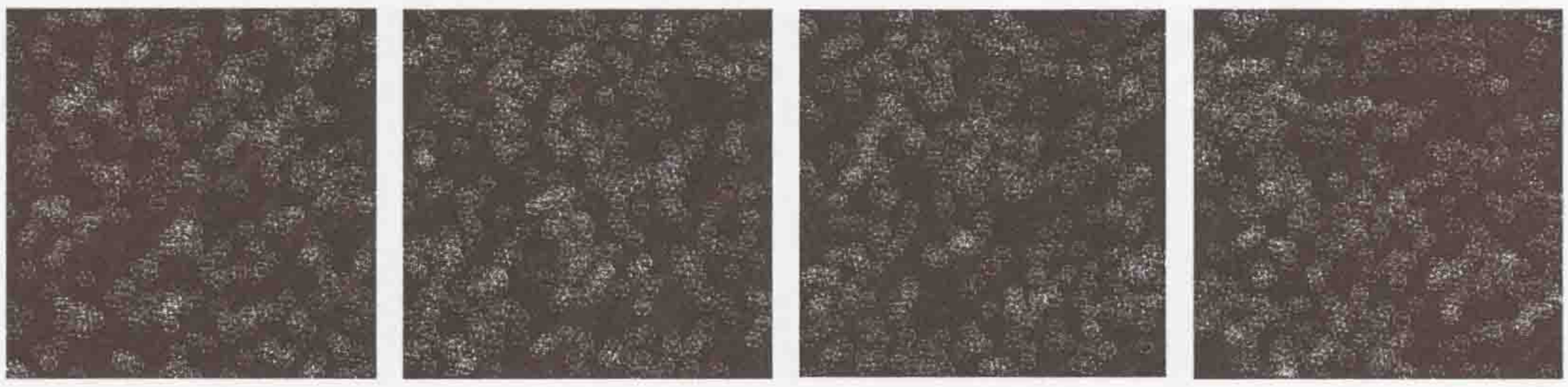
t+56000 t+60000 t+64000 t+68000



t+72000 t+76000 t+80000 t+84000



t+88000 t+92000 t+96000 t+10000



t+10400 t+10800 t+11200 t+11600

Figure 6.7: Environmental time slices, part 2.

6.3 Population lineage

Cell division brings about the possibility of recording relatedness between cells, this can then be shown as a digraph of parenthood, with the parent at the top and daughters below. On top of this other data can be incorporated, such as the times the division events occurred. Shown in figure 6.8 is an example of this drawing, showing the top section cell lineage 0143. As there were 2000 cell divisions and nearly 2000 cell deaths attributed to this one lineage, the full diagram is huge and unwieldy and would use around 4000 lines to display, so only the first few tens of lines are given. The generation of these diagrams makes some attempt to make them of usable size by ensuring the graphs have less then a given area of nodes, making the diagrams shorter as they get wider.

In this example, the lineage is headed with some general information on the tree size, followed by the cell number leading that lineage - cell 0143. On the left there are then time stamps for when the event on the line occurred and the time difference between this event and the last event. On the right there are numbers representing cells that lead to other cells, i.e. the divisions. In the diagram itself the numbers represent deaths of that given cell. Bold lines mark every tenth line to try to ensure some readability.

As the number of related cells grows the lineage becomes increasingly congested, as figure 6.9 shows. Here the time ranges from 1601 to 1607 minutes, just 6 minutes and yet there the large number of events per cell with no synchronisation means that each event will have its own line, greatly increasing the required space. The result in this case is barely readable and is here only as an illustration of the problem. It could well be that a less structured diagram with only local labelling and no synchronisation between unrelated but identical events will compress this to something more manageable.

This can be made even more congested by showing all the cells that existed at the same time rather than from a lineage, as shown in figure 6.10. Here the structure between cells is necessary, at least in the beginning of each new cells life, and so the diagram is bound to be large. The obvious addition here are new cells being introduced on the right to maintain the minimum population,

as cells die those cells then tend to move to the left. It was considered if the graph would be better shown as a time line in which height was time (generally as is the case now) but that cells never moved left and the related lineage hung under the original ancestor. This would have been easier to interpret but used vastly more space, on average occupying a diagonal. The per lineage diagrams carry most of the meaning but save space by keeping the lineages separate.

Even in its basic form, this is a novel structure that has yet to be introduced into mainstream biology, presumably because the figures on which it is based are impossible to obtain. Figures could be obtained using genome sequence analysis, which would give a general lineage, never as specific as found in COSMIC but even this is too time consuming to yet be realistic. As shown by COSMIC, this method is not without its problems, approaches to scaling up this visualisation represent future work.

6.4 Cell statistics

The previous sections were concerned with the environmental scale and so the population of cells as a whole. This section and those that follow change the scale of view to that of the cells internals, where we find most of the challenges to visualisation.

This section shows some of the general dynamic parameters associated with each cell. Each graph normally covers the time of the cells life, unless the cell went onto live a long time, in which case the graph is a maximum of 7.5 hours. These graphs can be useful in finding the high level reason for a cells death or its popularity as they show key indicators of the state of transcription.

All graphs show cell mass to start at some value determined by the random uniform initialisation. The mass then increases according to Section 4.11 to the maximum of around 0.4 femtolitres, which then triggers cell division and the volume of this cell is halved as well as halving the enzyme concentration.

The x and y position graphs correspond to the position in the environment as shown later. $(0, 0)$ is top left and (200,200) covers the 2mm x 2mm area.

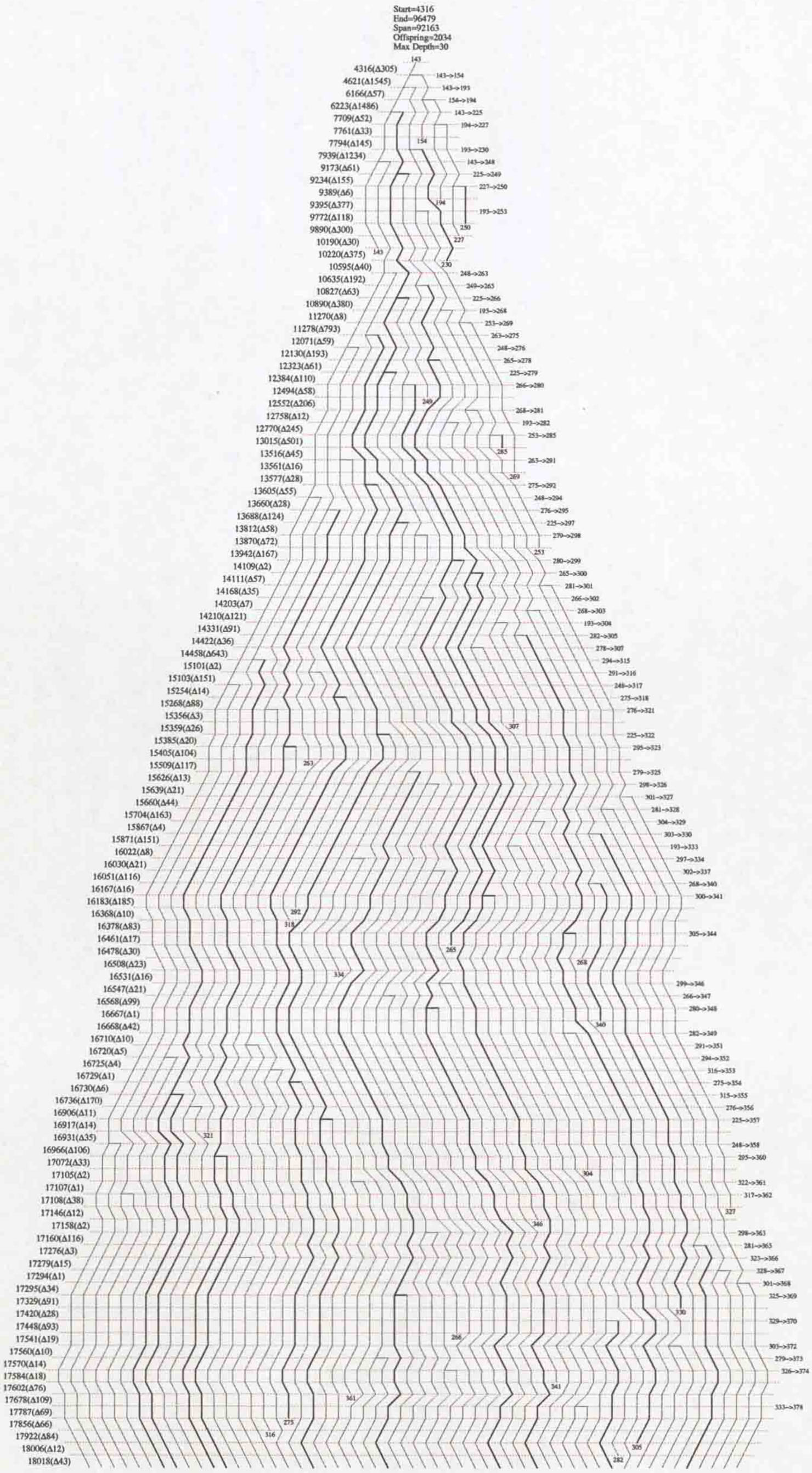


Figure 6.8: First 5 hour lineage of cell 0143.

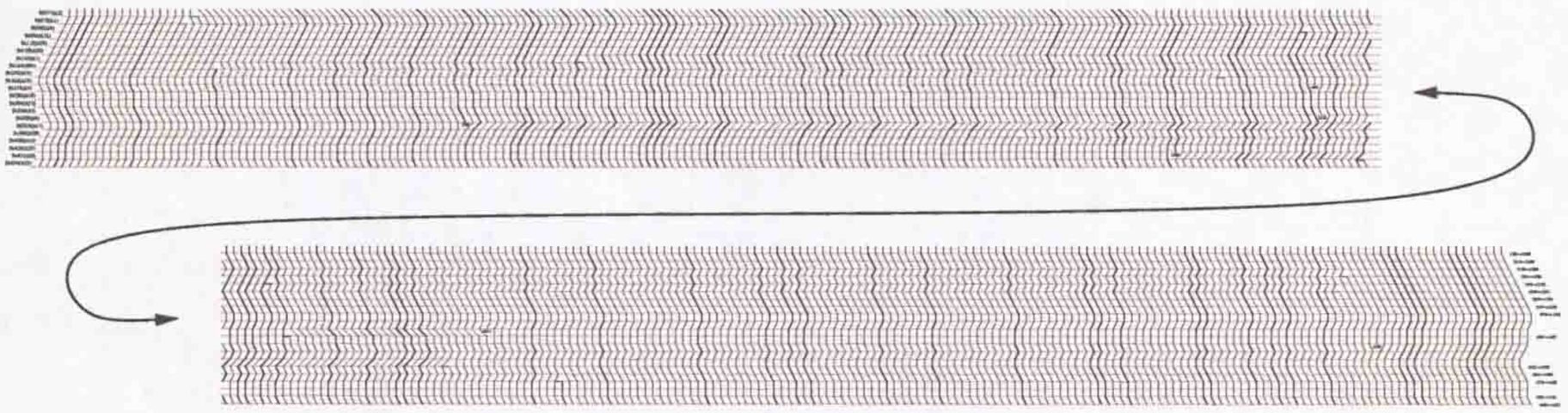


Figure 6.9: Last 6 minutes lineage of cell 0143. Shown cut in half in improve its reproduction.

The enzyme population graph plots both the minimum and maximum number of enzymes in each time frame, the line thickness is the same for all graphs and yet all enzyme populations shows that a small difference persists

The input figures come from an average of all activated receptors, hence a typical figure of $0.02 = x = |\Phi|$ (i.e. the input receptors) shows one receptor to be used in a given time interval - x is the maximum of all time instants in that interval. Clearly this value must be expected to increase during convergence.

The output figure is based on $|\Omega|$ (i.e. the flagella activation receptors) but is otherwise the same as the input. There is always a strong link between this value and changes in the (x, y) values. Ideally there should be a strong link between the output and glucose concentration in the environment.

6.4.1 Cell 143

As the original ancestor of the lineage that will take over the whole simulation, cell 0143 in figure 6.11 would be expected to have all the qualities that enable it to survive the environment and avoid the early termination algorithm. The survivability of this cell comes across from all these graphs, the cell volume is growing at near maximal rate and so dividing at near maximal rate of 22 minutes. The x/y positions in the environment show the cell to be moving at speed and so always being positioned over fresh environment - at least while the environment is sparsely populated. The enzyme level is high, seemingly high enough to survive cell division 3 times, it was the 4th division

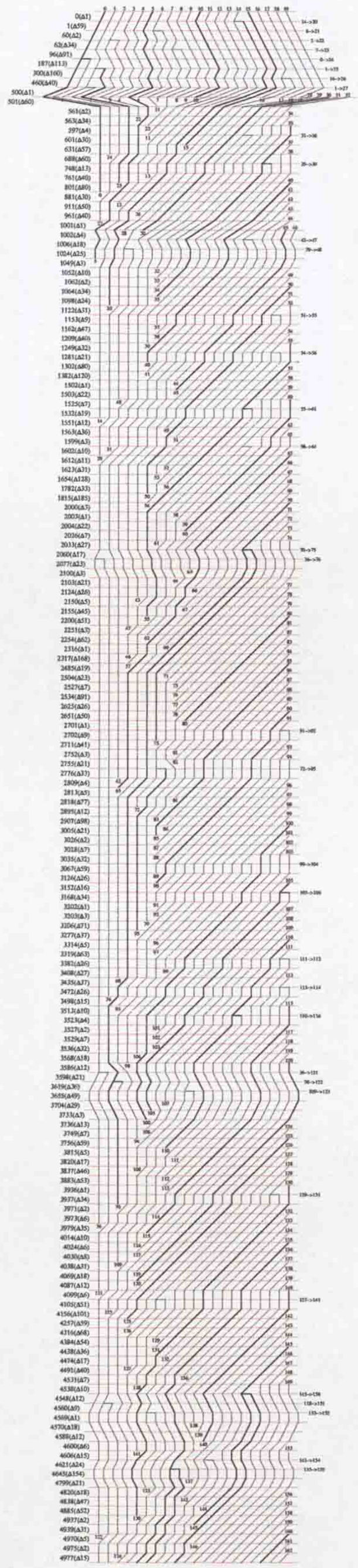


Figure 6.10: First 83 minute lineage of whole simulation.

that caused a failure of the cell, the enzyme graph shows the steady decline in enzymes shortly after the 4th division. As this is plotted on a log scale, this steady decline represents the natural decay of enzyme half-lives. The Output graph shows why the cell was moving so quickly, on average one of the flagella was activated most of the time, this pattern only stops shortly after the 4th cell division. This cell also looks promising because of its connection to the receptors, suggesting that a link between input and output is at least possible even if it doesn't exist at this moment.

6.4.2 Cell 101

The cell in figure 6.12 typifies a cell containing futile cycles, this cell makes no connection to the environment (as shown in the input/output graphs) despite having a steady enzyme population (as shown in the enzyme population graph) after initialisation. The cell grows using only the nutrients in its immediate environment, but as there is no movement the environment becomes depleted and so the growth rate slows (volume graph). The lack of movement fits the criteria of a failed cell and so is killed off early by COSMIC heuristics that attempt to remove useless cells.

6.5 Gene Expression

The expression level of individual genes can be useful information when accessing the genetic quality of a cell. This section demonstrates such a visualisation. The figures show the gene expression level over a period of time. Looking closely, each picture is made up of several rows, each row representing 15 minutes of the gene transcription, with each line (or lack of a line) in that row representing a given gene. Rows are read from left to right, top to bottom. As shown on the left of each row is time frame relative to the cell whose number

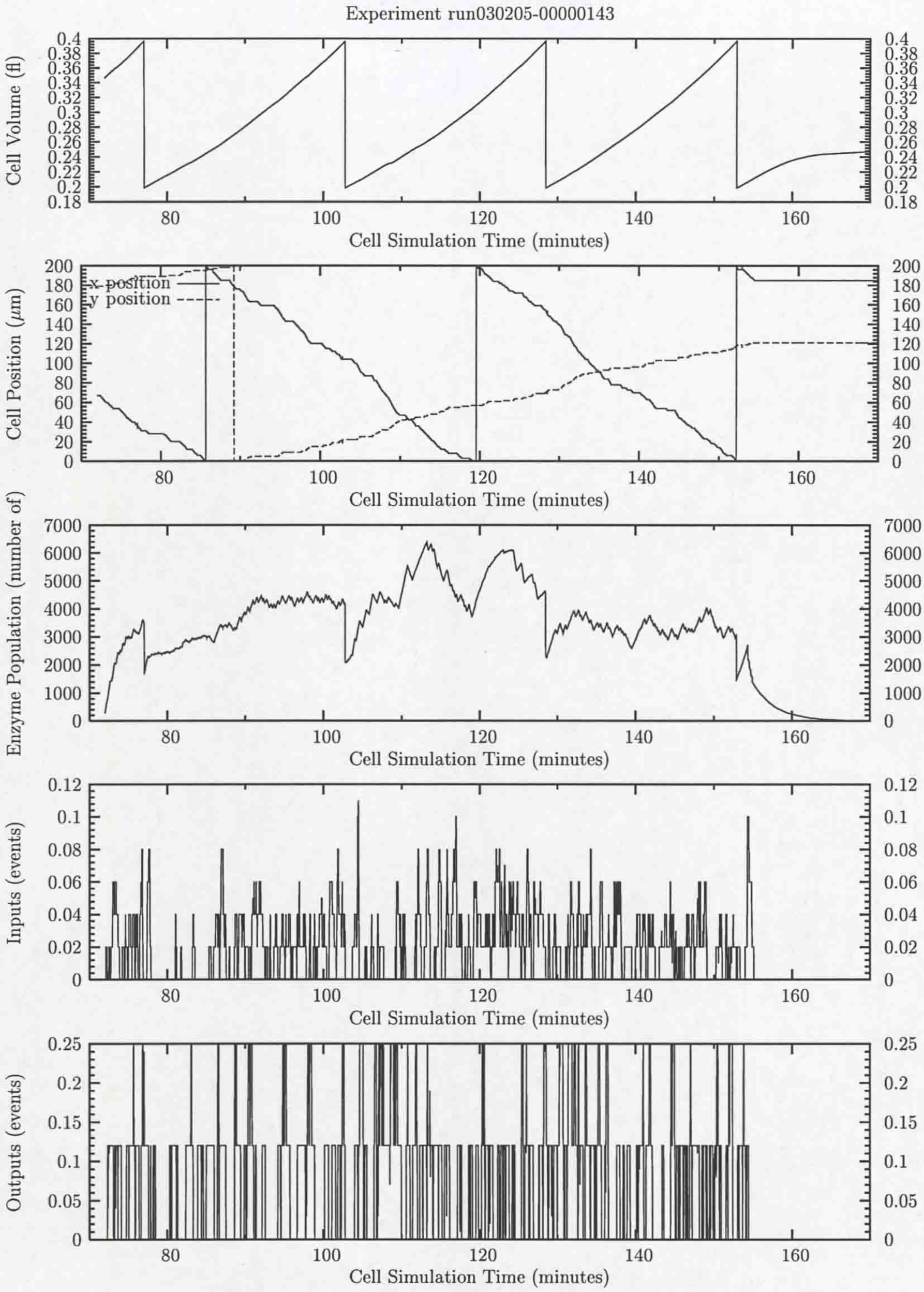


Figure 6.11: General variables of cell 0143

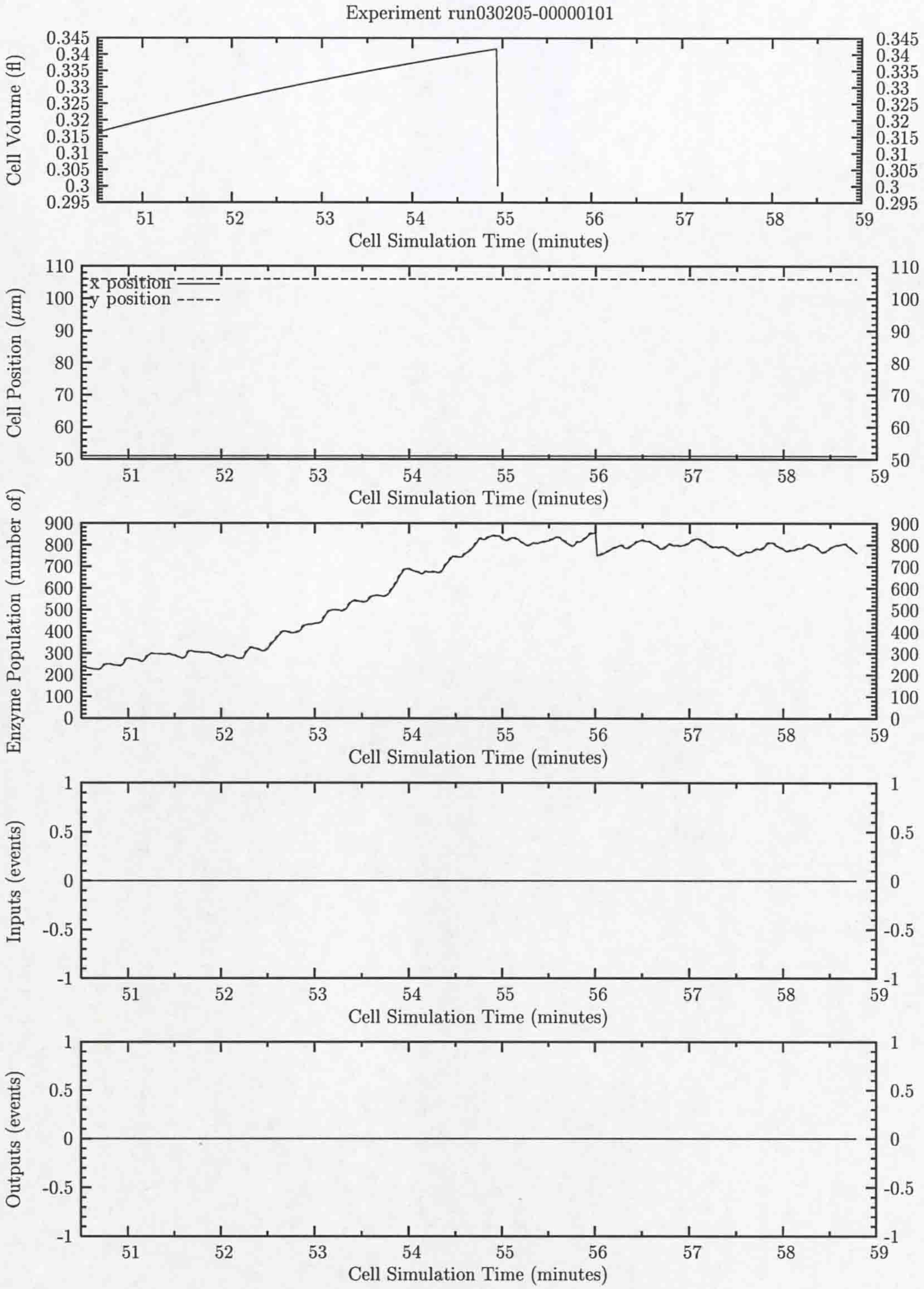


Figure 6.12: General variables of cell 0101

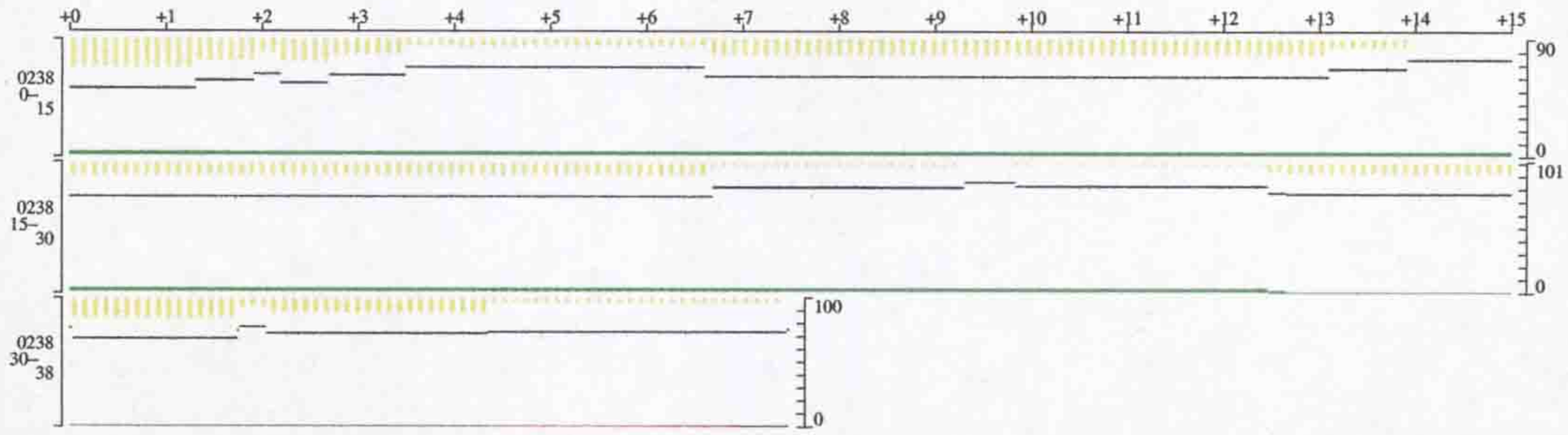


Figure 6.13: Gene expression of cell 0238, showing very little expression.

is between the time frame. On the right hand side are the gene numbers, the first gene being at the bottom of the row. When shown in colour, colour and shade denotes number of gene products present in the cytoplasm at that time, white denotes no expression, blue denotes 1-3 gene products, red denotes 4-20 gene products and green denotes 21-1029 gene products. Notice they are on the logarithmic scale to take account of the vast difference in volume of gene products.

In all cases cells are initialised as described in Section 4.13, there are 5 gene products for each identified gene.

Note that in cases where the cell was terminated and transcriptional data stopped, the unused area is removed. Also note that as sequence insertion and deletion change the size of the genome, so to does the height of each row. As the genome is inserted from the bottom, the top end point of the genome can vary. In this case the individual genes are painted from the bottom of their respective rows, and any space remaining at the top is padded with vertical alternately coloured bars.

6.5.1 Cell 0238

Figure 6.13 shows a typical example of a failed cell, after initialisation there is little activity. Looking closely the expression level can be seen on the first few genes and around gene 85. Enough remain for the simulation to consider the cell alive, simply because COSMIC is lenient in the heuristic decision to kill a cell that does not seem to be motile.

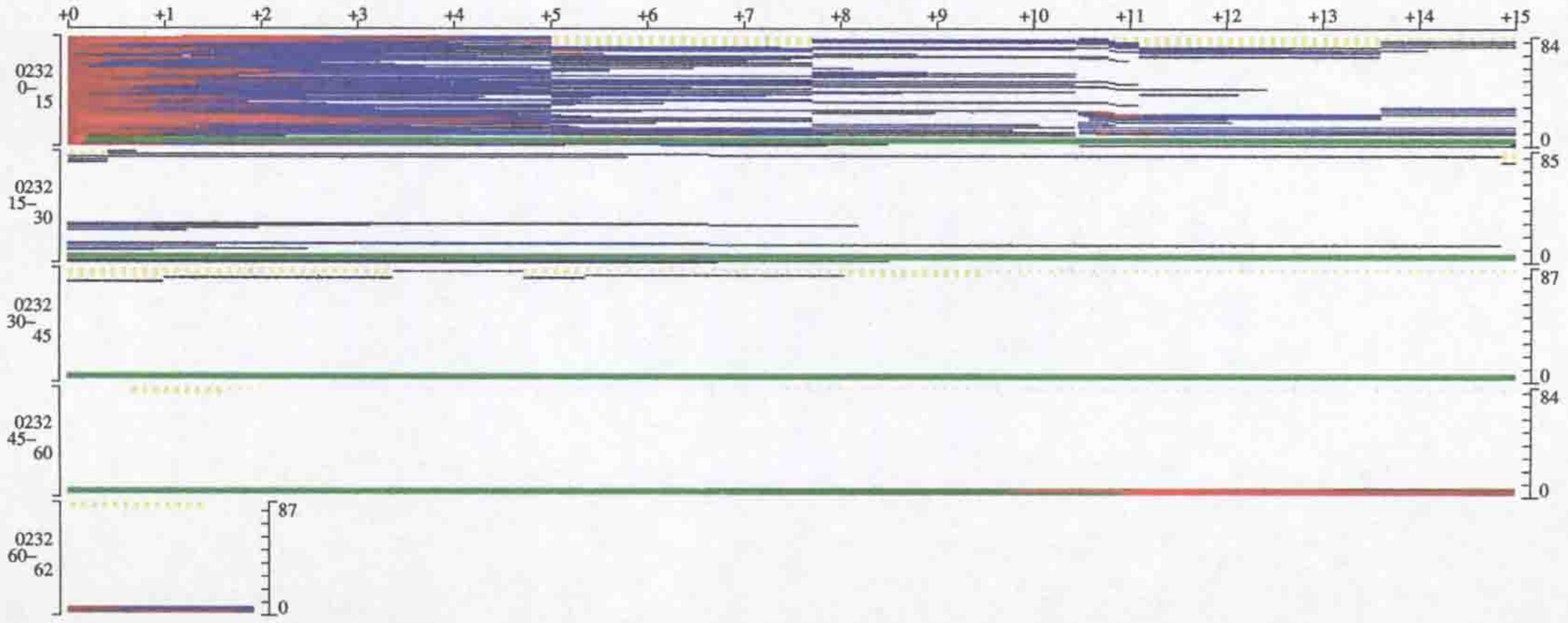


Figure 6.14: Gene expression of cell 0232, showing some strong gene expression but still suffering a fatal loss of expression latter on.

6.5.2 Cell 0232

Cell 0232 in figure 6.14 is similar to cell 238, the main difference is the more sustained transcription level involving more genes. This cell also shows fluctuations in the expression of these genes, for example at time 38 minutes. What seemed a stable network of transcription then somehow fails, ending at time 62 minutes when the cytoplasm is effectively empty.

6.5.3 Cell 0219

Figure 6.15 shows a cell with even more activity than the previous cell. Looking more closely we see multiple fluctuations at many points in time. This cell is also notably for having no visible initialisation at time 0 minutes, so this cell must be the result of a cell division.

6.5.4 Cell 0204

Cell 0204 in figure 6.16 clearly demonstrates an inactive cell. Initialisation occurs like other cells but no transcription occurs leaving the cell to a fate of death. The rapid death was dictated by enzyme life times, here chance dictated there would be no long lasting enzymes when initialised and so the cell is terminated when there are no enzymes remaining, at around the shortest

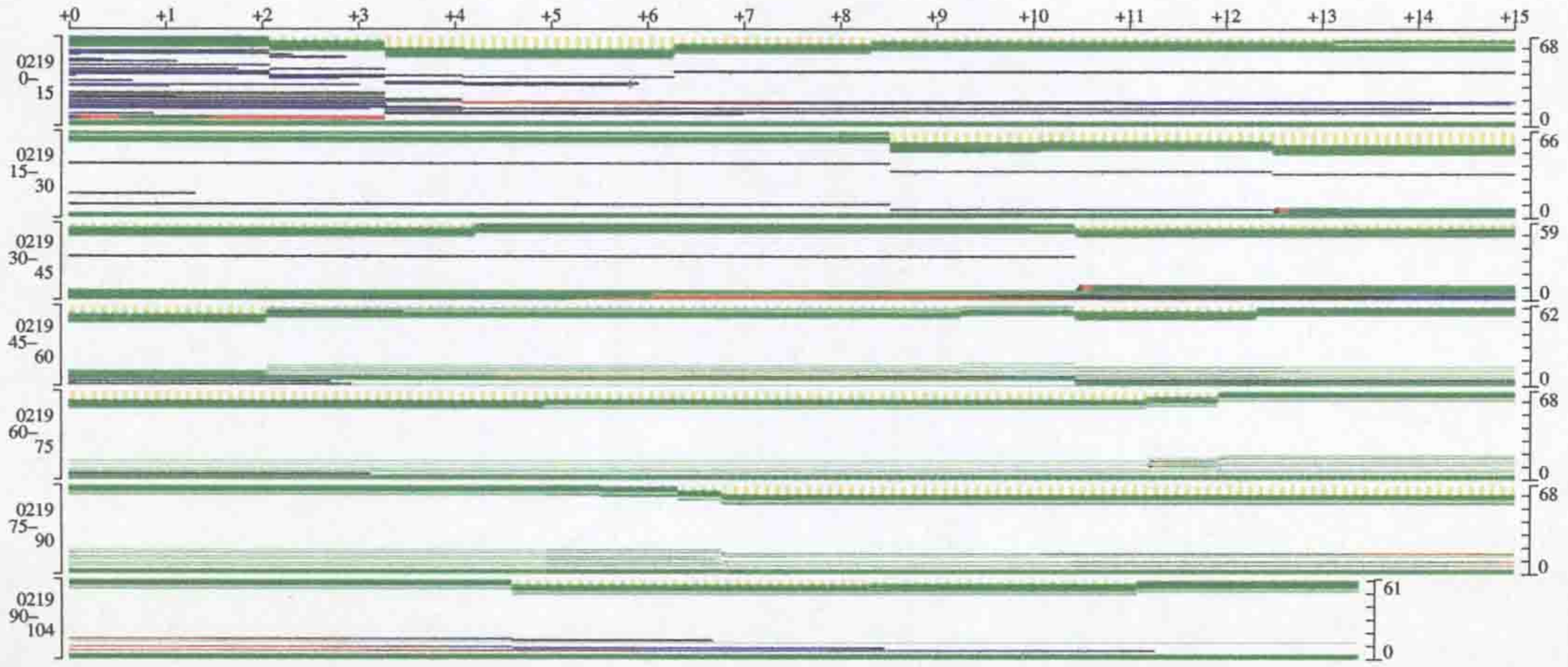


Figure 6.15: Gene expression of cell 0219, showing very some strong gene expression and a lack of initialisation that could only mean this cell is the result of a cell division.

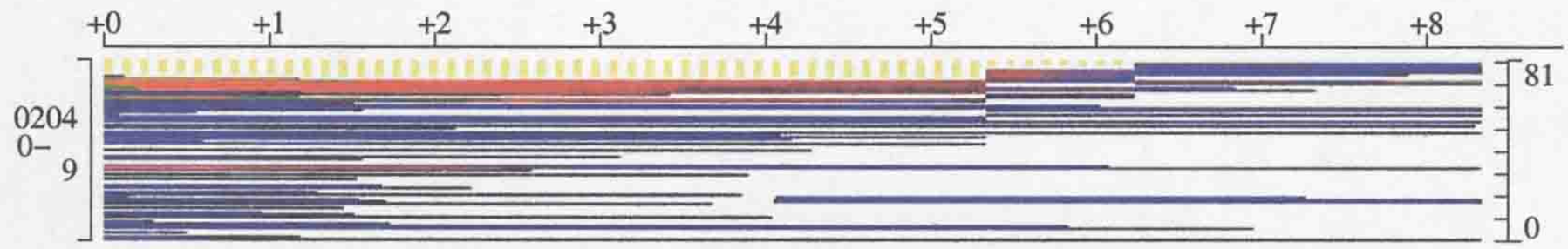


Figure 6.16: Gene expression of cell 0204, demonstrating the quickest cell death possible.

time possible.

6.5.5 Cell 0193

Figures 6.17 and 6.18 show an early success, the cell is clearly well connected and is also showing patterns of transcription across many genes. The number of some genes goes up by many times over a short time frame, this is caused by many of the right sigma factors being by the same gene at the same time. Notice the large changes in genome size caused by sequence insertion and deletion. Also note this cell is the result of a cell division, as there is no visible initialisation phase.

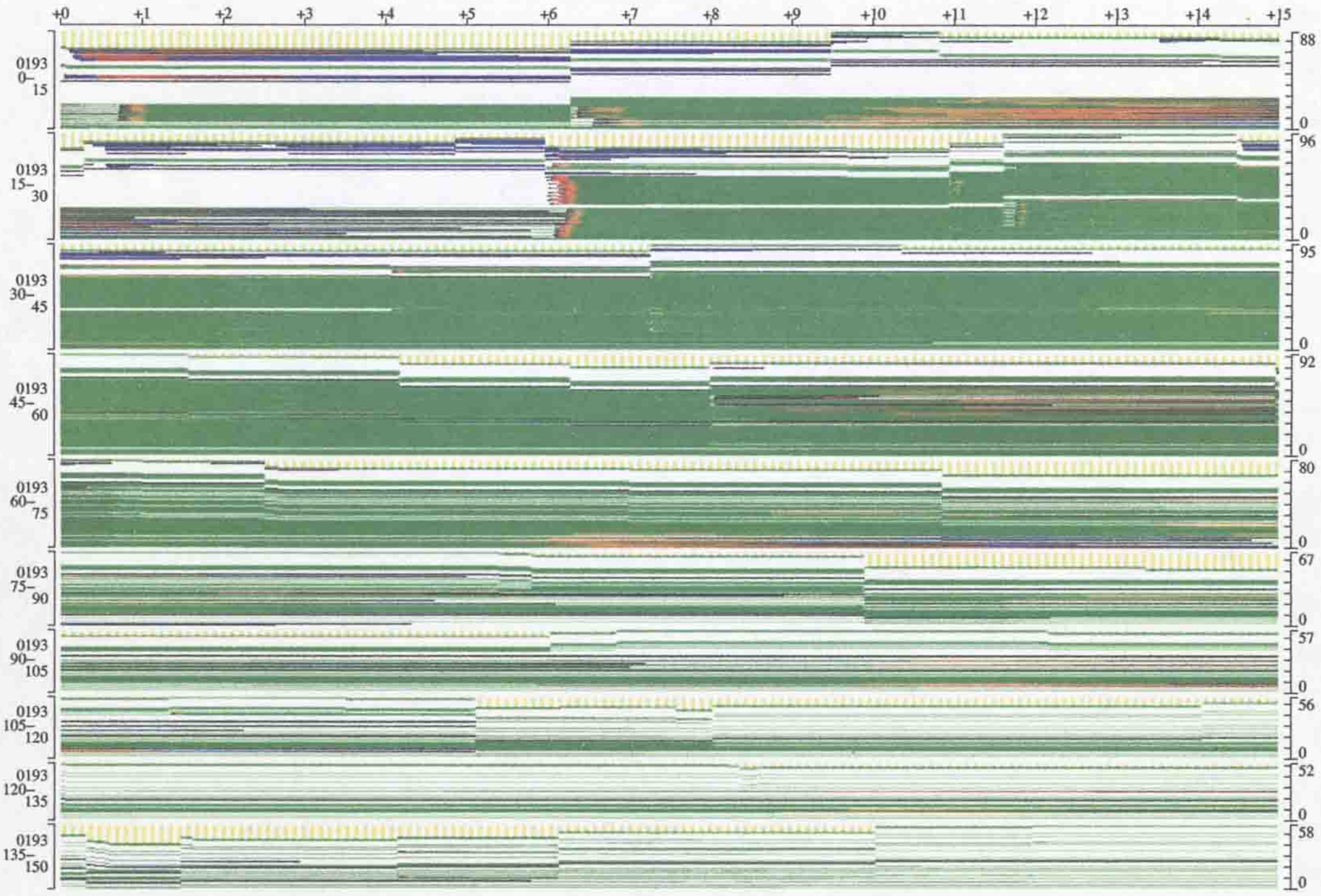


Figure 6.17: Gene expression of cell 0193, an early success compared to the others presented here.

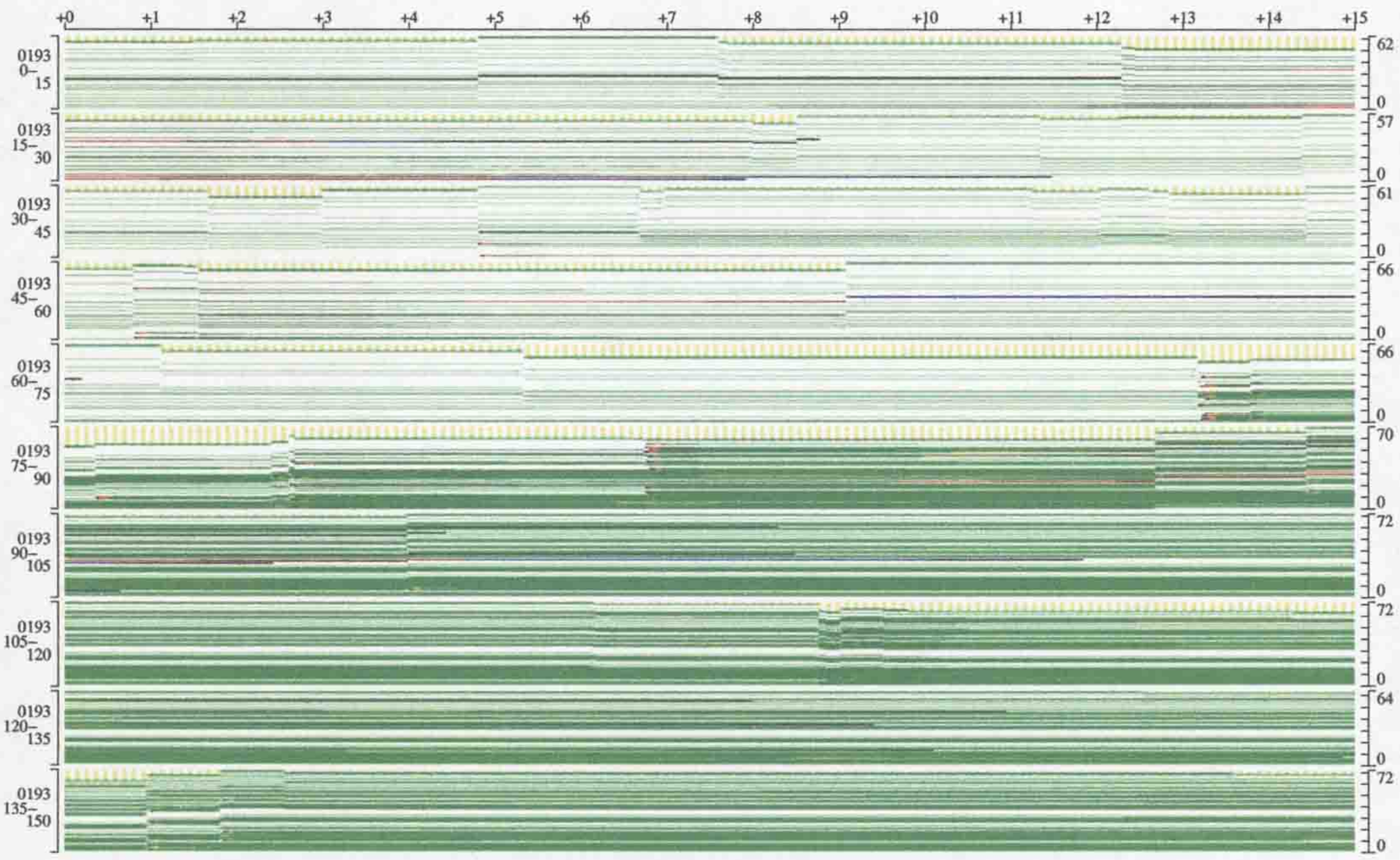


Figure 6.18: Continuing gene expression of cell 0193, an early success compared to the others presented here.

6.6 Gene Expression Pathways

It is possible to create digraphs of gene expression pathways using the linkage diagram shown in figure 4.6 and the gene expression data as shown above. The linkage create the graph structure, after folding common genes, removing dead ends and so on; and the expression data then quantifies the use of each pathway. This can then be divided into slices of time, or as shown in figure 6.19. This example shows cell 0143 in run030205 over the course of its life, this being the cell that parented the lineage which went onto take over the population in run030205.

Nodes represent genes and gene products. Ovals are specifically for genes and gene products from the genome, boxes represent FAP receptors and ovals represent input receptors. In each node are two lines of text, the top line is the gene sequence used in the anti-match function, which is useful for cross referencing. The bottom line is an abbreviated gene type that corresponds to the types in figure 4.4. Edges are numbered with a use count, binding events followed by unbinding events.

The kind of diagram is the starting point for examining the cell at the highest resolution possible. The problem comes from dealing with its size, making reading the graph a time consuming process. It should also be possible to highlight edges or groups of edges and graph time series of changes, as cross referencing to other datasets remains cumbersome. It is also be possible to combine two graphs from different time frames, and generate from these two a difference graph of what has changed. This is ultimately a data mining problem, in that there is no generic method that will take apart this large structure without at the same time destroying its value. As a result this is a research topic in itself, the most obvious starting point being a graphical user interface.

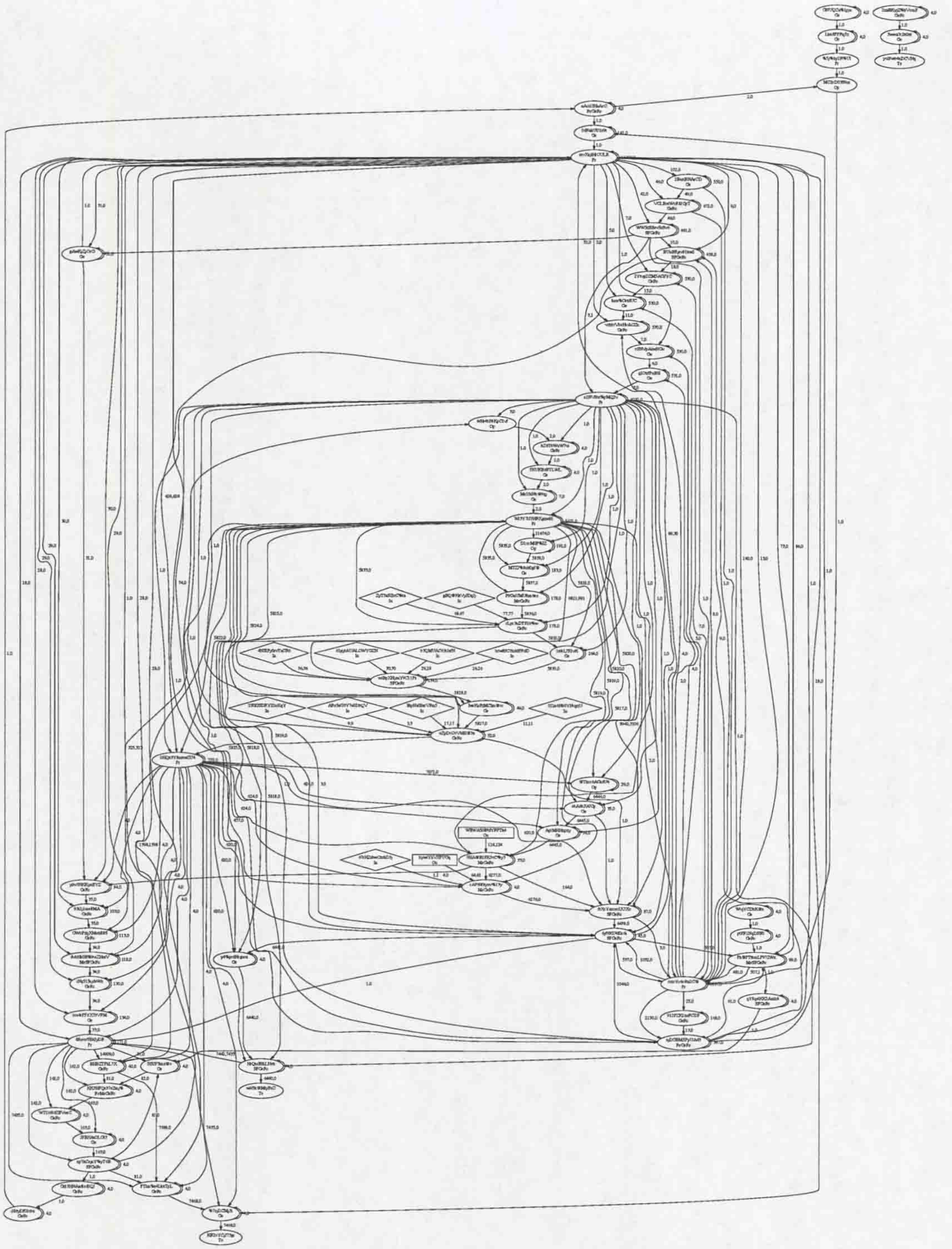


Figure 6.19: Pathway over the life time of cell 0143

6.7 Summary

Here we introduce the most common visualisations used to represent the raw data generated by COSMIC. It would be expected that many of these ideas come directly from their biological counterpart, but as was hinted during the biology background, biology can be severely limited by what can be measured. Available data dictates what visualisations can be constructed with any degree of usefulness and so the legacy of practical biology leaves COSMIC in the position of having to find visualisations for data sets that do not exist in biology. There is also the issue of data set size, which is largely avoided above by giving a few examples of the thousands of possible images.

To give some idea of scale, the visualisations are presented in the order of largest to smallest scale. Starting with environmental view, which consists of pictures of the substrate condition. Onto this image is placed each cell represented by a circle proportional in diameter to the mass of the cell. This image, or rather a series of images then shows which cells are moving and the change of substrate distribution.

As COSMIC records all relationships between the object sets, so to are the cell relationships. This means it is possible to draw a lineage chart in either direction of time rooted at any cell. From this it can be seen what cells are related and by how many generations. Useful where comparing the performance of cells that are genetically related.

There are however situations that require more tradition approaches, this is the case when considering the parameters related to the cell at the highest level. Shown are the important parameters of volume, total enzyme count, position in the environment and the activity of the input and output receptors. When searching for cells of interest, this can be a quick method of accessing the quality of a cell.

At much lower level, COSMIC can plot the expression levels of each gene on the genome over any period of time while taking account of sequence insertion and deletion events. Thess diagrams clearly show which areas of the genome are responsible for the success of a cell. Importantly they also give clues as to

the failure of cells, as cells fail for many reasons.

Changing dimensionality, this data can also be shown as a digraph. This then indicates which genes interacted with which other genes and how many times, over any given time frame. This gives the most detailed view but also the biggest challenges in terms of visualisation, as one static graph only represents the surface detail of the actual interactions.

Chapter 7

Results

7.1 Introduction

The previous chapter introduced some of the visualisation techniques used in this chapter. Some of those visualisations are research topics in themselves as the data generated by COSMIC is so rich that a single image only scratches the surface of what interactions actually occurred. They were separate from this chapter as a result of this. This chapter instead gives an account of some of the simulations, not just the data obtained but also the evolution of the simulation and its testing.

The key difference between COSMIC and all other simulators available is the implementation of detailed genetic interactions combined with evolutionary concepts, and with this the use of multiple scales in the same simulation. The behaviours of the system as a whole and the individuals are then very spread out, the size of the system makes it very much akin to examining a real world system, except that this system is different in nature and so needs different approaches from those learned over the years. This system gives the detailed genetic data that real world processes must approximate, but on the other hand offers no clue as to how real world methods can be used to read this data.

The results from COSMIC are difficult to classify, the whole simulation tries to be integrative with no clear boundaries between interactions that occur

inside the cell, obviously the implementation is still modular but the parts are so interconnected it can be difficult to see effects of the parts. That said, the approach here is give an overview of some of the simulations. The open ended nature of COSMIC ultimately meant there was no single result that showed evolution taking place. The single result was a simulation system that should have the ability to show evolution taking place. There has been data suggesting there is a slight overall improvement in fitness, but it is unknown whether this is a reliable indicator given that this data possibly a biased average.

Before experiments are described, the overall parameters will be explained in section 7.2. These provide overall control of the system by specifying limits to the environment, cell growth, cell division, cell genome size, genome mutation rates, enzyme half lives and genome-proteome interaction rates. Some of these are changed to vary the experiment and will be listed separately.

Section 7.3 introduces the data sets that make up the archived COSMIC output. The testing phase of the simulation is then described in sections 7.4 to 7.14. When simulation runs where made, problems were found and corrected and the bulk of the chapter is made up of those experiences. This provides some idea of the subtle effects of programming errors and more often, simple unforeseen consequences of some implementation decision. This chapter finishes with a summary of the main outcomes in section 7.15.

7.2 Parameters

Note. Each parameter is preceded by the C++ source file that it came from and the line number.

`cell.C:266://options.OperatorGene=options.OperatorLink = 0;`
Controls the inclusion/exclusion of operators when a genome is randomly initialised. Here for testing the implementation and showing operators do have an effect. Operators were tested long before simulations were recorded (there would never have been enough room) and so those returns are not shown, I'd like to do some more tests to show this. There are two controls, *Gene (i.e. any

control ending in Gene) enables the creation of genes of this type, and *Link enables the operators in the genetic network (Repressor-Operator interaction), without which operator genes would just be extremely short introns.

`cell.C:267:options.PromoterGene = options.PromoterLink = 1;`
Controls the inclusion/exclusion of promoters when a genome is randomly initialised, same as for operators.

`cell.C:268:options.InducerGene = options.InducerLink = 1;`
Controls the inclusion/exclusion of inducers (anti-repressors) when a genome is randomly initialised, same as for operators.

```
cell.C:269:options.AttenuatorLink = 1;
           options.AttenuatorGene = 1;
```
Controls the inclusion/exclusion of attenuators when a genome is randomly initialised, same as for operators.

```
cell.C:270:options.InputGene = options.InputLink  = 0;
cell.C:270:options.OutputGene = options.OutputLink = 0;
```
Controls the inclusion/exclusion of input receptors and output receptors when a genome is randomly initialised, same as for operators.

`Cell.C:271:options.CreateCytoEnzymes=1;`
Turns on the creation of the proteins based on the genome of the newly initialised cell, these being necessary to bootstrap the cell which would otherwise have no active proteins acting on the gene network with which to create new proteins. The number of proteins created per type is set elsewhere, normally 5 proteins for all expressible genes and 1 receptor per input and output.

`cell.C:271:options.EnzymeDeath = 1;`
Turns off the half-lives of all enzymes, ensuring that all enzymes live forever.

This is largely a debugging switch as the system soon becomes clogged with enzymes.

`cell.C:272:options.EnzymeBinding = 1;`

Turns on all gene-protein and protein-protein binding, leaving only initialisation of the cells. This is largely a debugging switch that allows testing of initialisation knowing that the systems dynamics will not be activated.

`cell.c:272:options.IterateCytoplasm = 0;`

Disables entirely effects of dynamics by not checking for possible protein binding and not taken action on protein unbinding. This is even more far reaching than the above.

`cell.C:273:options.ConsumeSubstrate = 0;`

Stops the cells from consuming substrate, the environment is not altered and the cells gain no glucose. Defaults to True. Note this is on a per cell basis, a similar flag for the environment sets the entire simulation.

`cell.C:273:options.KillUnviable = 0;`

True enables heuristics intended to speed up evolution by killing cells considered unviable. That is cells with less than `options.viable_generat...` enzyme population to transcribable genes, negative cell mass or has made no recent response to or sensing of the environment.

```
cell.C:278:options.insertion_rate = 0.0025;
cell.C:279:options.insertion_shape = -0.01;
cell.C:280:options.insertion_max = 0.1;
cell.C:283:options.deletion_rate = 0.0025;
cell.C:284:options.deletion_shape = -0.01;
cell.C:285:options.deletion_max = 0.1;
```

Gene sequence insertion/deletion parameters controlling the frequency and size

of these events. Normally set to a relatively high rate but can also be turned off using these same controls.

```
cell.C:929:pd_geneinsertion=new ProbDist(0.0,1.0);
cell.C:939:pd_genedeletion=new ProbDist(0.0, 1.0);
```

Both these functions generate uniform distributions which are then used to test if insertion/deletion should occur and if so what position and size. p here is then tested or multiplied by the relevant scalars, this is an implementation issue that removes the need for many distribution generators.

```
cell.C:289:options.longterm_in[CellOpt::Inflate]=0.002;
cell.C:290:options.longterm_in[CellOpt::Minimum]=-1.0;
cell.C:291:options.longterm_in[CellOpt::Maximum]=1.0;
cell.C:292:options.longterm_out[CellOpt::Inflate]=0.001;
cell.C:293:options.longterm_out[CellOpt::Minimum]=-1.0;
cell.C:294:options.longterm_out[CellOpt::Maximum]=1.0;
```

Parameters defining the above cell viability check heuristic for connectivity with environment. Allows for a peried of no activity at all, by setting a counter to `Minimum` if there is activity and always incrementing the counter by `Inflate`. If the counter reaches `Maximum` then the heuristic considers the cell not viable.

```
cell.C:297:options.viable_generatio = 0.05;
```

This figure multiplied by the genome size gives the minimum number of gene products, if the cell should have fewer then it is considered not viable.

```
cell.C:970:pd_genematch=new ProbDist(0.015,0.015);
```

When finding gene-gene interaction paths the hamming distance was originally taken as potentially probabilistic, or more precisely, fuzzy. Hamming distances in this region (here 0 as this feature is unused) would be compared against values from this distribution. Thereby making some pathways probabilistic when initialised. This was felt to be create an implementation bias and so was

not used.

```
cell.C:980:pd_potbindunbind=new ProbDist(0.0, 1.0);
```

Distribution generated for the potency coefficient and binding and unbinding times. For all instances of e^x in molecular interactions. Uniform as e^x returns the required function.

```
cell.C:990:pd_randomqueuing=new ProbDist(0.0, 1.0);
```

The implementation services events in a well defined way that is static over the course of a single cells execution and passed onto daughter cells. This ordering is initially defined by this random distribution. All possible pairs of pathway interactions are sorted on this random number to ensure no biases from the creation of the interaction pathway set.

```
cell.C:999:pd_genomelength=new ProbDist(70.0,130.0);
```

Newly created cells with no parent have a genome size (including control sequences) defined by this uniform distribution.

```
cell.C:1009:pd_genetype=new ProbDist(0, 4);
cell.C:   1012:float arr_pd_genetype[]={0.33, 1.0, 1.0, 0.5, 8.0};
cell.C:   1019:pd_genelength=new ProbDist(10,15);
cell.C:   1029:pd_locusvalues=new ProbDist(0,env->numloci-1);
```

Genome generation pdfs, the first selects one of the 5 gene types (4 control, 1 gene product) and the associated array then gives a distribution of the relative frequencies. That is, promoter, operator, attenuator, terminator and gene product. Once type is known, `pd_genelength` determines the length of that individual gene and then `pd_locus_value` generates the letters within the alphabet. These are then used in the hamming distance function to compute interaction paths.

```
cell.C:1965:int nomi_geneprod=5;
```

Cell bootstrapping requires gene products be created somewhere. Under steady state conditions gene products create a newer generation of gene products, the initial state is to generate `nomi_geneprod` gene products with the creation of a new random cell to start the transcription process.

`cell.C:1966:int nomi_glucose=1;`

`cell.C:1967:int nomi_flagel=1;`

Input receptors and flagella receptors are internally modelled as timeless gene products of static genomes. There is the option for multiple substrate receptors and multiple flagella receptors in the same physical space. This was never used in practice though could be useful to simulate larger enzyme populations without increasing search space, as each resultant gene product has the same encoding.

`cellstate.C:858:int partialdivide=TRUE;`

Boolean indicating the realism of the cell division process. Normally the enzyme population of the parent cell is divided equally among the daughters, False indicates that both daughters should receive all the enzymes the parent had. This is a physical impossibility but useful for debugging, given the same enzyme population both cells should perform similarly well and so is useful for testing.

`enviro.C:98:options.RefreshSubstrate = 1;`

Boolean indicating the environment glucose level should be increased every iteration to ensure the environment contains some new glucose source. Without this the environment is emptied of glucose, making growth of cells impossible.

`enviro.C:99:options.KillUnviable = 1;`

Simulation wide flag enabling the use of the cell viability heuristic. For viability to be checked, both this and the cell option must be True. The two options give the possibility to give some cells move chance at survival, by making it

easier to to implement exceptions to the rule.

`enviro.C:100:options.RecordSubstrate = 1;`

Boolean controlling the saving to hard disk of environmental pictures. True generates a jpeg of the current environment every 10 course grained iterations (10 seconds), saved with a filename based on the simulation time.

`enviro.C:110:pd_cellposition=new ProbDist(0.0,0.0002);`

When creating a new cell the initial cell position is obtained from this pdf. It can then be placed anywhere inside the 200μm square that is the environment. This can then also be used to place cells in a corner of the environment and so easily see which cells have been motile from a single view of the environment.

`enviro.C:127:pd_volumelvl=new ProbDist(0.2*F_FEMTO,0.4*F_FEMTO);`

When creating a new cell the cell volume is obtained from this pdf, placing the cell inside the bounds of normal steady growth between 0.2 and 0.4 femtolitres in volume. This potentially places cells within easy reach of division immediately after creation but does not give any real advantage to the cell.

`enviro.C:160:pd_ioposition=new ProbDist(-0.000002,0.000002);`

Receptors on the cell wall must be given a position, this pdf determines receptor positions relative to the cell for all cells - this same physical distribution is given to all cells to reduce search space and increase the chance of viability across generations.

`enviro.C:344:sim_time_delta=0.02;`

`enviro.C:346:secondsperround=1.0;`

Fined grained and course grained iteration times, `sim_time_delta` increments simulation time for each fine grained intra cell iteration. On reaching `secondsperround` the process synchronises with the environment. As a result, these parameters are simulation wide and set the possible resolution on the one hand

and sensitivity to cluster network congestion and load balancing on the other hand.

```
enviro.C:356:transcription_rate=0.025;
```

The RNA polymerase doing the transcribing moves at a specific speed, here it is specific in genes per second, adjusted to account for the size difference between COSMIC genes and real genes. The end result is transcription at approximately the correct speed.

```
enviro.C:362:transcription_matter=0;
```

A potential cost in terms of matter was placed on transcription to provide selection pressure against futile cycles. Calibration was a problem and so this constraint was never used.

```
enviro.C:367:chemotaxis_rate=0.000025/8*secondsperround;
```

Chemotaxis is computed as a vector, combining all the forces of each flagella. The maximum swimming speed is known to be 25 uM per second and this maximum is taken as the maximum the vector can total to, hence the division by 8 flagella. The real mechanism of chemotaxis [CP97] is chemical and so beyond the scope of COSMIC.

```
enviro.C:371:f_maxgrowth_rate=0.0205/0.4444;
enviro.C:372:f_maxyield_ratio=0.4444;
enviro.C:373:f_saturation_const=0.00234;
enviro.C:376:f_maintainance_rate=5.791e-15;
enviro.C:379:f_volume2drymass=290;
```

Cell growth parameters based on [KBW98, NTT96, KW82]. These reduce cell growth over time to a function based on available glucose substrate. As a result they fit exactly with the requirement of COSMIC.

```
enviro.C:382:chemotaxis_rate*=1;
```

Reducing this value gives the opportunity to increase the effective surface area of the environment without also having to increase the resolution. Increasing the resolution is best avoided as modifying the environment involves subtracting a floating point number from a 2 dimensional array of floating point numbers, this will take a long time with a high resolution.

```
enviro.C:393:          matrixenergy_rate=1e-6*secondsperround;
```

This parameter sets the glucose replenishment rate to ensure the environment contains some glucose. This ultimately sets the size of exponential population growth, higher values ensuring the population can grow further before lack of nutrients limits growth. The exponential growth is mostly created by the initial environment, as this value would need to increase exponentially as the population explodes.

```
enviro.C:400:map_glucose.init_uniform( 0.0002, 0.0002, 2500000,
                                       0.0, 0.0045, 0.0045 ) )
```

These group of parameters define the dimensions in meters, resolution in pixels per meter and glucose level in terms of minimum possible, maximum possible and initial value, all in nanograms/litre. This amount to initialising the environment to 500x500 pixels with absolute limits of 0.0 to 200 uM with bounded values of 0.0 to 0.0045 and a value of 0.0045 nanagrames/litre all over.

```
enzymerates.C:25:#define HALF 0.0058
```

Gene product half life is defined by this value and by the follow tables. This sets the half life to around 2 minutes. It is known that gene product half life varies across gene product species, some lasting hours, some lasting minutes. For simulation purposes a brief half-life was chosen to reduce delays in the pathway cascade, making reaction to the environment potentially faster and so consuming less computing power. The more long lived gene products are of no interest to this simulation, for the most part they are not involved in rapid response cascades.

Three tables describe the reaction rates for the set of interactions, the above HALF is used by most of them, some have been reduced. Rather than reproduce them here, these are the same matrices as found in figures 4.2, 4.8.2 and 4.8.3.

`world.C:170:createinitialcells(environ, 20 ) )`

This parameter sets the initial population size. This is largely the result of the implementation generating one line of output per iteration per cell, so was large enough for testing but small enough to easily fit all the cell output in one small terminal window. There has never been any reason to change this figure as it also fitted well with the number of CPUs in the cluster.

`world.C:216:while( environ->ms_cells.size()<20 )`

This parameter sets the minimum population size, which is essential for the initial searching phase that looks for a viable cell. For the same reasons as the parameter above, this figure was chosen for ease of use and never changed.

`world.C:415:if(((long long)rint(environ->sim_time) % 2000)!=0)`

This parameter sets the state saving interval of the entire simulation. Should COSMIC fail due to a failed node in the cluster, the simulation can be restarted from any one of these save points.

`world.C:555:} while( environ->sim_time<500000.0`

Finally, this parameter sets the total maximum run time in simulated seconds. Though it is not required that this time actually be reached.

7.3 The Data Sets

There have been at least 10 experiments testing the environment, population growth, survival times and parallelisation. These simulations were originally based solely on the one machine. Later the simulation was run on the Computer Science farm, and starting with runs021102 the simulation used a

	Duration(Days)			Cells		Data size	
Dataset	Real	Adjusted	Simulated	Total	Peak	(Gbytes)	Focus
run020501	16	?	2.43		55		Increase env. replenishment rate
run020516		?	0.94		48		Increased cell division diversion angle
run020602	7	3	0.49	1745	165		Individually set PDFs per cell
run020610	8	4	0.56	1914	140	7.7	
run020623	5	2	0.23	1437	310	1.9	Test for exponential growth
run020813	7	3	0.44	1641	155	5.8	Test cause of growth synchrony 1
run020820	3	1	0.23	1948	475	2.2	Test cause of growth synchrony 2
run020905	21	10	0.42	1485	165	4.7	Compare direction against previous
run021102	4	4	0.55	2164	180	11	Run on the dedicated cluster
run030116	21	21	1.11	3804	340	8.5	Bugged printf() of 030205, jiffies
run030205	35	35	1.49	6074	420	93	Stress test and parallel results

Figure 7.1: Simulation runs archived for later analysis.

dedicated cluster. These simulation runs are tabulated in figure 7.3. There was also a large number of experiments before there was any useful recording, these were to test the interaction paths and made use of what are now the control variables above.

As each run was made, errors were found and the simulation evolved, removing artefacts of the simulation process and some more standard errors. As can be seen from the table above, all runs took a significant amount of time to compute, usually to the point that the sought after answer was known. Before hand there were many many more runs but constraints on storage space mean that only the more recent and therefore the most error free remain.

Simulated time is a function of cell complexity, number of cells and obviously total simulation time. Short simulations seem more effective than long simulations, this is simply a result of longer running simulations containing more cells, all simulation runs go through a phase of rapid simulation, but once viable cells are found the pace slows rapidly. It should be noted that a change in the cluster hardware significantly changed the ratio of real and simulated time, on the order of 1.8 per machine and with more machines. The real simulation time for runs before the large cluster have been reduced to better reflect the time it would have taken on the large cluster.

Simulated time was overall a little disappointing, in a completely realistic environment there is little opportunity for cells to evolve in such a short time frame. Fortunately this was expected and provision to better guide the

evolution process was made early on. Parallelisation in chapter 5 is part of the answer, using viability heuristics, limiting cell numbers and putting costs on cell interactions and activities are other avenues. The latter two were not used, all references described the cost of interactions as negligible. Limiting cell numbers seems too artificial, with or without some kind of elitist strategy. The two most important qualities were parallelisation, and the viability heuristic. Limiting cell numbers would have a confounding effect on interpreting results and so it was felt better to allow the simulation to slow and wait longer for reliable results.

The first two data sets show rapid simulated time for short simulation duration, this is a direct result of the cells never reaching an exponential state, for the most part there were 20 cells at any one time over the course of these two simulation. With the problem isolated and corrected all other data sets went exponential at some stage and so have much lower total simulated times.

Totalling the cells produced by each simulation gives some idea of simulation size and cell lineage success against whatever restrictions imposed by the environment. Each division process creates two daughters, and initialisation of a new cell to maintain a population of 20 creates one cell; this value is the total of both. For all available figures new cells account for around 150 of that total, cell division was the main mechanism for growth and so there were major cell lineages all containing similar genetic codes.

Initial simulation success can be seen from peak cell populations, before success can be measured in terms of finding glucose gradients it must be measured in terms of growth. These figures and more convincingly the charts demonstrate this success in a limiting environment. The high peak of `run020820` came about with an unlimited environment to test for exponential growth.

Dataset size is given here as a guide to the shear size of the simulation, the individual approach creates this potential problem. Fortunately this raw data can be mined using a set a scripts to produce more focused data. It might be expected that size correlates with total cell numbers, as the data is compressed there will be variation in size actually used simply through some data sets being

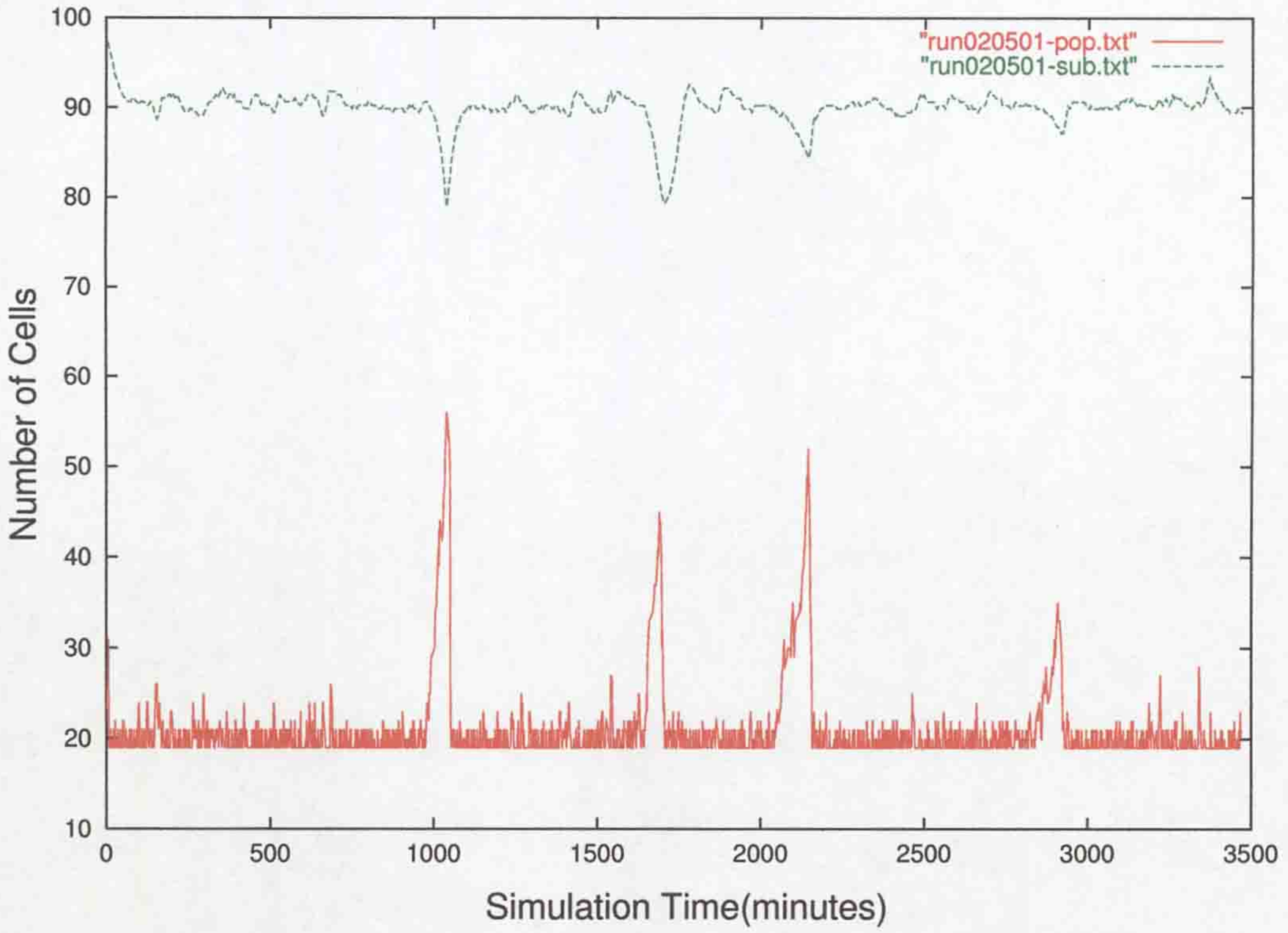


Figure 7.2: First archived simulation result. The vertical axis represents the total number of cells in the simulation, the horizontal axis represents time in simulation seconds.

more compressible, maybe the most popular cell line was also quite active in some simulations. For instance, `run020205` compressed 546Gbyte to 93Gbyte, whereas `run021102` compressed 65Gbyte to 10Gbyte. Even with these large averages there was still a significant variance in compression ratios.

7.4 Simulation run020501

This section covers the finding of the first simulation that was recorded and archived. It was run over the period 01/05/02 -16/05/02 and is shown in figure 7.2. It was planned to run for a few days but was left running longer after some cells looked promising. Note the naming scheme of this section (and those that follow) is based on the date the simulation was executed, as described in section 5.7.

The main focus of this simulation was to increase the environment replenishment rate from 0 to fast, as previously all cells died off. The result of this was that the environment is replenished fast enough to keep up, but not so fast that a cell can stay in the same place. However, the cells still don't proliferate, they divide for a short while but even better genomes always die out. The parameters used for this simulation were the same as above, but for the addition of energy parameters that were later removed. The next step was then to try splitting enzyme numbers 100/100 rather than 50/50 when a cell divides.

Looking over the resultant data, such as some successful cells (0357, 0415), it seems they fail because the cytoplasm division takes away important proteins that don't exist in the new cell in enough numbers to sustain it.

Cell 0357 leads to 7 cells, but at its height (time window 19420-13815) with 6 cells existing over 1815 iterations, they then all die out with 29 iterations. Cell 0357 had regular connections with the environment until some time after the last division, when it lost connection with both inputs and outputs and the enzyme population went into rapid decline only limited by enzyme half life, the cell was killed through lack of enzyme population. 0386 is 357's first daughter that lasts 1820 iterations before dying, the status shows the enzyme population recovered slightly but then faded away in two falls, at the end of the first fall the connection to the environment was lost, by the end of the second fall there were no enzymes remaining.

0411 is the daughter after 0386 and was among the last of this line to die off as well as having two daughter of its own. The general status graph shows the cause of death identical to 0357, at a point some 1000 iterations after division an enzyme population of 5000 enzymes declines at the maximum rate to 0, at the start of the decline the connection to the environment is lost. Cell 0411 confirms division timing, the cell is clearly moving at high speed and so is growing at approximately maximum rate. Division occurs at 1500 iterations, or 25 minutes.

In the 0415 case, the cell population is sustained but the connection to the

input and later the output is lost, forcing COSMIC to kill it. 0464 (descendant of 0415) died for the same reason, though the protein population was diving regardless. The cell halved and shortly after it lost its connection to input receptors, then shortly after to lost any connection to the output receptors. Cell 0481 (descendant of 0464) was identical to 0464.

Cell 0143 was very successful, though had for some reason gone unnoticed until much later. This one cell lead to 12 offspring over a maximum generation depth of 6. This result pointed out the need for more checking, to find out why it proved so successful but then died. In light of p.d.f. problem and the divide direction problem, it is presumed that this one cell was very inactive both in motility and transcription, making it divide quickly enough to survive the cell division process.

7.5 Simulation run020516

This section covers the finding of the partially recorded simulation, made over the period 16/05/02 -22/05/02. The main focus of this simulation was to increase the cell divergence angle of divided cells as a test if it improves life time. The divergence angle comes from a cell division, cells are initialised with a random orientation but at cell division the new cell must be given a new direction. This is based on the parent orientation, the parent orientation is incremented by this value and the new cell is set to the parent orientation minus this angle. Also, both cells offspring receive 100% of the parent cells contents. The parameters used for this simulation were the same as above, but *partialdivide* flag being set to false.

As shown in figure 7.3 this had no effect, suggesting the problem is elsewhere. Without further analysis of the resultant data set, it was noticed that all daughter receive the same PDF set and initial seed as their parents. Given the same initial conditions and the same random events the daughter cell is destined to follow the exact same execution path as the parent. This does not entirely explain the failure as good parents should lead to good daughter and

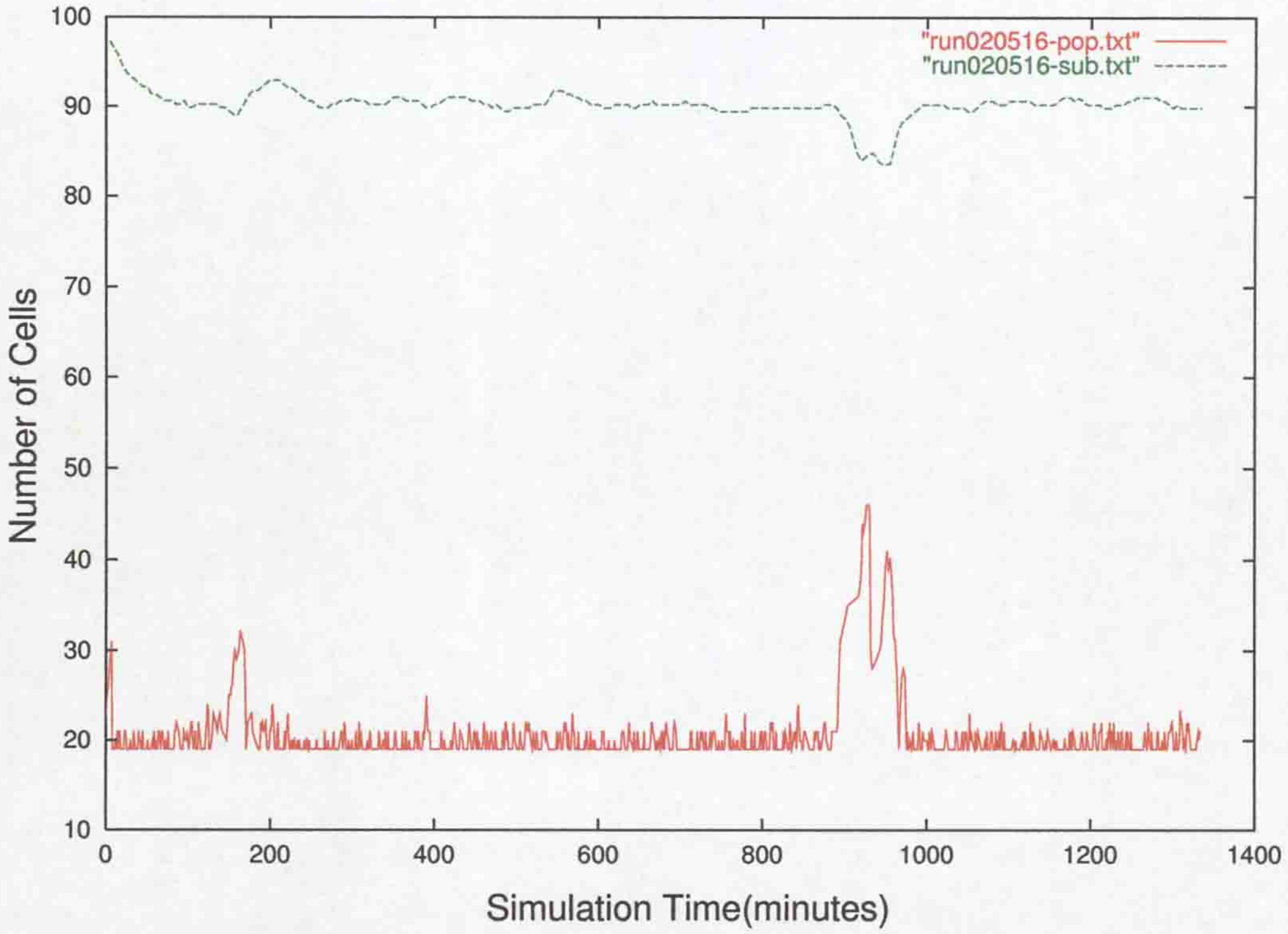


Figure 7.3: Early failed simulation, parental PDFs were passed in error to the daughter and so made the daughter cell effectively the same. The vertical axis represents the total number of cells in the simulation, the horizontal axis represents time in simulation seconds.

so an overall growth, albeit at a reduced growth rate.

7.6 Simulation run020602

The need for further testing lead to run020602, which is the first simulation in which all the data is available even if some of that data used an output format no longer supported by the COSMIC data mining utilities, which generate the various graphs and pictures. It took 3 attempts to run this simulation, finally covering the dates 06/06/02-10/06/02. The main focus here was to give each cell its own seed value and random number generator, the seed is based on the 8 digit cell id (which is here shown as 4 digits for brevity) and so is both random for each cell but deterministic should the simulation be run again with

Cell processes	Machine name
15	babbage.eeng.liv.ac.uk
9	linux19.csc.liv.ac.uk
6	linux24.csc.liv.ac.uk
10	linux28.csc.liv.ac.uk
14	linux26.csc.liv.ac.uk
14	linux25.csc.liv.ac.uk
10	linux18.csc.liv.ac.uk
9	linux17.csc.liv.ac.uk
8	linux16.csc.liv.ac.uk
12	linux11.csc.liv.ac.uk
12	linux10.csc.liv.ac.uk
12	linux08.csc.liv.ac.uk
13	linux04.csc.liv.ac.uk
10	linux02.csc.liv.ac.uk

Figure 7.4: Total number of cells per machine when simulation stopped.

the same start conditions.

The initial problem was one of file descriptors as discussed in chapter 5, brought about by running the simulation on a parallel machine for the first time. As the parallel machine was in fact a loosely coupled cluster of student access machines, the file descriptor problem was much worse than it would otherwise as been. Of course, the alternative viewpoint was it being a good test of the COSMIC implementation.

No parameters were changed as this was essentially a bug fix. Both parent and daughter cell still receive a full set of enzymes as in the previous run, i.e. Gene products are not shared between them but copied to the new daughter cell.

While running the simulation for the third time, the host machine ran out of hard drive space. Storage space was then recovered but the result was taken as it stood. This run stopped on iteration 42150 with 154 cells, cell density per machine is given in figure 7.6. Notice cell load is relatively evenly distributed among the machines.

The overall result then is a successful simulation, having found a cell that

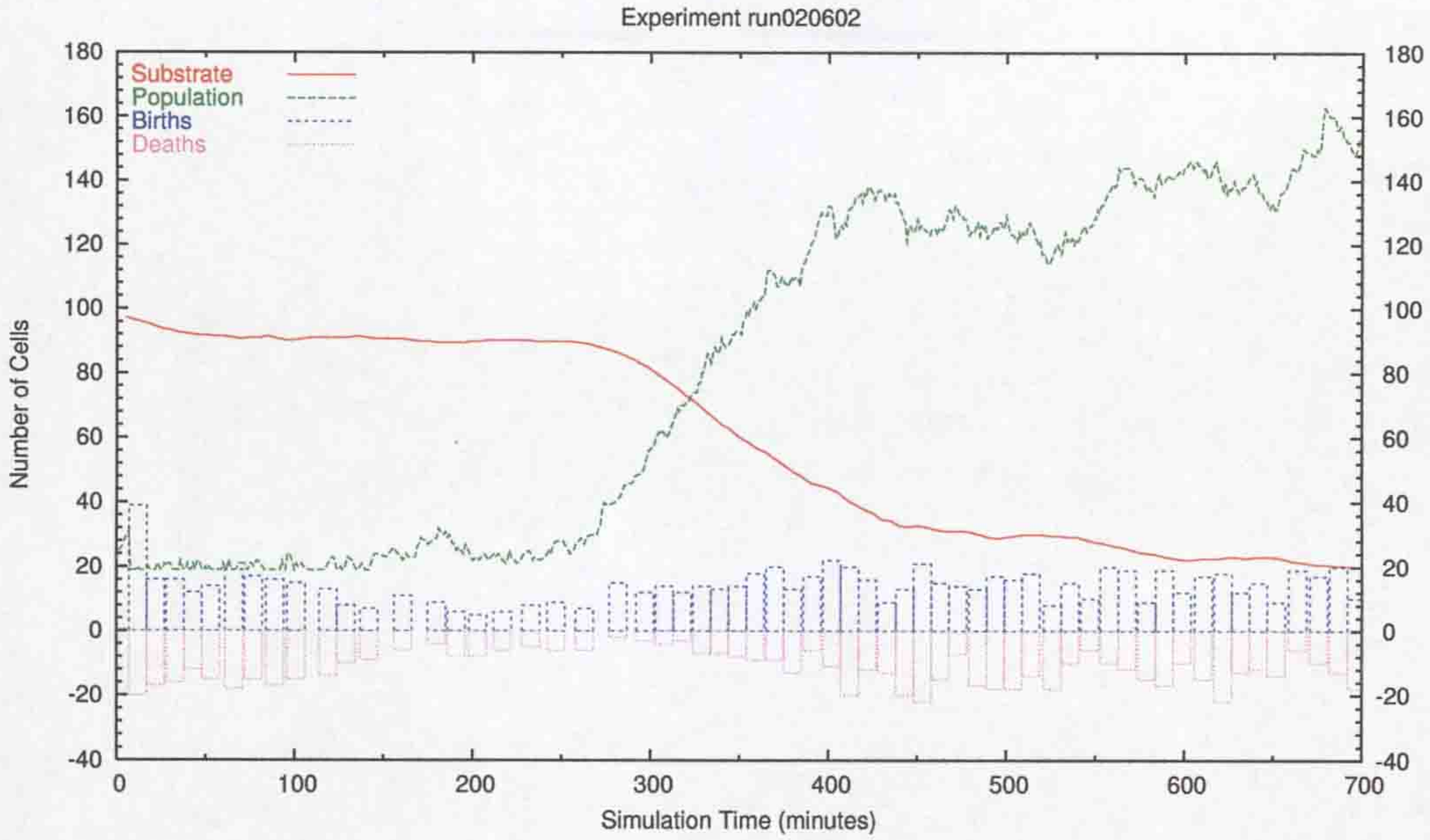


Figure 7.5: First successful simulation showing cell growth and division. This simulation was also the first multi-machine parallel simulation.

fits the initial viability criteria and reproduced that cell to produce a large lineage that takes over the environment and creates a self induced limit on growth rate. As shown in figure 7.5 and unlike figure 7.3 we can see a characteristic exponential growth curve followed by a slowing down as the glucose was consumed.

This was also a success for the parallel implementation, which showed that it worked well enough to support a simulation of this size at a reasonable efficiency - although quantitative efficiency data was not yet recorded it could be sampled, this and the numbers of cells per machine suggested efficiency can be high at least some of the time.

7.7 Simulation run020610

Following the success of the previous simulation, that is success in terms of the population increasing, the obvious next step was to run the simulation again but with the sharing of enzymes at cell division enabled (*partialdivide=True*).

This was a return to the normal mode of operation and will provide evidence that the PDF initialisation bug was the root cause of cell failure.

The simulation was then allowed to run for 2 hours short of 8 days. Unlike when using the same PDFs, these cells proliferate once a good cell is found. Only around two successful lineages where created and then these took over. It would seem exponential growth does not show as expected, the large effect of each cell on the environment ($*10^6$ increase in glucose usage) has a self limiting effect on the overall growth rate - the net effect looks effectively linear, from a flat search phase, to a linear increase, to a slight boom-bust cycle once the environment reaches saturation.

The difference between this and the previous run can easily be considered a result of the `partialdivide` flag. This run, shown in figure 7.6, has the same linear curve over a longer time frame, 460 minutes to create 115 new cells, previously 145 minutes to create 100 cells. The slower growth rate is the result of a higher failure rate in cell division, caused by a brittle genetic pathway. This can also be seen in earlier runs as brief spikes, these lineages achieve a rapid growth and then an instantaneous death. The effect of division can also be seen in the shape of the population increase, `run020602` is continuously increasing during the linear increase phase yet `run020610` increases overall but sometimes declines.

7.8 Simulation run020623

As the growth rate of the previous simulation was more linear than it should have been, it then seemed prudent to test if and how how quickly exponential growth could be found. In an unconstrained environment we would expect to see perfect exponential growth, even with the effects of brittle genetic pathways. The consume substrate control parameter (*options.ConsumeSubstrate*) was set to false and the simulation was run for three days.

The result was clear exponential growth after the usual searching phase delay. The growth peaked at around 330 cells before the simulation had to be

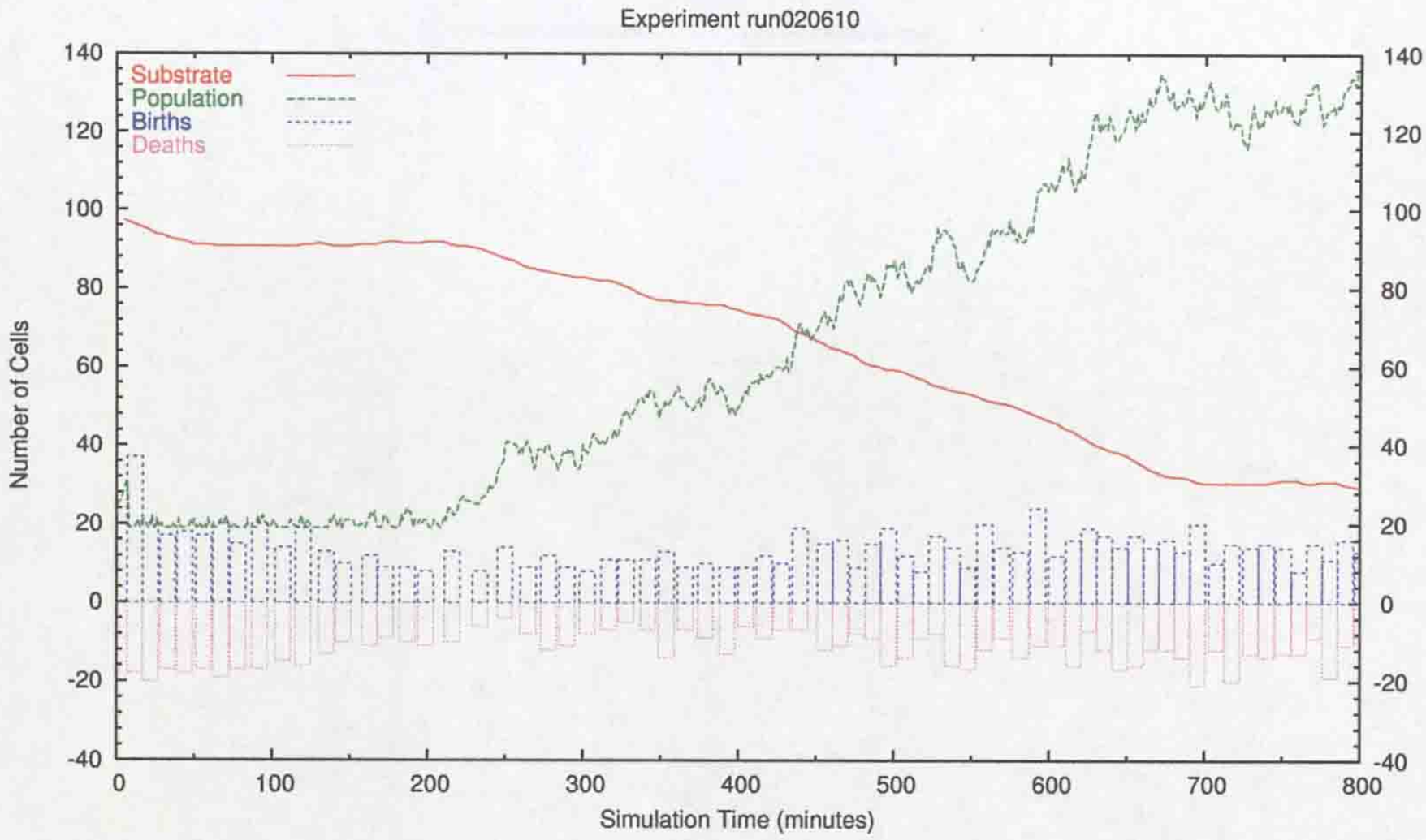


Figure 7.6: Enabling 50:50 cytoplasm sharing at cell division.

stopped - the server machine had run out of process space with there being so many incoming **ssh** connections from the Linux cluster.

Since then some of this process overhead was removed to support around 500 cells with **ssh** and 680 cells with NFS. Regardless of this limitation, 330 cells was enough to see another artefact of the simulation, there were large fluctuations in the population size during the exponential growth as shown in figure 7.7. This can be explained by the synchrony of all cells, since a homogeneous environment makes all cells grow at exactly the same rate. However it can be more thoroughly explained when considering the cell division process. At the end of the simulation there were 3 large lineages, the original cell of each would have its own size, but all cells coming from each parent would from then on be created at the same time, grow at the same rate and so divide at the same time. The end result being three sets of cells, the contents of each set being closely synchronised. What this does not explain is why so many cells die in the first place, it could simply be enzyme splitting bringing out the brittleness of the transcription network.

Looking at the lineages generated, most consisted of only three cells, the

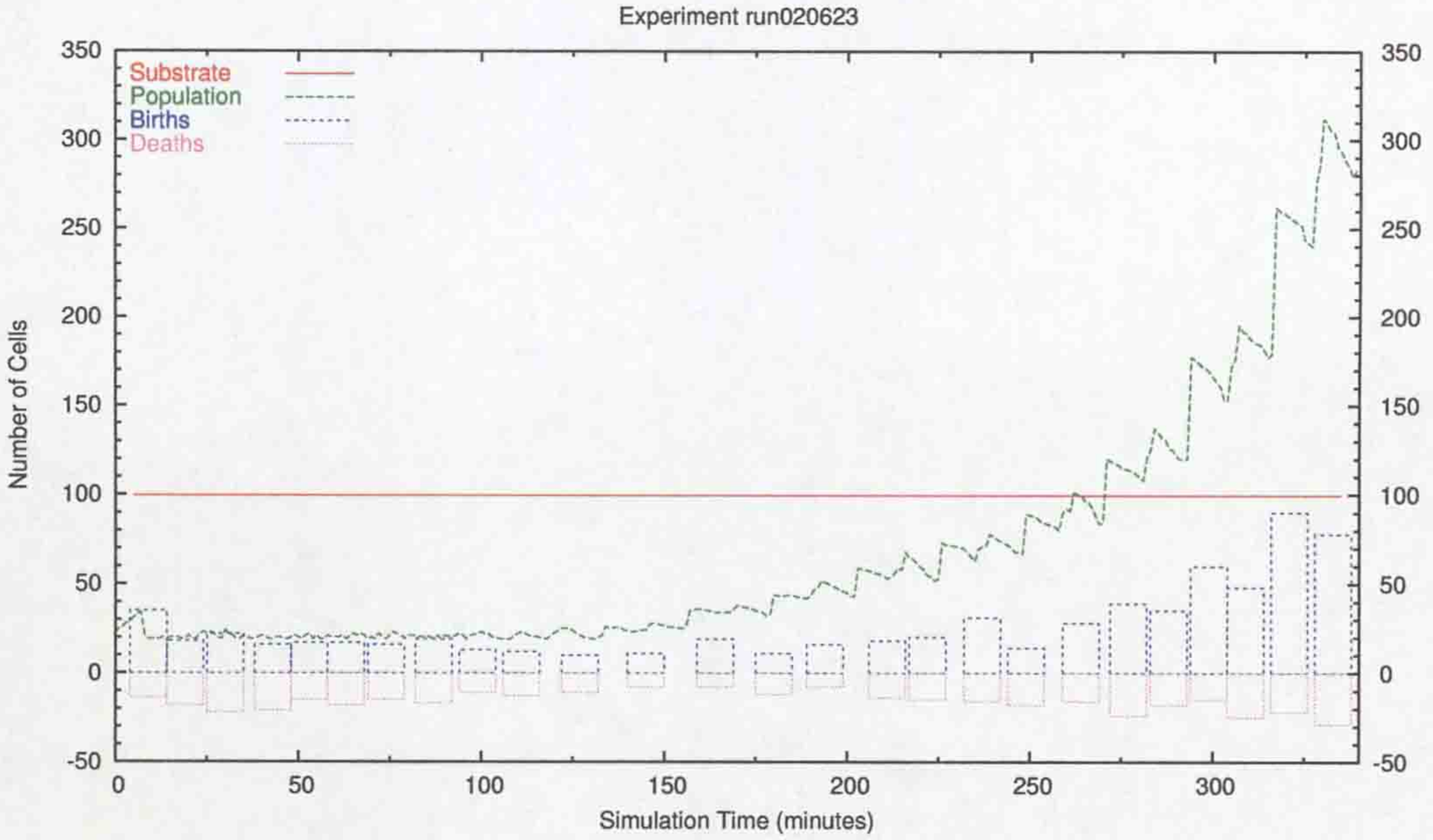


Figure 7.7: Unrestrained exponential growth, highlighting some kind of synchronisation artefact.

original and two daughters that are either siblings or daughters and granddaugthers. The five exceptions that accounted for the vast majority are cell 0211 with 5 daughters, 0160 with 43, 0198 with 117, 0042 with 119 and 0143 with 300.

Looking at the lineage of cell 0143 in figure 7.9, the timing between boom and bust hardly changes as the general size of that lineage increases. Starting at time 9460, with 22 cells and then measuring the time between cycles we see the clear patten of table 7.8.

The lineage of cell 0042 shows a near identical pattern in table 7.8. It starts at time 9068 with 2 cells and so less well established as lineage 0143 but this does not appear to matter. The lineage of cell 0198 in table 7.8 again shows the same pattern. This does however contain more noise as the decline time in the third row is clearly very different to the norm. This lineage started at time 11621 with 14 cells.

But for the initial increase and one case of noise, the period of decline is the same for the whole of the three lineages. As this environment is unconstrained,

Pop. Decline		Pop. Increase	
Time Span	# Cells	Time Span	# Cells
1237	13	134	25
1313	19	58	36
1253	23	118	48
1354	30	16	54
1279	39	92	78
1306	60	69	121
1232	91	135	180
1283	146	-	-

Figure 7.8: Timing differences between multiple peaks and troughs of run020623, cell lineage 0143 during unrestricted exponential growth.

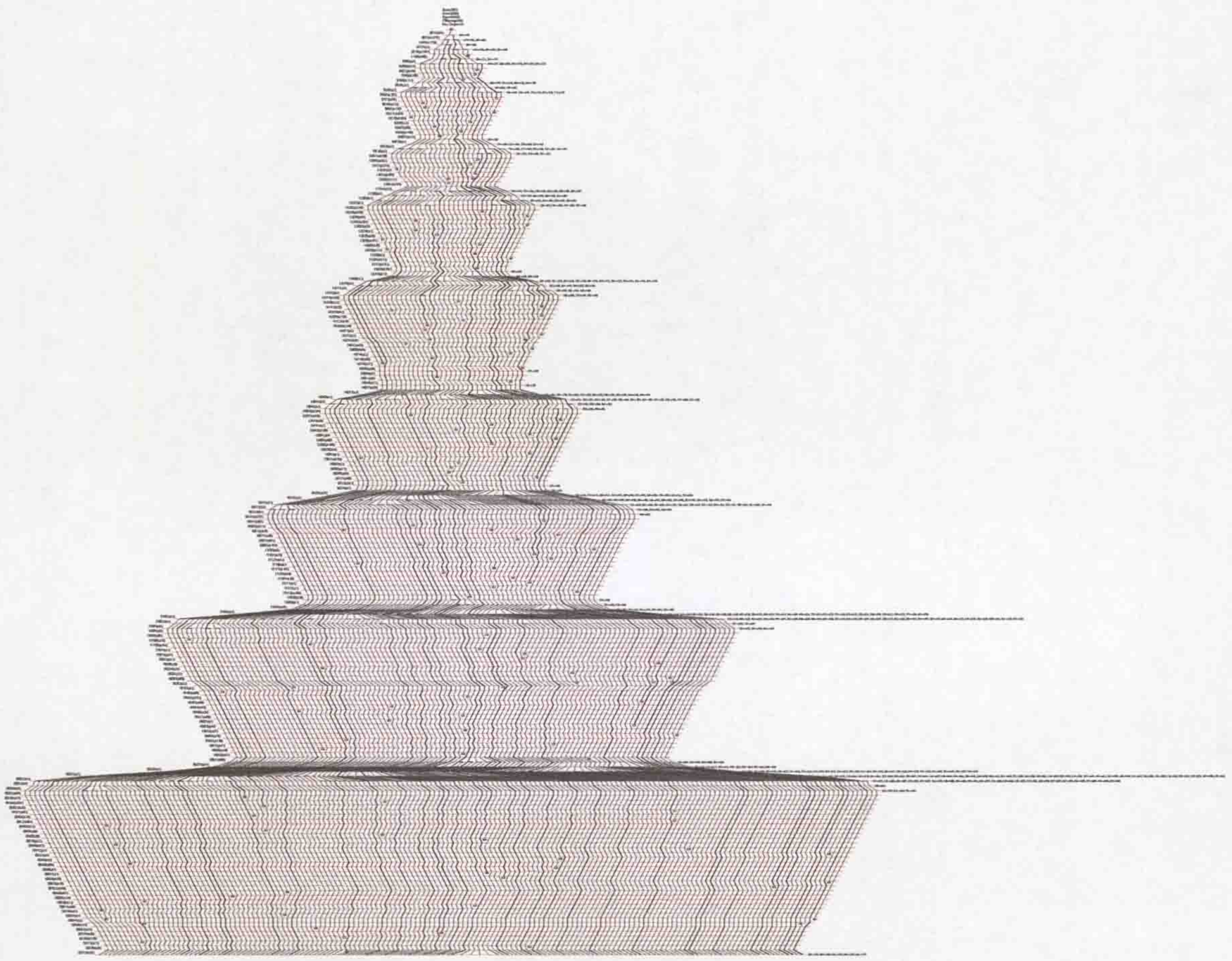


Figure 7.9: Whole lineage of cell 0143 in run020623, with unexpected synchronised deaths.

Pop. Decline		Pop. Increase	
Time Span	# Cells	Time Span	# Cells
2403	8	-	-
964	6	407	12
1311	8	59	16
1326	12	45	24
1341	15	30	30
1276	21	95	42
1613	37	55	74

Figure 7.10: Timing differences between multiple peaks and troughs of run020623, cell lineage 0042 during unrestricted exponential growth.

Pop. Decline		Pop. Increase	
Time Span	# Cells	Time Span	# Cells
1180	12	190	24
1246	10	125	20
335	13	35	26
1214	16	157	31
1277	22	94	41
1092	31	278	61

Figure 7.11: Timing differences between multiple peaks and troughs of run020623, cell lineage 0198 during unrestricted exponential growth.

doubling will occur every 22 minutes or 1320 seconds. The dynamics of these growth cycles are then related to the doubling time. Looking more closely at the timing of deaths and division there is another pattern, cell division occurs at around the same time (±1 second) and then while all cells are growing at maximum speed some are killed because they are regardless considered non-viable. As a result, the population increase timespan is quite arbitrary as it only records the last cell death before the next growth cycle. Also, cell death can be from a previous growth cycle, so although it can be said that death occurred in that cycle, the cycle is an interpretation placed on the data to explain the pattern. Deaths happen all the time, but are more likely before the next cycle.

This data does then allow us to calculate the best case growth rate in the presence of brittle genomes, what proportion of new cells will fail to themselves divide and hopefully some hint that evolution is increasing the survival rate and so working to avoid brittleness of genome.

Figure 7.7 shows the same exponential growth but with a uniform time step. Figure 7.12 shows the same graph but with the vertical axes focused on the cell birth and death rates. As can be seen, the population increase is proportional to the total population size. This is expected since all cells live in a maximum growth environment. The death rate however appears linear, at the final time step there were 280 cells, and 80 deaths; at 275 minutes there were half as many deaths but 120 cells. If the cells are not dying at an increasing rate then genome evolution must be making the genomes less brittle. This is surprising considering the selective pressure is minimal.

Non-viable cells are terminated after 8.33 minutes of continuous inactivity, inactivity occurring either immediately after division or some time after, following the 'death' of an unmaintained enzyme. Cell termination must occur before cell division or the inactivity counter is reset and the cell gets another chance. Enzyme half-life is the only mechanism that can delay cell death. As this is around 2 minutes and the minimum doubling time is 22 minutes, there is a high probability that inactive cells are killed. We fail to see an exponential

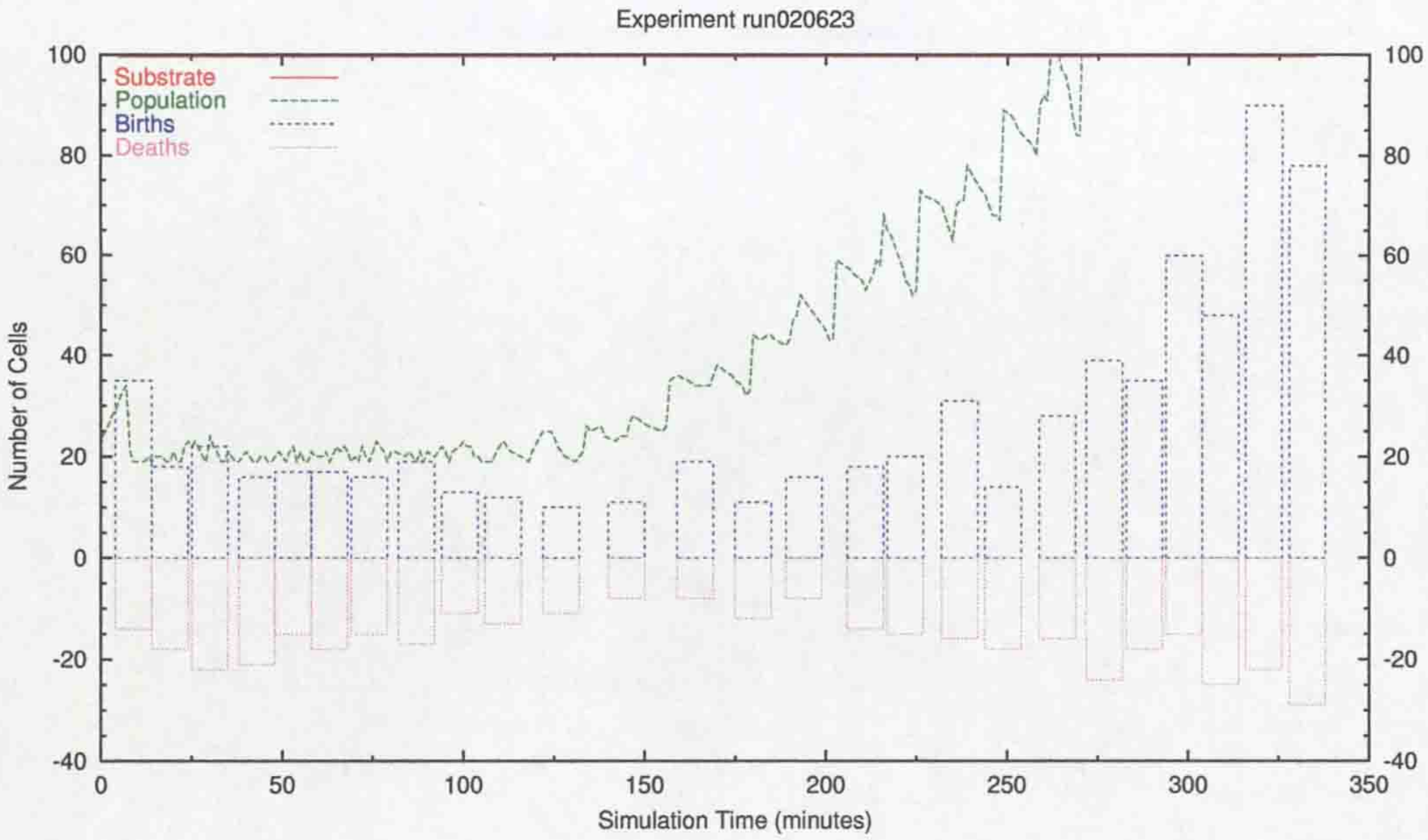


Figure 7.12: Birth and death rates of run020623.

increase in death rates and yet this mechanism must surely scale up, the only conclusion is then improved pathways, by this mechanism killing cells that are more brittle than others.

At this point the host system had effectively run out of hard disk space, short of deleting results a new hard drive needed. This took some time, and in the mean time the results of first two simulation runs were lost to a mistake. While in temporary storage space they were deleted by the system, configured to delete temporary files over a week old. Unbeknown to this author, the data sets were considered old the moment they were moved, the move did not reset the access date and so in the early hours of the next morning the first two datasets (run020501 and run020516) were lost. It is at least fortunate that these where the earliest part of the testing phase and considering the significant bugs had the least value.

With the arrival more hard dish space, the results of the previous run could then be considered. There was the option to either test the above with 100/100 enzyme sharing and so compare the death rate, or modify the cell division criteria so that division varies around 0.4 fg, rather than exactly 0.4

fg in all cases. This would then test the probable cause of division synchrony. The latter option was chosen as the next experiment, this was considered much more of a bug than the expected loss of cells through brittleness.

7.9 Simulation run020813

Before this simulation a change to COSMIC was then needed, the threshold of cell division based on cell mass was changed so that each cell had an individual threshold of 0.4fg $\pm$ 10%. This simulation was also looking at the "jiffies" computation algorithm (chapter 5) for better CPU time throttling to help COSMIC coexist with other users.

This simulation ran for 6 days but revealed an over sight with substrate consumption, it was enabled and so there was no chance of exponential growth. It is quite likely this was deliberate; following complaints from the Linux farm administrator that these simulation were having a big impact on the users of those machines. This complicated the simulation, not only did it need to simulate but it must also be considerate of other users. This aspect was taken further in chapter 5.

As this was then effectively a bug fixed COSMIC with a constrained environment it gave a result in itself. As expected there was smoother growth and decline of the population, especially in line with environment nutrients. The result looks nothing like the previous run, the regular cycle in population size has gone. As this was a nutrient limited environment we can expect large differences in population size but would still expect the cycle to be present had the cause not been found and corrected. The only oddity comes from the population drop at time 500 minutes as shown in figure 7.13, this hardly shows when looking at the lineage, either at the whole population or at cell 0143. This cell accounted for 629 cells of the 915 cells tested. It is suspected that some genetic event had started to take effect at around time 440 minutes. Identifying all the deaths at this time and checking for a ancestor in which the death rates on both sides of the cell division are difficult. Would require writing a program

to count lineage depths, heights, deaths per lineage and then computing death ratios, and another to extract all the death events that occurred in that time frame, and another to combine the result and so pick out all cell death ratios at that time. There did not appear to be any way of doing this. What is known is that there were 81 deaths over this time span, 37 had a balanced tree, 7 were within 50% (looking at the relative balance of each lineage tree), 24 had an 100% imbalance, 5 had an 200% imbalance, 7 had an 300% imbalance and only one had an 400% imbalance. 29 were right leaning, 15 were left leaning and 37 were balanced. This one cell would suggest an insertion or deletion event caused a deleterious cell line that survived along side other cells for some time but eventually its brittleness came to show. This was cell 0225, the tenth in the lineage. Cell 0255 is the eleventh and had 5 offspring, cell 0536 had one offspring, so clearly the ratio by itself is misleading as the death of 6 cells is insignificant compared to the 81 deaths. There needs to be an algorithm that can trace back to a ancestor or the most common ancestor, that would seem to be the most reliable. Alas it is also not clear how this could be done, as it cannot be done by hand it certainly cannot be done by machine.

This could potentially mean looking for nodes that lead to a lot more deaths than typical, yet only counting nodes from this window of time. The cleanest approach is then to plot the entire lineage for cell 0143 and rank each daughter cell based on the number of deaths that occurred in the time window. The effect is then a cumulative rise in the number of deaths going back to the original founding cell. It was hoped this would then show a knee point where a cell increased the death rate. As seen in figure 7.14 there is a knee when plotted on a logarithmic axis. The first cell after the knee is cell 0193 in the third generation with 94 deaths, before the knee is cell 0193 in the second generation with 140 deaths. How real this knee actually is is unclear, cell 0193 in the second generation also lead to cell 0253 with 46 cumulative deaths. It could be that these three cells represent some random event and then the death rates amount to some meaningless summation of death rates, nearer the founding cell there are fewer summations summing larger figures and so the

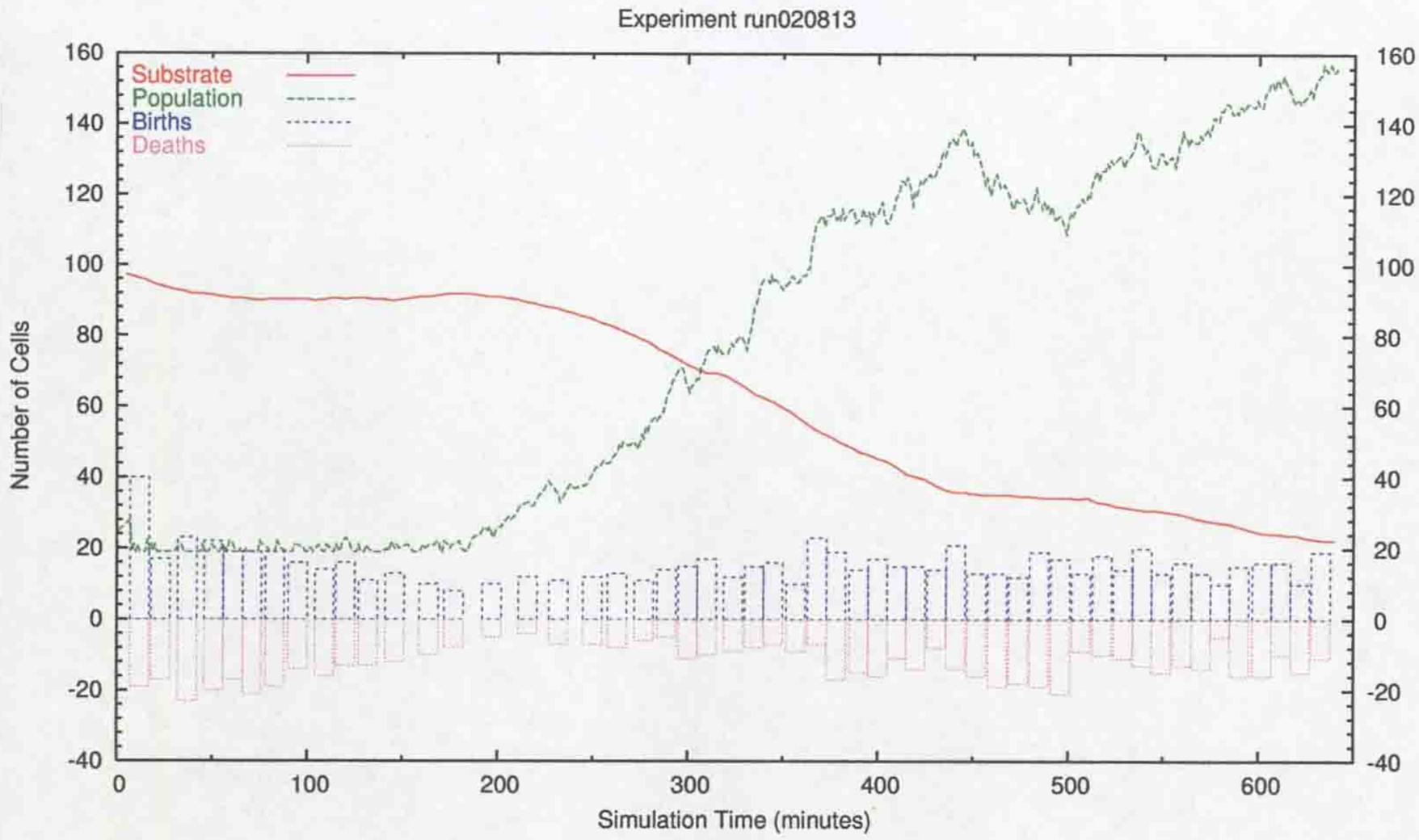


Figure 7.13: Population of run020813.

result is bound to be more noisy.

The only solid conclusion then is the difficulty in finding a suitable metric regardless of having access to the complete data set. The compounding feature is the lack of any steady states with which to compare, this data set starts and represents an exponential rise in cell numbers so is as far from stable as is possible.

7.10 Simulation run020820

As the previous simulation was run with a constrained environment by mistake, this simulation corrects that mistake by giving the environment unrestricted substrate. A hardcoded limit of 200 cells was placed in the code to stop the simulation running away with itself as the last exponential run showed it was important to limit the impact on the machines it executes on.

This simulation ran for 3 days, the limit of 200 cells was never acted on, the wrong variable was being compared and so the simulation went exponential. In the end around 490 cells were active, though very slowly as the server had

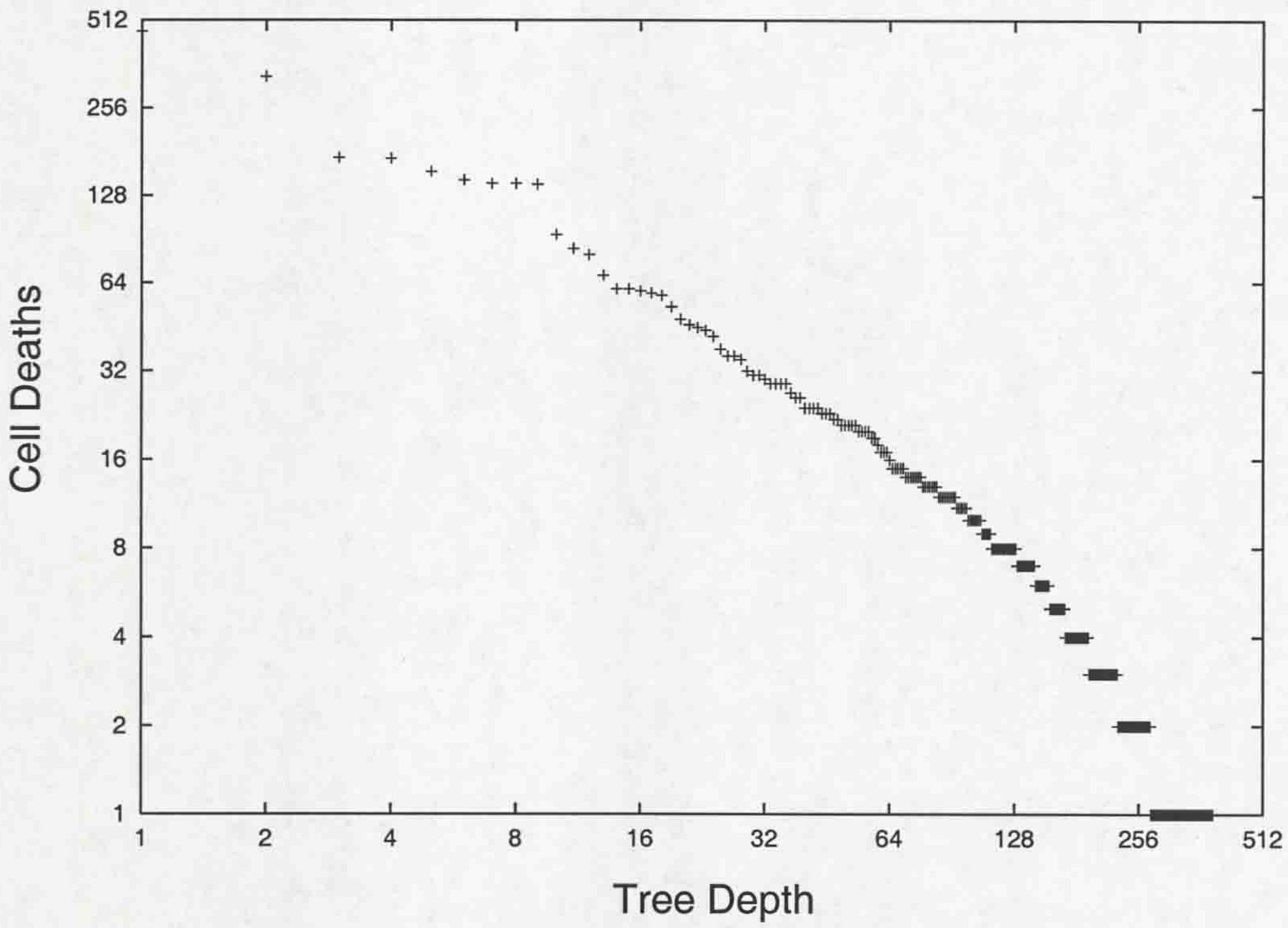


Figure 7.14: Ranked cell death counts for each cell of run020813. Counting only deaths between times 440 - 500 minutes, to coincide with the population drop at that time.

reached its maximum process limit and used around 1Gbyte of memory, ie. well into swap space and having to access the HD on every server(i.e. environment) iteration. PVM `reset` on the server took an hour to flush all the local processes and recover from this drain on resources. Nobody else was affected by this as the simulation was based on a single machine, had the 200 limit been acted on even this machine would have been able to cope.

Looking at the growth in general in figure 7.15, it is clear there is still a synchronous event tying together the cells. The lineage charts show one cell line (cell 0143) dominated and the population chart shows a repetitive shape to the large rise, small wobble and fall. The small size of the fall is very different to past simulations, suggesting the pattern isn't growth synchrony but something else, possibly related to the division of enzymes. Division of enzymes has been tested before, but at that time the growth synchrony problem existed in its main force - division at exactly >0.399 fg). To further ensure synchrony is not caused by division, the implementation was tested. The overall finding was that although the client cells never have their own cell division level, the server based minimal copies do and it is this copy that the division decision is based on. This proves that synchrony is caused by some other mechanism, there are no clues as to what that mechanism is.

It was also considered that the cause could be too little variation of the division level, resulting in the maximal cell growth rate always triggering division in the same round. But cell division is tested every second, which is a maximum growth rate of 0.15 attol and that is much smaller than the individual variation of ±20 al.

7.11 Simulation run020905

The lack of explanation then forced a search for a more serious problem. The simulation was run again with consumption of substrate (*options.ConsumeSubstrate*) enabled, to compare with run020813. This is because past runs have taken on a different shape of growth (run020813 seemed much smoother than past runs)

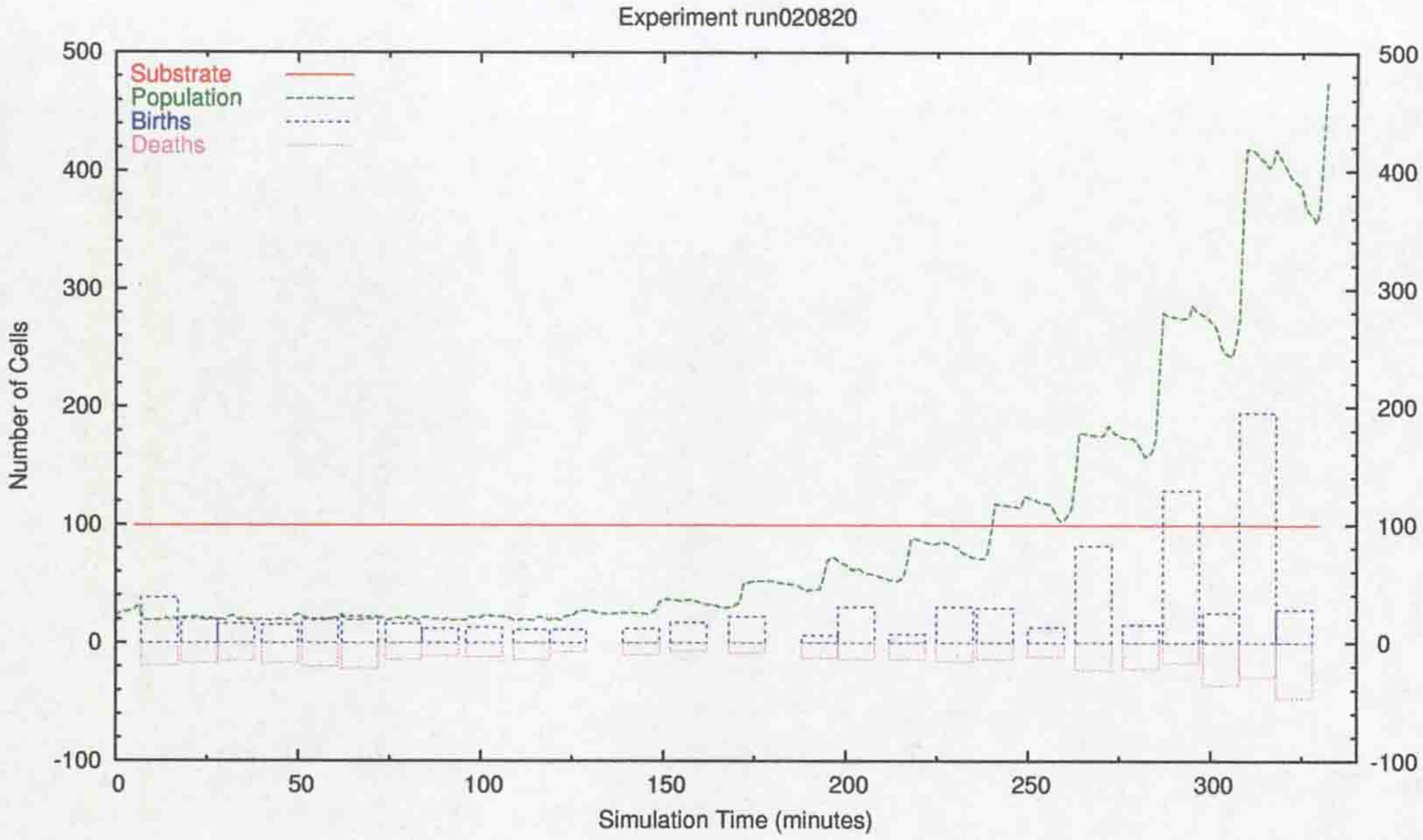


Figure 7.15: population of run020820

despite the changes should have made no difference.

Running COSMIC then turned out to be difficult, the PVM logs reported what it called bogus packets followed by messages saying that the server machine was unreachable. These messages were recorded on the server machine so something was clearly wrong. Client machines seemed to lock up, normal pings of 0.6-9 msecs were replaced by pings of 5.0-6.0 msecs, with no possibility of logging on to further identify the problem. All things pointed to a kernel problem and so before triggering the flaw a check was made on the kernel version and it showed it had been recently updated. Further emails confirmed it was an NFS kernel bug in the kernel running on the Linux farm and so COSMIC had to wait while the technical support staff found the solution.

After the problem was resolved COSMIC ran without any of these messages in the logs and so ran reliably. This run was meant to compare run run020813 (the first variable division run, in a dynamic environment), whose main feature was a period of mass death.

As shown in figure 7.16 the overall growth rate was similar, as would be expected. However there was no large population drop, so in that sense this

run was very different in terms of outcome. As the cause of the drop was never found, the reason for the lack of a drop in what should be an identical run is also beyond the analysis here, only a detailed examination of the genetics can hope to give any reasons.

There were also initial similarities, again there was only one dominate lineage and again cell 0143 was the founding cell. This seems to be a common event, given the same set of random numbers the simulation is deterministic enough to lead to the same cell number gaining the advantage, the non-deterministic effects come later and hence the overall divergence.

In the mean time, the possibility of heterogeneous environments was added to the simulation. Rather than resupply all areas of the environment with the same glucose level, this allows a picture to make that specification. The lighter the area of picture the more that area of environment is replenished. All based on a linear pixel value to femtolitre mapping.

Also added was the ability to scroll this replenishment map over time and so force cells to follow its movement, the scroll rate is specified at a sub map resolution giving cells time to evolve this skill. To this date this feature goes unused, a more complex environment was the not required when the data is so difficult to analysis at the moment.

7.12 Simulation run021102

Although run020905 did not match run020813 as it should have, this did not seem too much of a problem. Between simulations there are often minor changes that could change the output. It would never have been expected that such a big change could occur, but COSMIC is a complex system and it quickly becomes hard to say either way without running the same simulation again. Events outside of COSMIC meant there were other areas to test, a new cluster arrived that was initially dedicated to running COSMIC.

On 02/11/02 COSMIC was tested on the cluster for the first time, it ran from around 4 days. In that time the simulation went through 48500 iterations

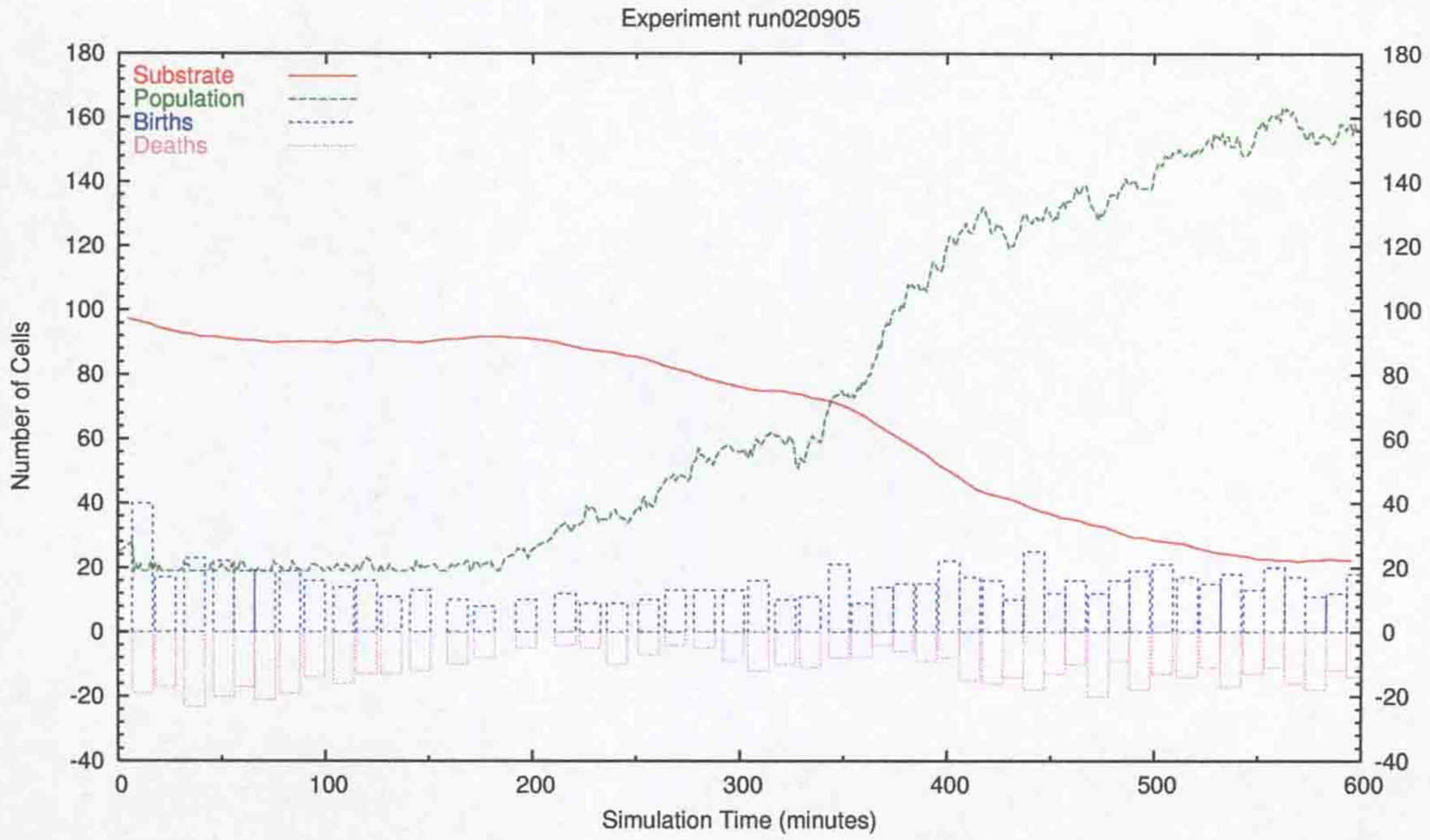


Figure 7.16: Population of run020905.

(13 hours, 28 minutes) of simulation, much faster than the general purpose Linux farm ever managed. When taking into consideration only the speed increase the new cluster it is still far faster than the public Linux farm. The reasons for this are not clear, the cluster is faster per CPU on the order of 25%, has slightly faster networking and is not held back by other users. NFS replaces `ssh` as the file transport but the trickle of data should never have made that a factor. In 4 days and with 26 CPUs the cluster simulated 13.5 hours, on the general purpose farm this would have taken 11+ days, normally never actually getting there because some problem on the Linux farm forced a halt to the simulation.

Ideally the simulations would need to use total cell complexity integrated with cell population for true accuracy, but that figure can itself only be estimated. Instead we use days simulated as the main measure of work done. From run020905 we see took 21 days to simulate .42 days on the farm, and run021102 took 4 days to simulate 0.55 days on the cluster, this amounts to a per system speed increase of 6.88 and so nearly transforms a week of simulation into a day. This is despite simulation run020905 on the farm spending less

time with a large population, simulating fewer cells in total and ending with a smaller population.

Looking at the systems themselves, the 26 dedicated CPUs need to be compared to the variable but always increasing number of CPUs used in the farm [1]. Counting the total number of machines used in the farm run reveals there were 10, one acting as host for both clients and server. Taking into account the number of machines reduces this to 2.6 and then taking into account the relative speed of the CPUs gives an overall increase of 2.1.

The reason for the large speed increase would then seem to be mainly the larger number of nodes by a factor of 2.6 and the lack of other users slowing down the whole simulation by a factor of 2.1. Though this latter point also includes `ssh` overhead (especially when starting a new cell). The effect of starting a new cell is usually to pause for a second or more while that new cell starts and creates result files on the server by using `ssh`, NFS shows comparatively no pause. Considering the situation where there are many cell processes per node and so nodes are heavily loaded, even a pause of several seconds is insignificant as many cells will still be completing their iteration. So the effects of `ssh` seem minimal, only really showing in the initial stages of the simulation.

Also notice how little difference the speed of the processor actually makes. Comparing the prices of complete systems and the speeds of CPUs shows that although speed might be doubling at a pace, the cost of a faster CPU does not justify the speed increase when compared to many slightly slower machines. The downside is of course that multiple machines requires parallel algorithms, more space and more infrastructure.

Coming back to the simulation itself we see again in figure 7.17 the same characteristic shape of population growth and again the equally characteristic differences when given the same conditions. This time the growth rate is

[1] Always increasing since PVM ensures CPUs can't be removed from a running simulation, only added. Machines were added when the load generated by other users had dropped to minimal. They could then be added under the assumption users would not choose to log into a heavily loaded machine.

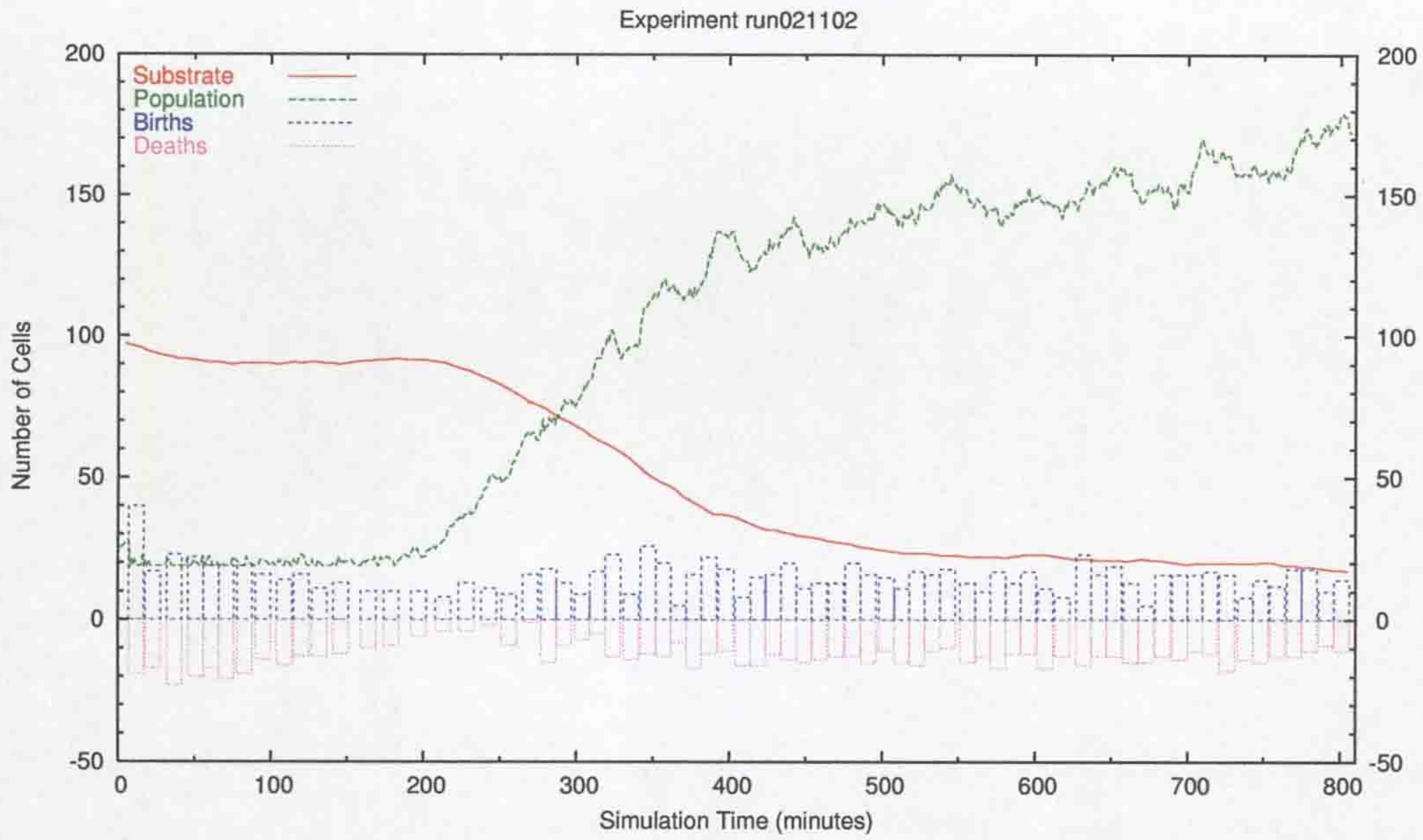


Figure 7.17: Population of run021102

arguably more aggressive.

7.13 Simulation run030116

As the parallel efficiency was deemed important enough to measure, this simulation was set up solely measure efficiency. An accurate measure meant simulating for a long time and so the simulation ran for 21 days in a restricted substrate environment. To calculate efficiency, a measure of jiffy usage per round for each client was converted to per second and passed back to the server. Under Linux there are 100 jiffies per second per CPU that the scheduler then distributes among processes wanting to run. COSMIC totals the jiffies used by each CPU, leading to an overall efficiency per CPU and then a total for the system as a whole can be calculated. This required some small additions to the existing code as system resource were previously unmonitored, largely because nothing can be done with the knowledge to aid the current simulation and so was not initially planned for. Not only is there no mechanism to pass more accurate load balancing hints to the PVM client spawning function, there is

no also way to change what turns out to be a bad load balance.

Unfortunately, after the 21 days were up the data was examined to find there was a bug in the jiffies calculator that made the result useless, it was fixed and quickly restarted to obtain the real data. The bug was simply %d instead of %f in the final printf() call. Regardless of this error the results had to be kept in some form as they also represented the latest output and so were used in a publication.

7.14 Simulation run030205

This run represents the last recorded simulation of COSMIC. More have been made since then but there was never the storage space to really check the results. In hindsight it was a mistake to run this simulation for so long and so use so much storage space. The main focus was the same as the previous simulation, to calculate efficiency, but keeping it running also allowed a chance to see how far the system runs, long term evolution having always been cut short in the past by some constraint of the underlying hardware.

COSMIC ran for just over a month, evaluated 3131 cells with around 410 coexisting when the simulation was stopped. It generated 100 gig of compressed raw data. Enzyme numbers were in the ranges 25-260000 enzymes per cell with a mean of 10000 (figure 7.19). Genome sizes were in the ranges 70-20000 genes (figure 7.18), with a slow tendency to grow in size over the population ending with a mean of 800. This growth approximately follows the function:

$$g = 10^{(t-400)/1700\ +\log 80}$$

A constant exponential growth despite gene insertion and deletion being unbiased. Even if there were a bias it should still be manifested as a linear increase.

The length of this simulation should have allowed evolution the chance to show itself more strongly, the signs of this should then have shown in the external sensing and reactionary controls regardless of finding direct evidence of change amongst a cells regulation network. Looking at the total IO levels over

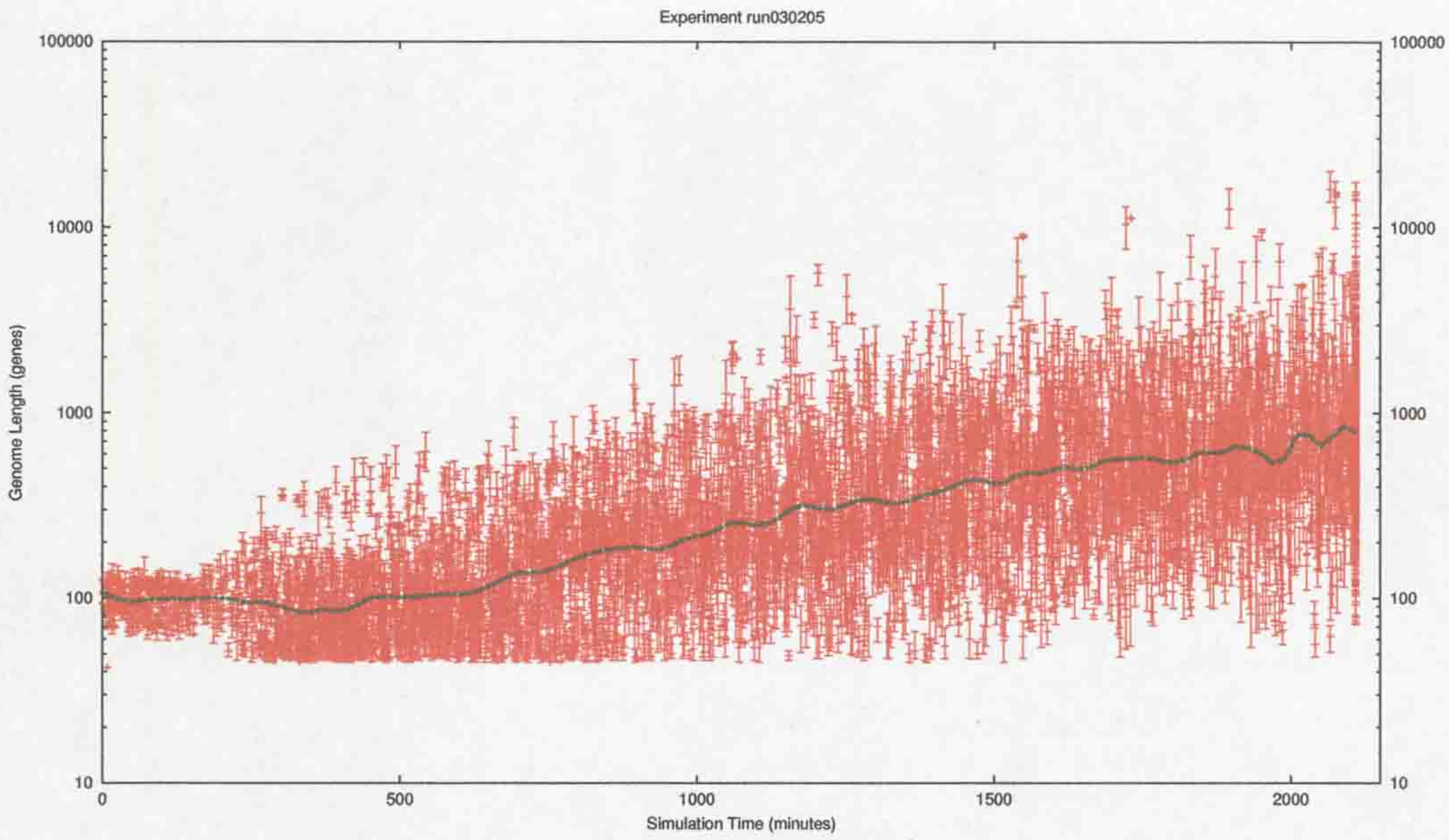


Figure 7.18: Gene distribution of run030205

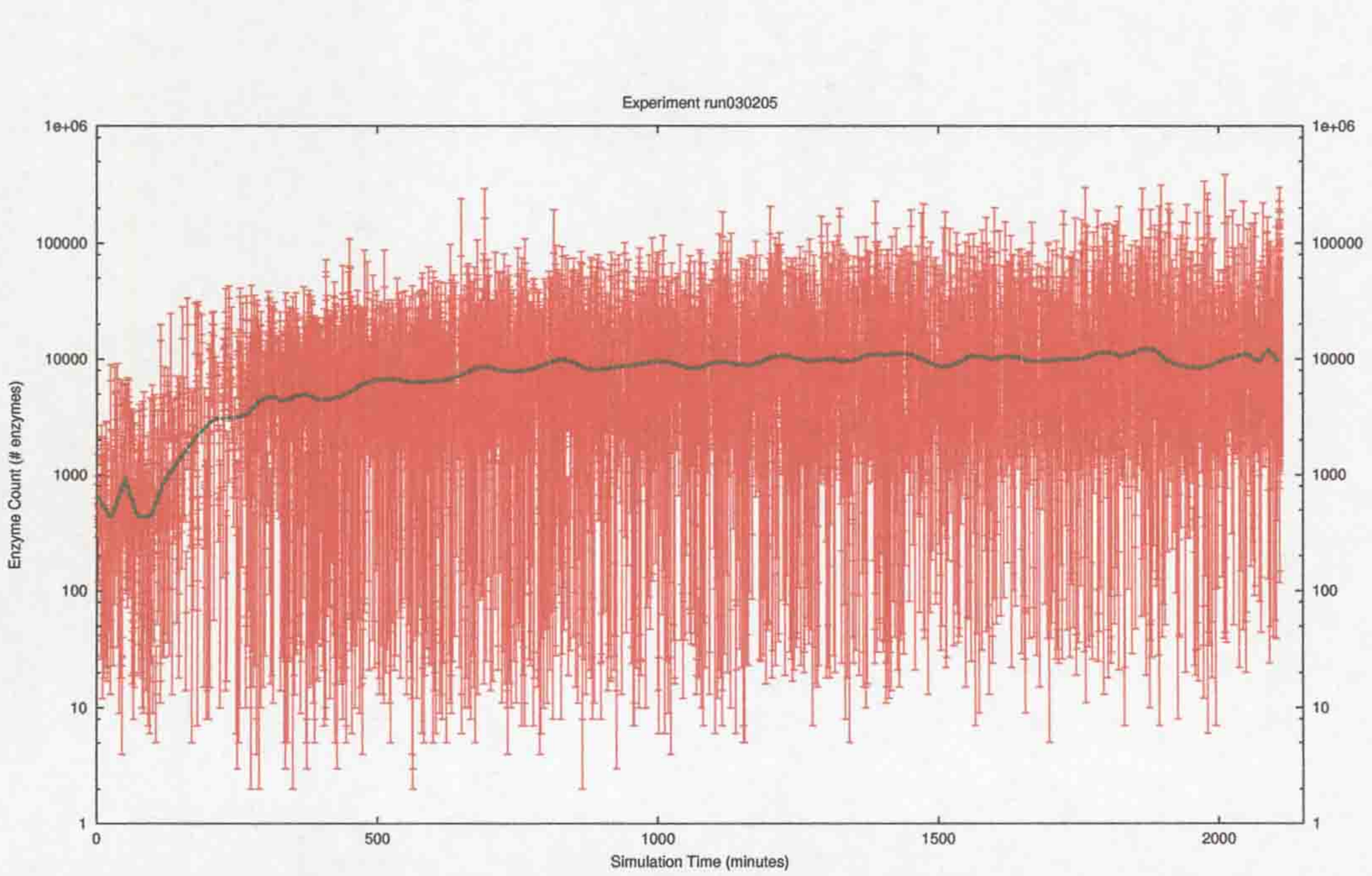


Figure 7.19: Enzyme distribution of run030205

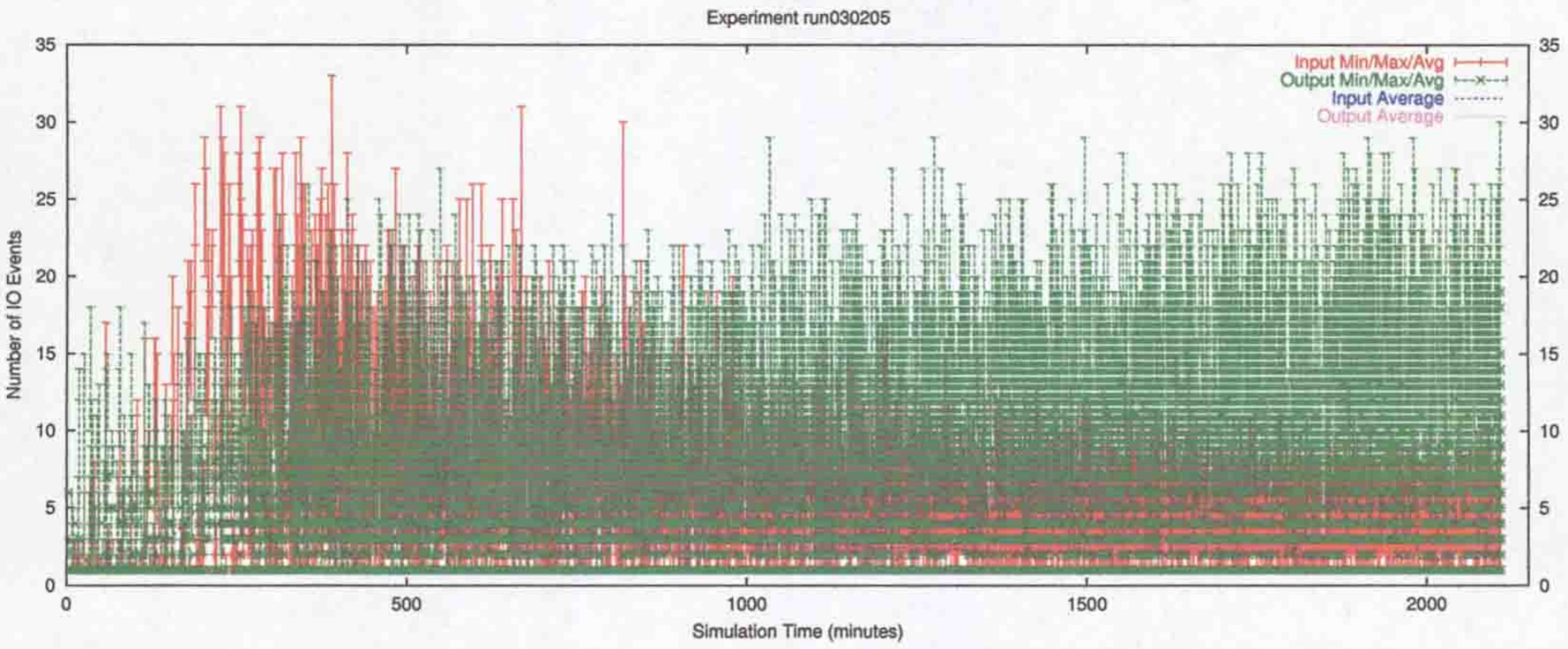


Figure 7.20: IO event activity of run030205

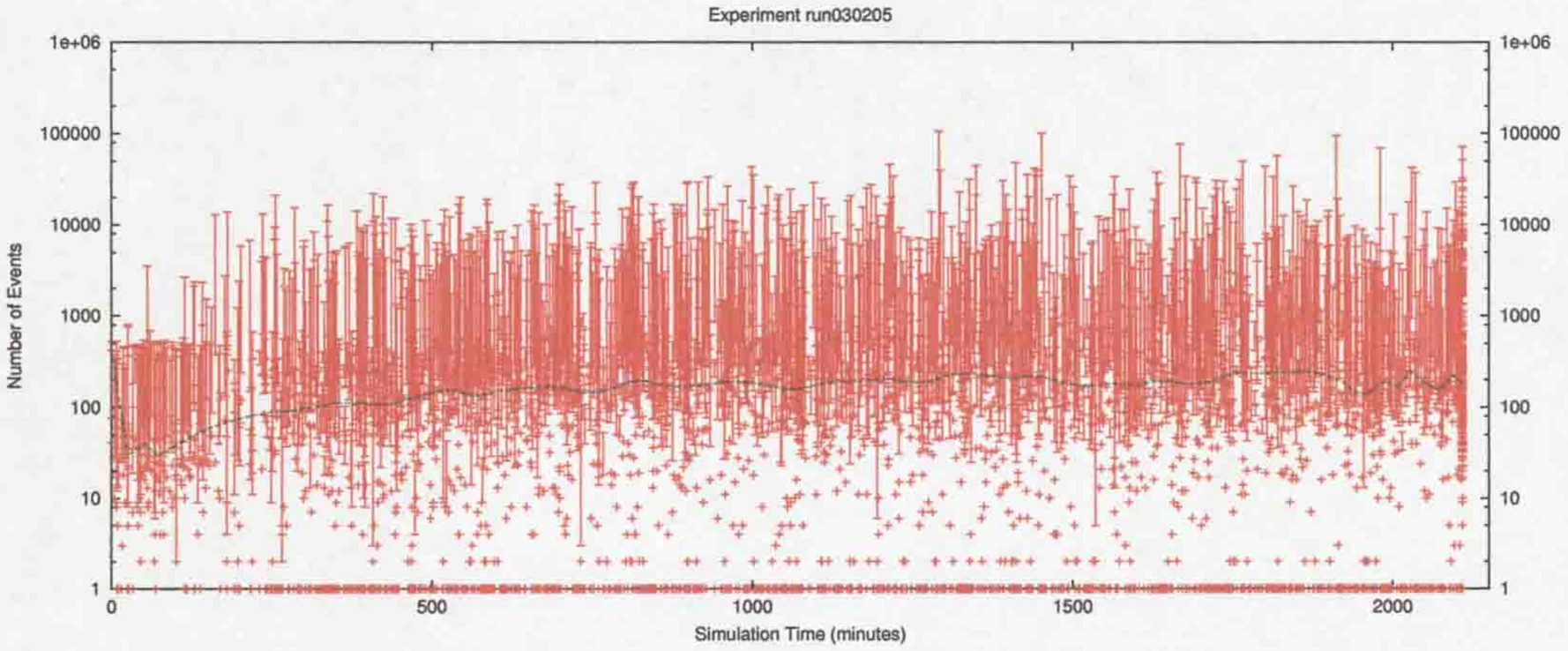


Figure 7.21: Event activity of run030205

time (figure 7.20) and the internal gene interactions over time (figure 7.21), the large number of failed offspring show cells can easily become damaged at cell division or possibly because of some mutation event. A successful cell needs to be motile, yet even when offspring of cell 0143 are dominating the population there are still many cells that suddenly don't move or suddenly stop sensing their environment.

These figures show overall IO activity diverged with cells increasingly moving more but sensing their environment less. The reduction in receptor activity is to be expected as this also relates directly to the substrate in the environment, dividing the receptor(input) average by the substrate concentra-

tion should then remove this bias and so show any real effect. The rise in the output events is presumably a result of the artificial selection for more motile cells, the increase is slow, noisy and linear during the stable population growth phase. In an ideal simulation the increase in motility would be caused by an increase in receptor activity over and above the rate in which the environment depleted (the receptor/environment ratio), then showing evolution is heading toward the full goal. If that ratio is constant then the only cause remaining is simply the artificial selection pressure for motility. If it should be negative then that would surely indicate the evolutionary pressure is failing to act and evolution to the full goal is impossible.

Note. The probability of receptor activity is based directly on the substrate concentration, with all other variables constant this relationship is linear.

The ratio of receptor activity to environment substrate over time was then obtained and is presented in figure 7.22. The obvious pattern comes from there being so few events that each band is an integer number of events and as substrate decreases, receptor activity ratio is pushed higher. The bands make it clear that this increase is not an actual increase in receptor activity as a constant substrate would render each rise as an unchanging integer. The overall result then leads to the belief that the ratio is falling and the drop in receptor activity is real and so evolutionary pressure is not working to force input receptor use.

However, what cannot be seen is the concentration of each band, there are too many data points to discern the concentration of each band. To give some idea, figure 7.23 shows figure 7.22 but only shows every 7th data point, here the band of ratio 0.2 comes out move clearly as a probable most popular average. There are few receptor events per cell in many cases and too many data points at the same time, so to be sure figure 7.24 shows the same information but sums together data points that occur at the same time. This should then average out the noisy receptor activity before the ratio is taken, making an average total of events per time frame that is lower than the maximum seen in figure 7.22 and leave the substrate unchanged (as the average of identical numbers is the

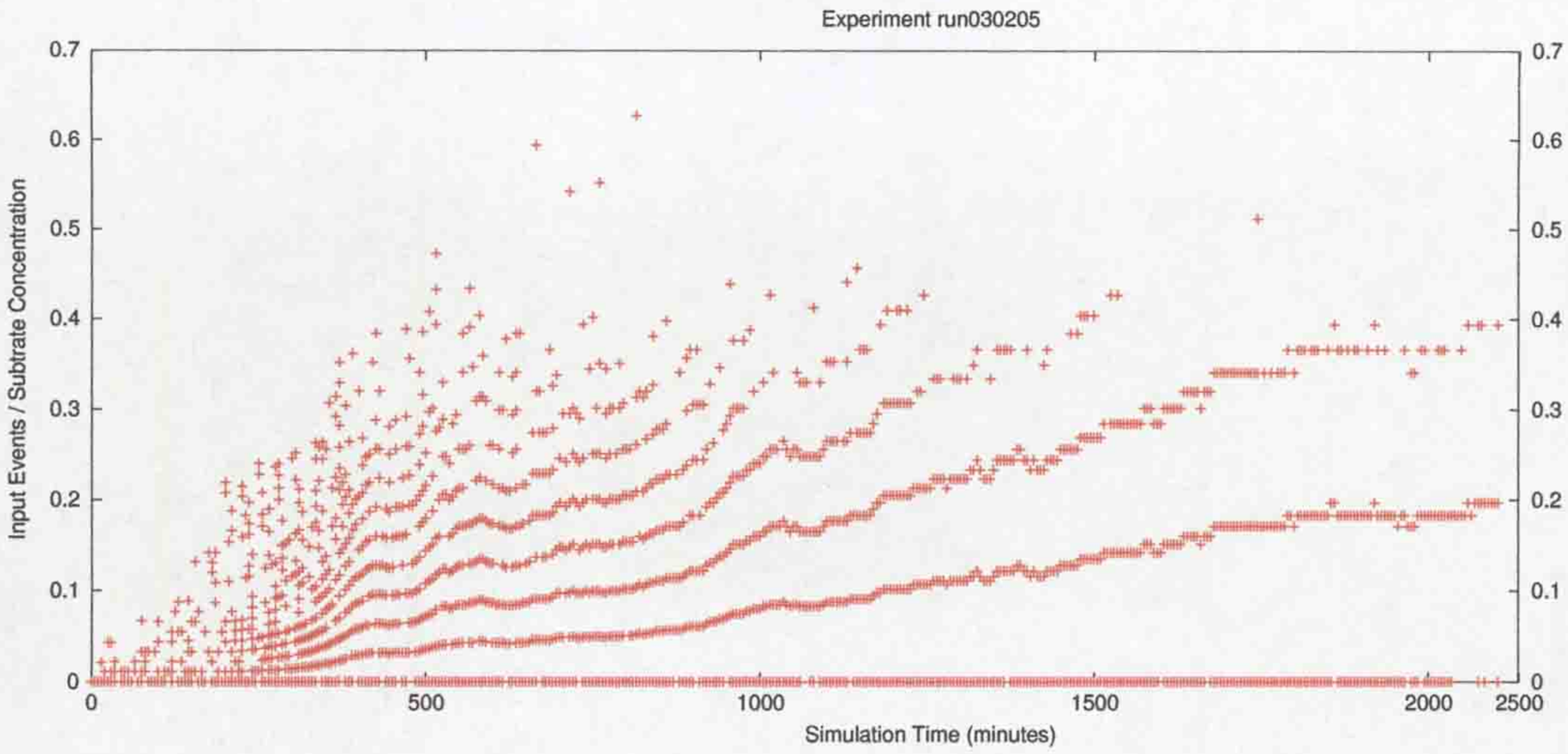


Figure 7.22: Ratio of receptor events to substrate concentration over time

only result). This avoids there being many cells having no receptor events and so showing as a ratio of 0 in a very crowded row, also reducing the average.

Strangely this shows the opposite scenario where the effective receptor activity increase more than the environment substrate decreases and so leads to the opposite conclusion. There is always the chance that the data mining script written to extract this information may be in error, as might any of the many scripts, but in this case the effect would seem to be real. Showing the scattering of receptor activity was misleading because it was too crowded to highlight the most popular trend. From this then I would conclude there is reason to believe that both the receptor activity and motility is increasing over and above modulating factors and so the COSMIC system is evolving toward the intended goal.

The original purpose of COSMIC was to test for evolution. What the previous sections have shown is that in a large computational system this can be hard to show. There are many reasons for this, the holistic approach to modelling, the size of the data sets, the possibility of modelling errors, both intended simplifications necessary for the model and unintended errors normally called bugs. There are also problems related to the hardware and software layers around the simulation, especially when using a parallel system.

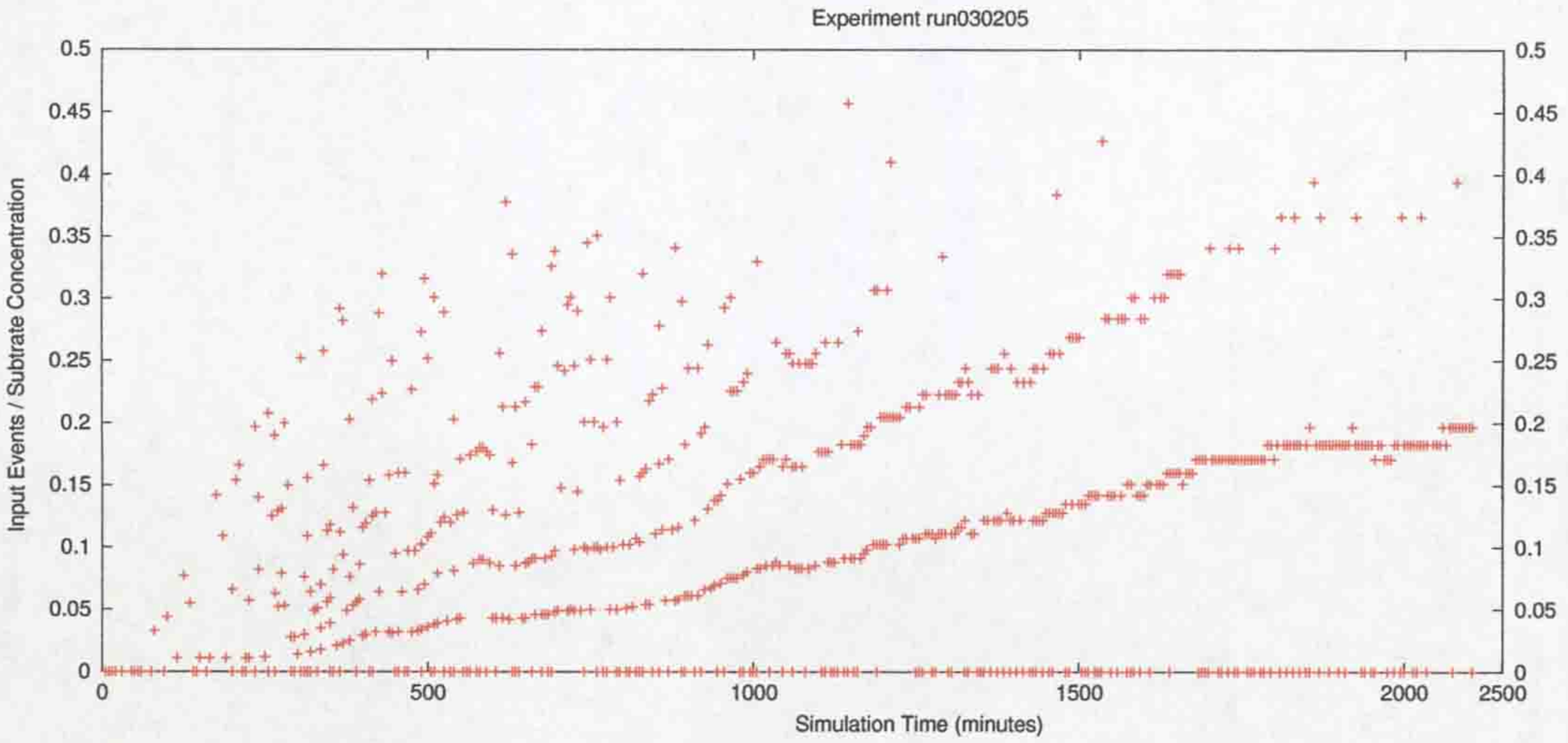


Figure 7.23: Sparse ratio of receptor events to substrate concentration over time

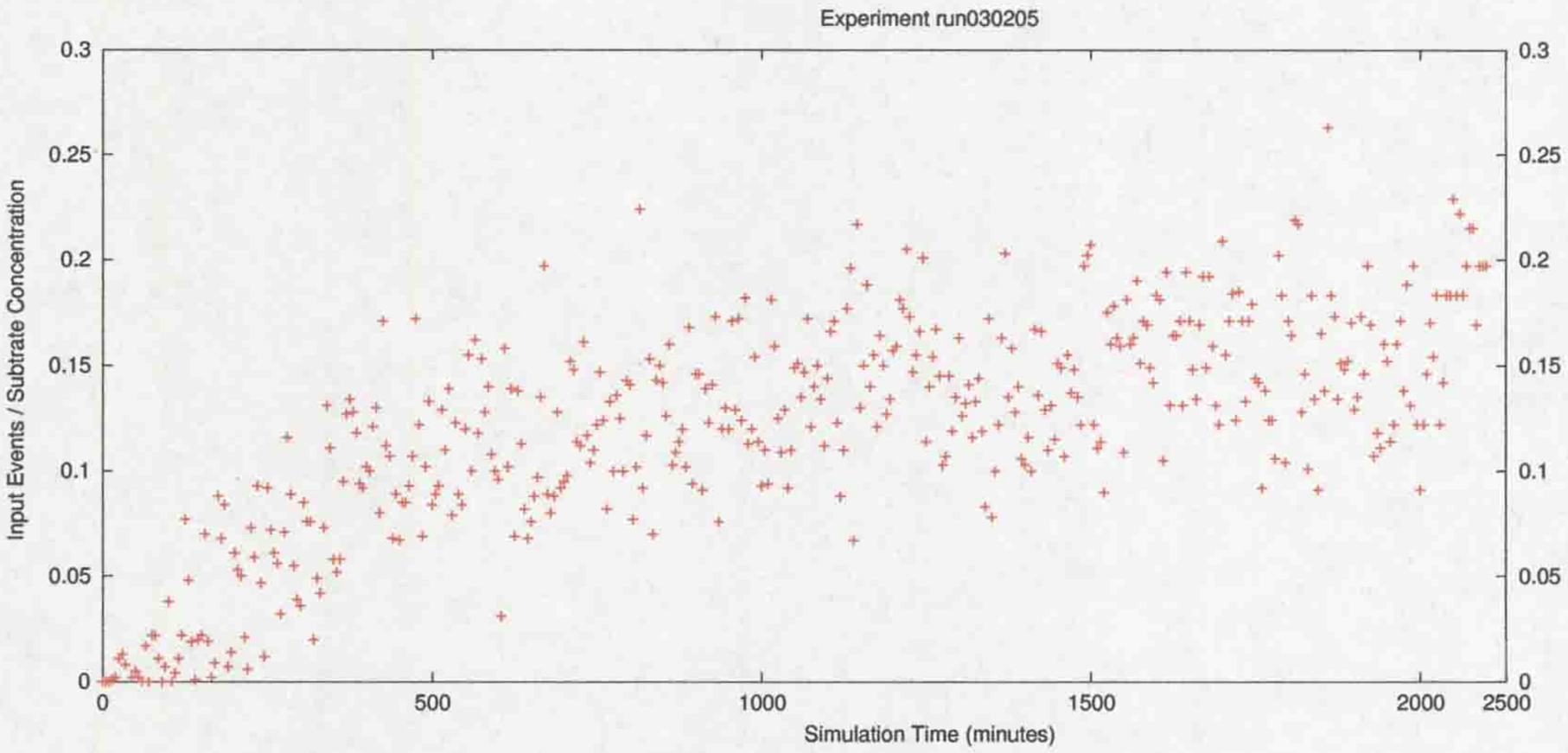


Figure 7.24: Lumped ratio of receptor events to substrate concentration over time

These problems have demonstrated that this kind of model is both hard to write and takes a long time to implement, even when the simulation is seemingly well defined. These problems and the new questions brought about through running COSMIC have meant the original goal of finding evolution was in fact only one of many possible avenues to explore.

7.15 Summary

This chapter has brought together a large collection of output data from the COSMIC simulation, and demonstrated a thread of development that ultimately highlighted how ambitious the goals of COSMIC are. The single result was a simulation system that should have the ability to show evolution taking place, if only the data sets could be mined for that information. Considering the bigger picture, COSMIC has gone beyond most simulations in that it is also a generic framework of bacterial growth and development, taking it closer to wide range of applications and other scenarios.

The problem of analysis was originally intended to be a central part of COSMIC. As time progressed it became increasingly clear that COSMIC is not the whole solution to simulating evolution and maybe no single system can. The complex holistic interactions between genes, gene products and the changing genome they come from make this a topic of research in itself, as are efficient parallelisation, effective visualisation, implementation quality and modelling itself. In the beginning it was thought partial solutions to the analysis problem already existed, of the solutions seen they are certainly not applicable inside a time frame of a few years work.

Chapter 8

Conclusion

8.1 History

Many months were spent searching the literature looking for some optimisation approach that is novel, technically possible and would lead to something bigger rather than be a conclusion in itself. The idea of COSMIC had yet to be formed but it was clear that the fields of Genetic Algorithms, Evolutionary Algorithms, Evolutionary Strategies were already crowded and all had something in common, a simple abstract structure that used very few ideas to implement. At the same time, experiences with the biological field showed that you could never truly distil a real world system down to an abstraction and still have it perform in the same way, biological systems were always complex and never existed inside a black box with clear boundaries between model, input and output.

During this same period the field of genetics seemed to be gaining public popularity, with the human genome nearing completion and the sequencing techniques that project had developed being more accessible. It appeared we were on the verge of a new age of genetics spurred on by the possibility of being able to read genetic information in enough detail to make some understanding possible. On reading the genetics literature, genetics seemed to be made of up of apparently simple structures and yet at the same time was so hard to

decipher and measure. There were also frank talks by biologists who said they cannot use a computer, that programming was a personal nightmare[1]. Seeing their computer based work that was no exaggeration, there is a clear difficulty in being expert in multiple areas. As genetics was obviously a transformed field, there was a gap in which a Computer Scientist could fit.

There was however a choice of directions, optimisation or simulation. Having initially come from optimisation there was a clear application of using biological information to add something to the current evolutionary optimisation algorithms. This seemed plausible, there were many possible evolutionary mechanisms available beyond mutation and crossover (which boldly takes no account of the difference of scale in crossing over chromosomes and crossing over parameters).

However, on reading of the No Free Lunch theorem [WM97] the future direction became much more biased. For me this amounted to saying there was no better global optimiser as each method is extremely application specific. As a result, there was no point in applying novel biological evolutionary operators when they can all be as good or bad as each other, each depending on the exact scenario.

There was also the stark difference between evolution in biology and evolution for optimisation, and that is biological evolution never tries to optimise to an global optimum. Normally optimisation (or specialisation as it could be termed) is bad for the organism concerned, a change in environment leads to its death. Nature is instead a trier with infinite patience.

These two points reinforced the need to move in the genetics direction. Especially the problem of dealing with measurement. Being based in Electrical Engineering some of their approaches seemed applicable, if reality cannot be measured directly then build a model and measure the model. Originally the thinking was of simulating genetic networks with neural networks, but rather than there being a clear input and output, the network would be a network of networks which ultimately fed back into itself. Again this came from an

[1]Often publicly stated by biologists at multidisciplinary conferences such as IPCAT and MIPNETS.

engineering viewpoint. As neural networks take a long time to train, it was also thought that the sub-networks should be predefined units having some known function. This approach had its advantages of being clear and abstract, but at the same time had the disadvantage that the representation did not really fit with known genetics. There seems to be a better analogy to brain function, with higher and lower levels.

The deep-seated belief that to be a reasonable model it also had to be simple meant the simplest organism possible had to be used, this was not the time for considering brain function, or even familiar animals.

The ideas of neural networks remained while it became clear that bacterial genetics and optional transcription could be considered in the same way. The best choice for a model organism was clearly *E.coli*, being comparatively simple but also one of the most widely researched organisms. In the context of the above neural networks approach, here the higher level functions define themselves through evolution rather than being defined before hand. It was at this point that the outline of an internal COSMIC cell was created, with a view to evolving optional transcription pathways on the genome.

8.2 The COSMIC Model

COSMIC needed to be designed in such a way that would be an open-ended genetic simulation. Firstly it needed a representation that was extremely open ended to support evolution, but at the same time was machine readable. From this came the idea of genes having an encoding like real genes but also a high level meaning that essentially told the reader that it was a gene of some kind. Secondly it needed some form of gene product that avoids the protein folding problem but still makes gene products specific to some genes, and those relationships had to be stable over time and across generations if evolution was to be possible. This came back to the low level encoding placed on each gene, matching genes to gene products based on a deterministic matching rule gives a stable relationship that can be inherited across generations. Thirdly, this

same representation could also support transcription by having a transcription mechanism which was sensitive to gene and gene product type, effectively implementing promoter sites, RNA polymerase and sigma factors. Yet another relationship could then easily represent the action of repressors and operator control genes. Lastly, the actions of these could also be modelled on this representation using stochastic functions that take parameters directly from the gene and gene products.

This early COSMIC model appeared to have the potential for exploring many genetic issues of real bacteria, largely evolution of the genome and adaptation to adverse conditions. Simulating evolution is such detail is an extremely novel approach, nobody has tried to couple evolution to the fine grained simulator of genetics. As a result, COSMIC then focused on this area by also including all the other aspects that were required.

In simulating evolution it is clear that evolution takes place in the cell but needs something to push it there, this is true for all of us and is the space where we live. A bacterial environment is not so complex. As BacSim [KBW98] showed, bacteria might only require one source of food, and if they share that environment then they are also effectively in competition with each other for that food. Looking at BacSim also showed that bacterial growth with competition could also be considered a kind of roulette wheel selection as found in genetic algorithms. BacSim had no concept of genetics, but in the context of COSMIC, the cells that grow faster will divide sooner and so pass on their genetic information more often.

The required step was to link the transcriptional network to the environment, and this was done using receptors on the cell wall that sensed substrate (food) concentration, and other receptors that activate a swimming action. This stage then forms a feedback loop in which the cells use their genetic network to sense the environment and initiate an action. This is something of a departure from how real *E.coli* achieve motility, the process is chemically driven and random. These facts make realistic motility out of the scope of COSMIC and would massively increase computation time, when COSMIC is

already a highly computationally demanding application. This was a very necessary simplification. The incorporation of the environment and so linking it with genetics made COSMIC novel in another way, the interaction of multiple scales in the same model. This is new for genetics and the environment.

With the addition of an environment was the chance to simulate evolution in a much more realistic way. Although still abstract there is much more detail in COSMIC than normally found. It was expected that powerful mutation operators could be applied to the genomes of each cell - transposons, sequence insertion, plasmids, bacteriophages, the latter making COSMIC a multi species simulation. Unfortunately the complexity and time involved in actually implementing the few paragraphs above meant most could not be implemented. Sequence insertion and deletion was implemented as a proof of concept, the others represent future work. Regardless of the type of evolutionary operator, the current state of COSMIC still puts it well ahead of any other genetic simulation simply by combining some evolvability with genetics.

In the beginning COSMIC was envisaged as a simulator of evolution, but it has become a framework for simulating bacteria-like genetics, one application of which is evolution. It was expected that sequence insertion and deletion should provide the cells and their genomes with enough adaptive ability to evolve the task of following substrate gradients. The evidence supporting this remains inconclusive and highlights how ambitious the COSMIC project is, and so how much of an achievement it was to take COSMIC this far.

8.3 Outcomes

Rather than finding an instance of evolution, the outcomes of COSMIC have become much more interesting, even though not always expected. This project has created the COSMIC framework, which is modular and would allow another similar model to be built, or the current COSMIC improved on. COSMIC was always considered a system that required additions, looking at biology there were always new processes to add, so COSMIC was written with

modularity and extendibility it mind.

There is currently support from the DTI to port the COSMIC system to a Grid enabled cluster[2], with option of free time on one of the countries largest clusters as well as time on a research companies cluster. The process of porting is expected to give the opportunity to further expand the scope of COSMIC, both by increasing cell numbers and making it more applicable to known bacterial problems.

There have been three publications [GPSW03a, GPSW03b, VGPSW04] derived from the output of COSMIC, the topics have been both in terms of simulating evolution and reducing simulation time to a practical level. Many more publications could come from COSMIC, the visualisation aspect has yet to be published as has the detailed cell model. Aspects of the transcription network representation are also worthy publication topics.

8.4 Challenges

In conceiving the COSMIC model there have been a significant number of challenging problems which have been overcome. Like so many aspects of carrying out a design, there were some problems that seemed to be solved only to find later that reality has more limitations than can accounted for by reasoning and design alone.

The modelling of genetics is itself a major piece of work. COSMIC brings together the genes, optional transcription mechanism and the possibility of many kinds of mutation mechanism; all in a form that is computable, while also being a very good analogy to real bacterial mechanisms. Solving this challenge has meant COSMIC can then be applied to any similar problem.

Combining the genetics with an environment was also a challenge that had to be solved in order to complete the feedback loop for evolution. Apart from cell population density averaging and substrate diffusion, the COSMIC model of the environment is just as complete as BacSim and yet effectively couples

[2]DTI (e-Science support), reference THBB/008/00134C.

these two scales.

Making COSMIC computable was another major challenge. Unlike any other known genetic level simulator, COSMIC was designed and implemented for a parallel machine, as it was clear computers are pitifully slow when compared to real world processes, even after simplification. This in fact created three challenges, actually implementing the system to some degree, making it efficient enough to be worth the added effort of using a parallel system and making COSMIC more robust in the light of reliability problems inherent in using clusters.

With the detailed interactions occurring inside the cell, one of the unforeseen problems was recording the data in a form that could later be retrieved. Not knowing the required resolution of output meant recording all data, and arranging the data so that it can be cross referenced to other data. The visualisations that appear throughout this work are all based on that data. The main problem is not knowing what was important, in an open-ended simulation such as COSMIC it cannot be seen before hand. As a result, the visualisations were created after some experience had been gained, but this cannot be taken to an extreme as past examples can be bad examples. The main experience gained from this is in designing simple textual file formats that are readable by common utilities, with a file format for each possible kind of output. Using bit stream compression then avoids the problem of recording unnecessary resolution as it automatically removes redundancy. Recording the data was only one side of the storage task. Data also had to partitioned in a form that was accessible for years to come despite many other simulations. Recording also had to avoid underlying limitations of the file system. Experience showed the best indexing scheme, based on date, cell number and type of data.

The final major challenge was obviously visualisation and analysis. This has been the biggest challenge of all, and in many respects will always be incomplete. This comes from COSMIC capturing much more than wet lab experiments can hope to capture. The various visualisations provide views of the simulation during its execution, with different visualisations for each scale.

In the context of real biology the data is not available to draw these diagrams, and so they do not exist. The COSMIC project had to find ways of extracting indicators from the many data files and translate them into something understandable. This has been done but we then find that being understandable is not enough, and the data must be manually mined using yet more visualisations that are highly specific to the question being asked. So, following this challenge we realise that the real challenge then is not to create visualisations but to create something that will easily create user specified visualisations.

It was also soon realised that analysis was beyond the scope of the COSMIC project even though it would have been a huge help in finding evidence of evolution. Of the mathematical methods researched they all made fundamental assumptions that were not true of COSMIC. It could be that those methods were more generic than they appeared, if that is the case then they are themselves research topics.

8.5 Complexity

The Individual based modelling paradigm is used throughout the COSMIC simulation. This approach considers all elements of the system to be individuals, with their own state. For COSMIC this means that the genes were individuals linked together with their own set of events at a given time. The gene products are also individuals that exist inside the cytoplasm of the cell, and interact with each other and with the genome on an individual by individual basis. At the level of the environment, all the cells are individuals with their own genomes and their own set of gene products, in their own position in the environment.

It is a consequence of this that the COSMIC model had to use a parallel implementation, as even with careful design this requires a significant increase in computing resources. The advantage of this approach is that the simulation and the data sets are much richer in information than they would be if differential equations were used. Differential equations only give averages,

but we consider the individual variations to be very important. Using this approach means we could potentially track every interaction that ever occurs and consider if it represents a source of evolution.

Another consequence has been the difficulty in understanding the interactions involved. We can watch a population of thousands, we can watch an individual, but it can easily be some other individual (be it cell or gene product) that is responsible for change.

8.6 Future Work

There are now many avenues which COSMIC can travel. As part of a separate project, COSMIC is currently being ported to Grid enabled cluster. Standards are converging on Grid so this would seem to be prudent. During that time it is also expected that the full COSMIC model will be published.

Inclusion of the originally planned plasmids, phages and simple mutation are also welcome additions, as it is hoped these additional mechanisms will improve the chances of seeing evolution.

There has always been an interest in taking COSMIC toward a more specific application, such as the modelling of biofilms at the genetic level or bacterial drug resistance. Each would required extensions to COSMIC but are quite possible.

Finally, and has been said many times, visualisation and analysis present the next major hurdles to understanding what is happening in a COSMIC simulation. This is in both areas of visualisation based analysis methods and mathematical analysis methods. It is hard to see which would be the more fruitful.

8.7 Final Word

This material might seem far removed from typical engineering but in the long term it is impossible to say. In around 1936, Otto M. Schmitt completed

his doctoral thesis on Electronic Computer Simulation of the Nerve Action Impulse. Many years later he found those ideas have been incorporated into common place devices such as the Schmitt trigger, emitter-follower, the differential amplifier and the heat pipe [Sch93]. This shows how simple ideas from a biological source can have huge applications in engineering, and any other field too. I do not claim that COSMIC will be seen in the same light, but do I do claim that many aspects of the COSMIC approach are useful for truly complex real world problems.

Bibliography

[AR94] Arkin, A. & Ross J. (1994). "Computational Functions in Biological Reaction Networks". In *Biophysical Journal*, 67, pp. 560-578.

[BD96] Bremer, H. & Dennis, P.P. (1996). "Modulation of Chemical Composition and Other Parameters of the Cell by Growth Rate". In *Escherichia coli and Salmonella: Cellular and Molecular Biology*, F.C. Neidhardt *et al* (Eds) Volume 2. 2nd Ed. ASM Press. pp. 1553-1569.

[Bec96] Beckwith, J. (1996). "The Operon: An Historical Account". In *Escherichia coli and Salmonella: Cellular and Molecular Biology*, F.C. Neidhardt *et al* (Eds) Volume 1. 2nd Ed. ASM Press. pp. 1227-1231.

[Bha00] Bhalla, U.S. (2000). The Many Faces of a Biological Switch. In *Computation in Cells: Proceedings of an EPSRC Emerging Computing Paradigms Workshop*. Hamid Bolouri, Raymond C. Paton (Eds). Tech Report 345, April 2000, pp. 33-36.

[BL97] Barkai, N. & Leibler, S. (1997). "Robustness in simple biochemical networks". In *Nature* 387. pp. 913-917.

[Boo97] Booth, G. (1997). "Gecko: a continuous 2-D world for ecological modeling". In *Artificial Life* 3, pp. 147-163.

[Bra90] Bray, D. (1990). "Intercellular signalling as a Parallel Distributed Process". In *Journal of Theoretical Biology*, 143, 215-231.

[Bra95] Bray, D. (1995). "Protein molecules as computational elements in living cells". In *Nature*, Vol. 376. July.

[BS00] Becskei, A. & Serrano, L. (2000). Engineering stability in gene networks by autoregulation. *Nature 405*, pp. 590-593.

[BSS00] Bonsma, E., Shackleton, M. & Shipman, R. (2000). "Eos - an evolutionary and Ecosystem research platform". In *BT Technology Journal* Vol 18, No. 4. October 2000, pp. 24-31.

[CD98] Colombetti, M. & Dorigo, M. (1998). "Evolutionary computation in behavior engineering". In *Evolutionary Computation: Theory and Applications*, X. Yao (Ed.), World Scientific Publ. Co., Singapore, in press. BC.03-WORLD98.ps.gz (834K) (Also Tech. Rep. TR/IRIDIA/1996-1, IRIDIA, Université Libre de Bruxelles.)

[CDM92] Colorni, A., Dorigo, M. & Maniezzo, V. (1992). "An investigation of some properties of an "Ant algorithm"". In *Proceedings of Fourth International Conference on Parallel Problem Solving From Nature.* Elservier Publishing, 509-520.

[CGE98] Collado-Vides, J., Gutièrrez-Ríos, R.M. & Bel-Enguix, G. (1998). "Networks of transcriptional regulation encoded in a grammatical model". In *BioSystems*, V47, No.1,2. pp. 103-118.

[COM89a] Cairns, J., Overbauch, J. & Miller, S. (1989a). "The origin of mutants". In *Nature* 335, 142-148.

[CP97] Clark, L. & Paton, R.C. (1997). "Toward Computational Models of Chemotaxis in Escherichia Coli". In *Information Processing in Cells and Tissues*, Mike Holcombe and Ray Paton (Eds). pp. 39-45.

[CW98a] Cao, Y.J. & Wu, Q.H. (1998). "A Cellular Automata Based Genetic Algorithm and Its Application in Mechanical Designed Optimisation". In *Proceedings of UKCCA, International Conference Control*, 1-4 September 1998, University of Wales, Swansea, U.K., Vol.2, pp.1593-1598.

[CW98b] Cao, Y.J. & Wu, Q.H. (1998). "An Improved Evolutionary Programming Approach to Economic Dispatch". In *International Journal of Engineering Intelligent Systems.*

[Dav89] Davis, B.D. (1989). "Transcription bias: a non-Lamarckian mechanism for substrate-induced mutations". In *Proc. natn. Acad. Sci. U.S.A.* 86, 5005-5009.

[DC94] Dorigo, M. & Colombetti, M. (1994). "Robot Shaping: Developing Autonomous Agents through Learning". In *Artificial Intelligence*, 71, 2, 321-370.

[DG96] Dorigo, M. & Gambardella, L.M. (1996). "A Study of Some Properties to Ant-Q". In *Proceedings of Fourth International Conference on Parallel Problem Solving From Nature.* N.-H. Voigt, W. Ebeling, I. Rechenberg and H.-S. Schwefel (Eds.), Springer-Verlag, Berlin, 656-665.

[DG97] Dorigo, M. & Gambardella, L.M. (1997). "Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem". In *IEEE Transactions on Evolutionary Computation*, V1, N1.

[DHB00] Duan, Z., Holcombe, M. & Bell, A. (2000). "A logic for biological systems". In *BioSystems* 55, pp. 93-105.

[DKH94] Duan, Z., Koutny, M. & Holt, C. (1994). "Projection in temporal logic programming". In *Proceedings of Logic Programming and Automated Reasoning*, Lecture Notes in Artificial Intelligence, a subseries of LNCS. Pfenning, F. (Ed.). Springer Verlag, Vol. 822, pp. 333-344.

[DL94] Durret, R. & Levin, S.A. (1994). "The importance of being discrete (and spatial)". In *Theoretical Population Biology*, 46, 363-394.

[DMC96] Dorigo, M., Maniezzo, V. & Colorni, A. (1996). "The Ant System: Optimization by a colony of cooperating agents". In *IEEE Transactions on Systems, Man, and Cybernetics*-Part B, Vol. 26, No. 1, 1996, pp. 1-33.

[Dor95] Dorigo, M. (1995). "Alecsys and the AutonoMouse: Learning to Control a Real Robot by Distributed Classifier Systems". In *Machine Learning*, 19, 3, 209-240.

[DPa97] Devine, P. & Paton, R.C. (1997). "Biologically-inspired Computational Ecologies: a Case Study". In *Corne D & Shapiro, J. (eds) LNCS Springer: Berlin.*

[DPA97] Devine, P., Paton, R.C. & Amos, M. (1997). "Adaptation of Evolutionary Agents in Computational Ecologies". In *Biocomputing and emergent computation: Proceedings of BCEC97.* World Scientific. ISBN 981-02-3262-4. Dan Lundh, Bjrn Olsson, Ajit Narayanan (Eds.), pp. 66-75.

[DPb97] Devine, P. & Paton, R.C. (1997). "Herby, an Evolutionary Artificial Ecology". In *IEEE International Conference on Evolutionary Computation*, USA.

[DPc97] Devine, P. & Paton, R.C. (1997). "Individual Based Modelling in an Explicitly Spatio-temporal Ecosystem". In *Proceedings of IMACS 97 World Congress*, Berlin.

[DPd97] Devine, P. & Paton, R.C. (1997). "Adaptation of Evolutionary Agents in Computational Ecologies". In *BioComputing and Emergent Computation Conference*, Sweden.

[DR96] Donachie, W. H. and Robinson, A. C. (1996). "Cell division: parameter values and the process". In *Escherichia coli and Salmonella: Cellular and Molecular Biology*, F.C. Neidhardt *et al* (Eds) Volume 2. 2nd Ed. ASM Press. pp. 1578-1593.

[Fel98] Feldman, D. (1998). "Information Theory, Excess Entropy and Computational Mechanics". In *http://leopard.ucdavis.edu/dave/index.html*, 5/2001.

[Fos95] Foster, I.T., (1995). "Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering". Addison-Wesley, Reading (Mass.).

[Fre00] Freeman, M. (2000). "Feedback control of intercellular signalling in development". In *Nature,* 408, pp. 313-319.

[GBDJMS94] Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Mancheck, R. & Sunderam, V. (1994). "Parallel Virtual Machine - A Users Guide and Tutorial for Networked Parallel Computing". In *http://www.netlib.org/pvm3/book/pvm-book.html.*

[Gil87] Gilbert, W. (1987). "The Exon Theory of Genes". In *Symposia on Quantitative Biology*, Vol. 52, 1987, 901-905.

[Gla96] Glansdorff, N. (1996). "Biosynthesis of Arginine and Polyamines". In *Escherichia coli and Salmonella, cellular and molecular biology.* Second Edition, Volume 1, pp. 408-433.

[GPSW03a] Gregory, R., Paton, R.C., Saunders, J.R. & Wu, Q.H. (2003). "A model of bacterial adaptability based on multiple scales of interaction", *Computation in Cells and Tissues - Perspectives and Tools of Thought.* Edited by: Ray Paton, Hamid Bolouri, Mike Holcombe, Howard Parish & Richard Tateson. Springer-Verlag: Heidelberg. Natural Computation series.

[GPSW03b] Gregory, R., Paton, R.C., Saunders, J.R. & Wu, Q.H. (2003). "Parallelising a Model of Bacterial Interaction and Evolution". In *Fifth International Workshop on Information Processing in Cells and Tissues.* September 2003. To appear in *BioSystems.*

[Hal91] Hall, B.G. (1991). "Adaptive evolution that requires multiple spontaneous mutations: mutations involving base substitutions". In *Proc. natn. Acad. Sci. U.S.A.* 88, 5882-5886.

[Hau97] Haussler, D. (1997). "A Brief Look at Some Machine Learning Problem in Genetics". In *Proceeding of COLT 97.*

[Hay99] Haykin, S. (1999). "Neural Networks - A Comprehensive Foundation". *Prentice Hall.*

[HJF97] Hraber, P.T., Jones, T., Forrest, S. (1997). "The ecology of echo". In *Artificial Life 1997* Summer;3(3), pp.165-90

[Holter *et al.*, 2001] Holter, N.S., Maritan, A., Cieplak, M., Fedoroff, N.V. & Banavar, J.R. (2001). "Dynamic modeling of gene expression data". In *PNAS*, February 13,2001. Vol. 98, #4. pp. 1693-1698.

[HPTW95] Hjelmfelt, A., Postma, P.W., Tommassen, J. & Westerhoff, H.W. (1995). "Signal transduction in bacterial phospho-neural network(s) in escherichia coli?". In *Federation of Microbiological Societies. Microbiology Review*, 16. pp. 309-321.

[IS00] Imhof, M., Schl̀otterer, C. (2000). "Fitness effects of advantageous mutations in evolving *Escherichia coli* populations". In *PNAS*, January 30, 2001. Vol. 98. #3, pp. 1113-1117.

[JM93] Jones, T. and Mitchell, M. (1993). "Introduction to the ECHO model", *Springer-Verlag*, Berlin, pp. 704-720.

[Kam96] Kampis, G. (1996). "Self-modifying systems: a model for the constructive origin of information". In *BioSystems,* 38, pp. 119-125.

[Kau93] Kaufmann, S.A. (1993). "The Origins of Order". *Oxford University Press.*

[KBW98] Kreft, J-U, Booth, G., & Winpenny, J.W.T. (1998). "BacSim, a simulator for individual-based modelling of bacterial colony growth". In *Microbiology*, 144, pp. 3275-3287.

[KH01] Kent, W.J. & Haussler, D. (2001). "GigAssembler: An Algorithm for the Initial Assembly of the Human Genome". *In press.*

[Koc93] Koch, A.L. (1993). "Genetic Response of Microbes to Extreme Challenges". In *Journal of Theoretical Biology*, 160, pp. 1-21.

[Karplus *et al.*, 1997] Karplus, K., Sj̀olander, K., Barret, C., Cline, M., Haussler, D., Hughey, R., Holm, R., Sander, C. (1997). "Predicting protein Structure using hidden Markov models". In *Proteins: Structure, Function, and Genetics.* pp.134-139, Supplement 1.

[KW82] Koch, A.L. & Wang, C.H. (1982). "How Close to the Theoretical Diffusion Limit do Bacterial Uptake Systems Function?". In *Archives of Microbiology* Volume 131, pp. 36-42.

[Mac96] MacNab, R.M. (1996). "Flagella and Motility". In *Escherichia coli and Salmonella, cellular and molecular biology.* Second Edition, Volume 1, pp. 123-144.

[Men97] Mendes, P. (1997). "GEPASI: a software package for modeling the dynamics, steady states and control of biochemical and other systems.", In *Trends in Biochemical Science,* 22, pp.361-363.

[MK97] McFadden, J. & Knowles, G. (1997). "Escape From Evolutionary Stasis by Transposon-meditated Deleterious Mutations". In *Journal of theoretical Biology*, (1997) 186, pp. 441-447.

[MS99] Marín, J. & Solé, R.V. (1999). "Macroevolutionary Algorithms: A New Optimization Method on Fitness Landscapes". In *IEEE Transactions on Evolutionary Computation,* Vol 3, No. 4, Nov. 1999.

[Nei96V1] Neidhardt, F.C. Ed. (1996). "*Escherichia coli.* Cellular and Molecular biology, Volume 1". *ASM Press, Washington, D.C.*, Second Edition.

[Nei96V2] Neidhardt, F.C. Ed. (1996). "*Escherichia coli.* Cellular and Molecular biology, Volume 2". *ASM Press, Washington, D.C.*, Second Edition.

[NTT96] Neijssel, O.M., Teixeira De Mattos, M.J. & Tempest, D.W. (1996). "Growth Yield and Energy Distribution". In *Escherichia coli and*

Salmonella: Cellular and Molecular Biology, F.C. Neidhardt *et al* (Eds) Volume 2. 2nd Ed. ASM Press. pp. 1683-1692.

[NU96] Neidhardt, F.C. & Umbarger, H.E. (1996). "Chemical Composition of *Escherichia coli*". In *Escherichia coli and Salmonella, cellular and molecular biology.* Second Edition, Volume 1, pp. 13-16.

[OLG00] Ochman, H., Lawrence, J.G. & Groisman, E.A. (2000). "Lateral gene transfer and the nature of bacterial innovation". In *Nature*, 18th May 2000, pp. 299-304.

[Pat98] Paton, R.C. (1998). "The Ecologies of Hereditary Information". In *Cybernetics & Human Knowing: A Journal of Second Order Cybernetics, Autopoiesis & Cyber-Semiotics*, 5, No.4, pp. 31-44.

[Pir67] Pirt, S.J. (1967). "A kinetic study of the mode of growth of surface colonies of bacteria and fungi". In *J Gen Microbiol*, 47, 181-197

[Pre48] Preston, F.W. (1948). "The commonness, and rarity, of species". In *Ecology*, 29(3), pp.254-283.

[Pre62] Preston, F.W. (1962). "The canonical distribution of commonness and rarity: Part I". In *Ecology*, 43(2), pp.185-215.

[PS84] Prigogine, I. & Stengers, I. (1984). "Order out of Chaos: Man's new dialogue with nature". *Flamingo, London.*

[PSD99] Pasemann, F., Steinmetz, U. & Dieckman, U. (1999). "Evolving Structure and Function of Neurocontrollers". In *Congress on Evolutionary Computation 1999*, volume 2, pp. 1973-1978.

[Rai99] Rainey, P.B. (1999). "The economics of mutation". In *Current Biology*, 9:R371-R373.

[RBL01] Riehle, M.M., Bennett, A.F. & Long, A.D. (2001). "Genetic architecture of thermal adaptation in *Escherichia coli*". In *PNAS*, January 16, vol. 98, #2. pp. 525-530.

[RLM96] Reddy, V.N., Liebman, M.N., Mavrovouniotis, M.L. (1996). "Qualitative analysis of biochemical reaction systems". In *Computers in Biology and Medicine* 26(1) pp. 9-24.

[Sch93] Schmitt, O.M. (1993). Personal communications with Paton, R.C.

[SFSCL97] Schaff, J., Fink, C.C., Slepchenko, B., Carson, J.H. & Loew, L.M. (1997) "A General Computational Framework for Modeling Cellular Structure and Function", In *Biophysical Journal,*, 73, pp. 1135-1146.

[Sha91] Shapiro, J.A. (1991). "Genomes as smart systems". In *Genetica,* 84, pp. 3-4.

[Sha97] Shapiro, J.A. (1997). "Genome organization, natural genetic engineering and adaptive mutation". In *Trends in Genetics* 13 (3). pp. 98-104.

[Sha99] Shapiro, J.A. (1999). "Genome System Architecture and Natural Genetic Engineering in Evolution", In *Annals of New York Academy of Sciences*, May 1999, 870, pp. 23-35.

[SL99] Schaff, J., Loew, L.M. (1999). "The Virtual Cell". In *Pacific Symposium on Biocomputing*, 4. pp. 228-239. *http://www.nrcam.uchc.edu/publications/vcell_publications.html*

[SLD79] Shuler, M.L., Leung, S.K. & Dick, C.C. (1979). "A mathematical model for growth of a single bacterial cell". In *Ann N Y Acad Sci* 326, pp. 35-55.

[Sta88] Stahl, F. (1988). "A unicorn in the garden". In *Nature*, 335, pp. 112-113.

[SW97] Shackleton, M. & Winter, C. (1997). "A Computational Architecture Based on Cellular Processing". In *Information Processing in Cells and Tissues*, Holcombe, M. and Paton, R.C. (Eds). pp. 261-271.

[Swa00] "Swarm Development Group", 2000. *http://www.swarm.org/.*

[TH96] Thain, M. & Hickman, M. (1996). "Penguin Dictionary of Biology, Ninth Edition". *Penguin Books.*

[Tomita *et al.*, 1999] Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T., Matsuzaki, Y., Miyoshi, F., Saito, K., Tanida, S., Yugi, K., Venter, J. C., Hutchison, C. (1999). "E-CELL: Software environment for whole cell simulation". *Bioinformatics,* 15, pp.72-84.

[TMBW97] Turner, P.C., McLennan, A.G., Bates, A.D. & White, M.R.H. (1997). "Instant Notes in Molecular Biology". *BIOS Scientific Publishers.*

[TN84] Tempest, D.W. & Neijssel, O.M. (1984). "The Status of Y_{ATP} and Maintenance Energy as Biologically Interpretable Phenomena". In *Annual Review of Microbiology* Volume 48, pp. 459-486.

[Taddei *et al.*, 1997] Taddei, F., Radman, M, Maynard Smith, J., Toupance, B., Gouyon, P.H. & Godelle, B. (1997). "Role of mutator alleles in adaptive evolution". In *Nature*, 387, pp.700-702.

[Ush99] Usher, M.M. (1999). "A Concurrent Visual Language Based on Petri Nets". In *Ph.D. thesis*, Liverpool University.

[Way01] Way, E.C. (2001). "The role of computation in modeling evolution". In *BioSystems*, 60, pp. 85-94.

[VGPSW04] Vlachos, C., Gregory, R., Paton, R.C., Saunders, J.R. & Wu, Q.H. (2004). "Individual-Based Modelling of Bacterial Ecologies and Evolution". To appear in *Compartative and Functional Genomics*, February 2004.

[WM97] WolPert, D.H. & Macready, W.G. (1997). "No Free Lunch Theorems for Optimization". In *IEEE Transactions on Evolutionary Computation*, V1, N1.

[WSF89] Whitley, D., Starkweather, T. & Fuquay, D. (1989). "Scheduling Problems and Travelling Salesman: the Genetic Edge Recombination Op-

erator". In *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, 1989.

[YD02] Yanai, I. & DeLisi, C. (2002). "The society of genes: networks of fuctional links between genes from comparative genomics". In *Genome Biology* 3(11), pp. 1-12.

[ZDB97] Ziegler, J., Dittrich, P. & Banzhaf, W. (1997). "Towards a Metabolic Robot Control System". In *Information Processing in Cells and Tissues*, Holcombe, M. & Paton, R.C. (Eds). pp. 305-317.