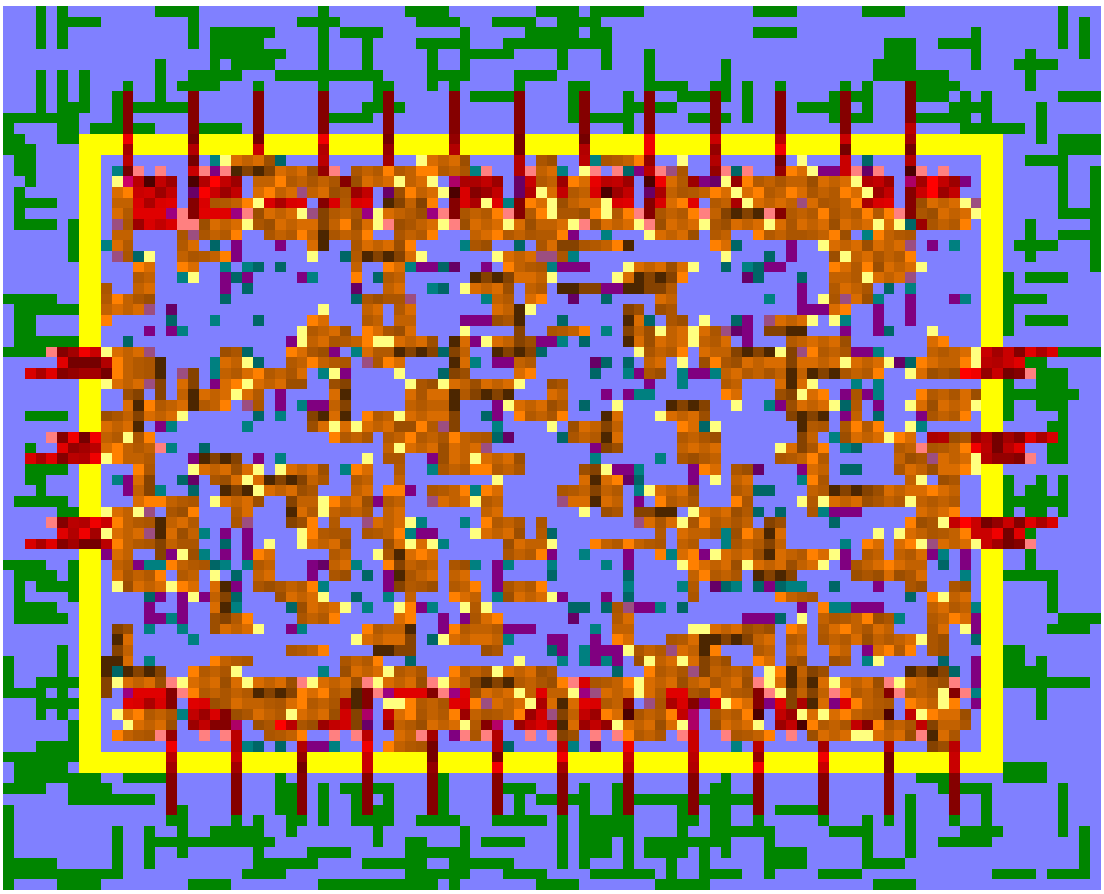
Protein-Based Neural Networks and Their Potential Applications in Building Adaptive Systems

Ken Webb

EASY MSc Dissertation



August 30, 2004

Table of Contents

Abstract

Bray (1990) proposed that protein networks, especially in single-celled organisms such as bacteria, provide some of the same benefits as neural networks in higher organisms. These organisms adaptively move around in their environment, finding food and avoiding dangers, without having any neurons. Bray argued that artificial protein networks, what he called *protein-based neural networks* (PBNN), should be able to do at least some of the same computational and behavioural tasks as artificial neural networks. Very little work has been done since then directly exploring these hypotheses. This dissertation describes one possible silicon implementation of a PBNN, an agent-based (individual-based) approach using simulated monomers that combine to form binding domains, proteins, small molecules, lipid bilayers, and bacteria. This architecture is used to build two quite different applications, one that solves logic gate problems such as AND, OR, and XOR, and the other that acts as a controller for aspects of chemotactic behaviour in bacteria. This work is intended as a proof-of-concept that a simulated PBNN can in fact perform some of the neural network-like tasks that Bray ascribed to them.

Acknowledgements

I would like to express my thanks to Emmet Spier who supervised my work over the summer, and kept me on my toes by challenging many of my ideas.

Introduction

This dissertation is intended as a proof-of-concept for protein-based neural networks (PBNN), an idea proposed over ten years ago (Bray, 1990) but not substantially developed since then. It builds on work begun in my Adaptive Systems project (Webb, 2004) which dealt with the externally-observable behaviour of bacterial chemotaxis, and also provided a simple implementation of internal chemotactic control using a continuous-time recurrent neural network (CTRNN) (Beer, 1995) rather than a PBNN. Using proteins, a PBNN senses chemicals in the environment, and, through internal interactions, adaptively converts this input into an output.

I claim that the implementation described here mirrors the way that biology works, at the levels of individual monomer, binding domain, and polymer. It provides a considerable simplification and abstraction of that biology sufficient to allow computer-based simulation.

In this paper, I first describe the biology behind the concept of a PBNN, then describe my own kPBNN abstraction in considerable detail, and then present two contrasting examples. First, a kPBNN implements AND, OR, XOR and other logic gates, demonstrating that it can perform a type of recognition task often performed using artificial neural networks. In the second example, another kPBNN, using the same set of rules and parameters but different proteins, simulates bacterial chemotaxis, the biological domain that has inspired the PBNN concept, suggesting that a kPBNN can be a useful tool for modelling and simulating biological systems.

The Biology Background

Biological cells and organisms need to take in and metabolize food, but must first find the food. For this and other control purposes they use signal transduction pathways consisting primarily of interacting proteins. A typical pathway senses the presence of a chemical, light, heat, pH or something else in the environment, transduces this signal across the cell membrane, and conveys it to some other part of the cell where it is acted on. (Becker *et al*., 1996, ch.23)

The proteins that do this work are polymers, made up of strings of amino acid monomers held together by strong covalent bonds. The sequence of amino acids in each protein string is specified by a gene containing DNA. The one-dimensional (1D) string folds into a 3D shape which determines its function. There are 20 amino acids, each with its own characteristics. Some amino acids bend relatively easily, especially Glycine. Some are hydrophobic and are found away from water, typically inside a membrane. (Becker *et al*., 1996, ch.3)

Proteins bind with other proteins and with other polymers such as lipid bilayers (membranes) and small molecules, at binding domains, specific regions of the folded structure. If protein A and protein B have compatible binding domains, they have a high probability of binding together if they come in contact with each other. Some proteins are permanently bound to each other, while others bind for a short period and then unbind. (Hancock, 1997)

Binding causes conformational changes in other parts of the protein, one way in which signals progress through a pathway. A small molecule binding to a protein domain that extends beyond the membrane may cause a slight inward movement of a whole section of the protein relative to other parts of itself. This *piston effect* can be detected by other proteins at binding domains inside the cell, which may consequently bind or unbind. Other binding events may cause a more *global conformational change* that effects the alignment of monomers at multiple other binding domains. At the molecular level, the behaviour of a cell is determined through a large number of concurrent pathways, all involving multiple proteins that influence each others shapes through binding and unbinding. (Falke *et al*., 1997)

Protein-Based Neural Networks

Bray (1990, 1995) suggested that single-celled organisms such as bacteria have evolved surprisingly complex behavioural repertoires for organisms that lack neurons. They can't have neurons and networks of neurons because neurons are themselves individual cells. Bacteria depend exclusively on proteins organized into protein networks to adaptively find food and avoid dangers in the environment.

> The imprint of the environment on the concentration and activity of many thousands of proteins in a cell is in effect a memory trace, like a 'random access memory' containing ever-changing information about the cell's surroundings. Because of their high degree of interconnection, systems of interacting proteins act as neural networks trained by evolution to respond appropriately to patterns of extracellular stimuli. The 'wiring' of these networks depends on diffusion-limited encounters between molecules, and for this and other reasons they have unique features not found in conventional computer-based neural networks. (Bray, 1995, p.307).

Bray introduced the concept of what he called a *protein-based neural network* (PBNN). "Because proteins also integrate inputs and produce outputs it seems inescapable that the highly interconnected network of protein-based pathways in living cells will share some of the properties of neural nets" (Bray, 1995, p.312). By analogy with artificial neural networks (ANN) (Bishop, 1995), Bray suggested that artificial PBNNs might be capable of performing useful computational work. As far as I have been able to determine, this idea has never been taken beyond the broad concept stage. Nobody has defined what an artificial PBNN should be like, although several other authors have discussed the same broad concept (Hellingwerf *et*

al., 1995; Agutter & Wheatley, 1997; Wolf *et al.*, 1998). Hellingwerf *et al*. (1995), for example, refer to a 'phospho-neural network', and state that

> Signal transduction in a single living cell has not yet been identified as an example of a neural network, in part perhaps because it falls far short of the highly sophisticated behaviour of the human brain. Yet, essentially all the fundamental mechanistic requirements which classify a system as a neural network are present in prokaryotic signal transduction. (Hellingwerf *et al*. (1995, p.317).

Bray himself seems to still hold with his original ideas.

> From the standpoint of a living cell, the closest approximation to a neural network is probably found in the pathways of intracellular signals. Multiple receptors on the outside of a cell receive sets of stimuli from the environment and relay these through cascades of coupled molecular events to one or a number of target molecules (associated with DNA, for example, or the cytoskeleton). Because of the directed and highly interconnected nature of these reactions, the ensemble as a whole should perform many of the functions commonly seen in neural networks. Thus, in aggregate, the signalling pathways of a cell are capable of recognizing sets of inputs and responding appropriately, with their connection 'strengths' having been selected during evolution. … From what we know of neural nets, it seems that some signalling molecules in the pathway should perform the function of 'hidden units' that embody, in their state of activity, an abstraction of the outside world. It also seems reasonable that the networks of cell signalling reactions should, like highly connected neural networks, be resistant to damage and continue to function even if some of their connections are severed." (Bray, 2003, p.1865).

In this section I propose one possible approach to PBNNs, by specifying a much simplified abstraction of the real biology. I call this abstraction a kPBNN, for Ken's concept of a Protein-Based Neural Network. In later sections I describe a software implementation and two applications.

Bray mentioned a number of important features of PBNNs that are not part of this initial kPBNN prototype. Real cells have a large number of pathways operating in parallel. Cross-talk (Bray, 1990) between these pathways integrates multiple environmental inputs. This parallel activity resembles that found in artificial neural networks which are also often referred to as parallel distributed processing (PDP) networks. The kPBNN applications described in this paper only have a single pathway, although the multiple protein units operate in parallel.

Protein units

The individual unit within a kPBNN simulation is a protein. A protein maintains three types of information about itself, its structure, its relationships, and its behaviour. Its structure is what monomers it is composed of, the string ordering of these monomers, and the 2D shape

into which the string folds. Its relationships are what else if anything it can bind to, or is currently bound to, through its monomer binding domains. Rules that describe its behaviour are attached to its various monomers.

Unlike a traditional ANN, a kPBNN has hierarchical structure. A protein contains monomers, and is itself contained within a lipid bilayer.

Protein structure

Within the kPBNN simulation, a protein monomer is one of twenty amino acids, with functions somewhat analogous to those of biological amino acids. A single monomer fits within a single grid space in the contents of a protein. All monomers are the same size and shape.

A protein maintains its structure within a rectangular grid that is large enough to hold its folded string. The grid has rows and columns, with a grid size equal to rows * columns. The lower left corner, at or near the start of the protein string, is the origin (0,0 point) of the grid. Typically, only some of the spaces within a grid will contain monomers, due to folding.

A protein is essentially a string of covalently-bound amino acid monomers. Covalent binding is the strong binding between the monomers of the string, much as the letters in a word or sentence are much more strongly bound to each other than to the letters in the row of text above or below. An example protein string, in which monomers are represented by letters, is:

```
HRNRGYLNK
```

The monomer G, nominally the Glycine amino acid, has a special role. The 1D string always folds into a 2D structure on finding a G. The above protein string will fold to form the grid structure shown in Figure 1, with numbers shown representing the implicit rows and columns (origin 0,0), and a light string (line) drawn in to represent the covalent connectivity structure.

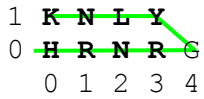


Figure 1 - Example of a folded protein string.

The basic set of twenty amino acids can be augmented through the addition of monomer side chains. Several of the amino acids can be *phosphorylated* by rules, meaning that a side chain (not part of the main covalent string) containing a phosphoryl group can be temporarily added. They can subsequently become unphosphorylated. A methanol side chain can also be temporarily attached through methylation and demethylation rules. The addition or removal of

side chains changes the structure of monomers in the protein's grid, and may change its behaviour.

Protein relationships

Relationships between proteins and between proteins and other polymers, are effected through *binding domains*. A binding domain is a sequence of monomers, in a kPBNN, almost always four monomers long. The folded structure in Figure 1 has potentially two binding domains (if the special-purpose G monomer is ignored), KNLY and HRNR. A binding domain binds if every monomer in it exactly matches a sequence of monomers in another protein. The two proteins can be rotated or flipped relative to each other. Thus KNLY can match YLNK in another protein. If a protein has at least four rows, a column sequence can match a column or row in another protein.

If a rule finds that the binding domains of two proteins match, then they bind together until some other event causes them to become unbound. Protein A can only bind to protein B if B is already bound to something else such as a lipid bilayer. This is a temporary limitation to ease the job of movement in the simulation.

A monomer can only be bound to one other monomer at a time. Thus a four-monomer binding domain can only be bound to one other four-monomer binding domain at a time.

Binding domains are not explicitly defined. Any sequence of four or more monomers in a protein's structural grid is potentially a binding domain. Thus NNNN could bind to three different domains within LNNNNNNK.

A protein can also bind to a matching sequence that spans two or more other proteins. Thus

```
KNLY
```

could match both of

```
  K       L
DRNG    RGYG
```

together if these latter two were properly lined up with each other.

With kPBNNs, anything can interact with anything else as long as the two have matching binding domains and appropriate rules. The human designer of a kPBNN can either explicitly specify all the protein strings, to control what type of relationships will be able to exist, or can leave at least some of the monomer sequences to be randomly generated, perhaps to allow a genetic algorithm (GA) or other evolutionary algorithm to evolve a useful pattern of interaction.

Some monomers are hydrophobic, and prefer to be away from water. If a substring of length four in every row of a protein's grid structure consists of hydrophobic monomers, then it will preferentially bind itself to a lipid bilayer.

Conformational change

A kPBNN uses the bio-molecular concept of conformational change to transmit signals and store state. A kPBNN can "instantly" move a signal from the outside to the inside of a lipid bilayer, a relatively long distance in molecular terms, by moving an entire row of the folded structure of a transmembrane protein. This is an abstraction of the "piston" (Falke, 1997, p.478) found in MCP proteins. Sometimes an external event causes a "global conformational change" (Falke, 1997, p.498) in a protein, in which relative positions of monomers within several binding domains may all be altered at the same time. A kPBNN implements a global conformational change by swapping the positions of two adjacent monomers. Both the piston and swap operations are easily reversible. Both produce an effect that can be detected by other domains within the same protein or within other proteins. Both operations alter the ability to bind. Note that in a kPBNN a conformational change is implemented through a *direct physical change* at the 2D monomer level, rather than through a *change in the value of a symbol* at the protein level.

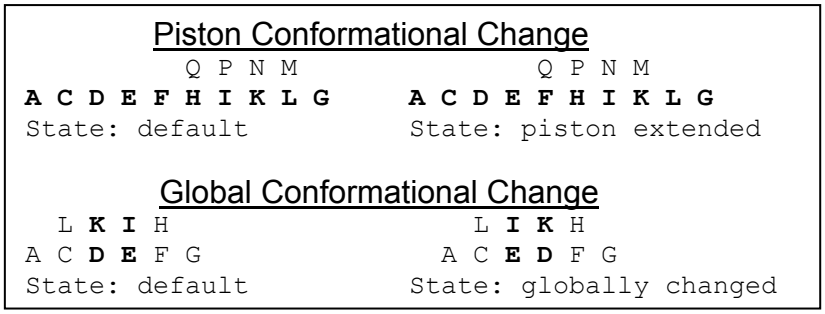


Figure 2 - Two types of conformational change in proteins, as implemented in a kPBNN.

Protein behaviour

A protein behaves by following rules attached to specific monomers in its structural grid. Some rules are explicitly specified by the human designer, while others, such as the hydrophobic rule, are automatically specified as part of the necessary "physics" of the system.

At every step, each protein tries to execute each rule currently associated with any of its monomers. Currently eleven rules are available. Rules are as generic as possible. Column and row positions are always relative to the location of the current rule. A rule may for example instruct a protein to inspect the monomer in the adjacent column, and, if it exists, change the state of that monomer from phosphorylated to unphosphorylated if it is currently phosphorylated. Details of the eleven rules are provided in appendix B.

The protein-protein binding rule implements the concept of relationship between proteins. The protein-membrane binding rule implements the hydrophobic requirement that binds proteins to a lipid bilayer membrane. Seven of the rules have to do with phosphorylation and methylation. The piston binding rule detects small molecule signals (ligands) in the environment, and relays the signal across the lipid bilayer to the inside of the organism. The motor rule turns a simulated rotary motor which could be attached to something outside the organism. These rules provide for sensory perception of the outside environment (inputs), internal processing of that signal, and activation of external actuators (outputs). Each rule can be applied to many different protein structures.

Rules at specific monomer sites can be added, removed or replaced as the simulation is running. A concept of cyclically changing state can be implemented by having two rules swap with each other. If a rule is added, removed, or replaced at any time other than at simulation initialization time, it should be done by another rule.

Small molecules

Small molecules are another type of polymer. They are operated upon by proteins and do not (at present) contain rules. They represent signals in the environment (ligands), and act as internal acceptors and donors of protein side-chains such as phosphoryl and methyl groups. Proteins recognize small molecules in the same way that they recognize other proteins, by matching a sequence of monomers.

A small molecule is a short sequence of monomers all within one row, with no folding. The monomers are the same size and shape as protein monomers, but they have a different set of identifiers and names. Protein and small molecule monomers match if they have corresponding identifiers. Each of the twenty protein amino acid monomer types corresponds to exactly one type of small molecule monomer. Small molecules have the same concept of binding domain as proteins.

Lipid bilayers

Lipid bilayers are another type of polymer. They are continuous sequences of lipid monomers that enclose a space containing other polymers especially proteins and small molecules. They prevent (regulate) polymers in different compartments from mixing together, and allow a stationary binding place for hydrophobic domains. They do not (at present) contain rules. Each lipid is the same size and shape as a protein or small molecule monomer.

A more complete simulation environment would also include two other polymer types, polysaccharides and nucleic acids (DNA/RNA). These consist of monosaccharides (sugars) and nucleotides (C G A T U) respectively, two additional types of monomers.

The kPBNN as a whole

In addition to individual units, artificial neural networks (ANN) also contain explicit connections, and weights on the connections. In a kPBNN, connections are made through the plasticity afforded by binding domains, while weights are probabilities of something happening that are equivalent to kinetic rate constants in biochemistry. The weights in a ANN are typically learned during a series of trials, while the probabilities in a kPBNN could be learned using an evolutionary algorithm. The inputs into an ANN are integer or real-valued numbers, while in a kPBNN they are some number of individual small molecule ligands, possibly of more than one type. Although a kPBNN does not explicitly have the ANN concept of input, hidden and output layers, it does have something roughly analogous with its sensory proteins, internal proteins, and actuator proteins.

In addition to the behaviour rules described so far, there is also an entropy force in the simulation. This force is thought of as not being connected with specific polymers, but as directly under the control of the implied "physics" in the system. Once an MCP binds to a ligand, there is a certain probability every time step that the ligand will become unbound.

Logic Gates application

A logic gate is a device that combines inputs to produce outputs. In a binary logic gate the individual inputs are each either 0 or 1. In the special case in which there are always two inputs, there can be four input types – 00, 01, 10, and 11. Sixteen different logic gates can be constructed, depending on which combination of these four input types are recognized. If the device only recognizes 11 then it is called an AND gate. If it recognizes a 1 at either or both inputs (01, 10, 11) then it is an OR gate. In an XOR gate it only recognizes the situation in which either input is 1 but not both at the same time (01, 10). Table 1 shows truth tables for these three situations.

Table 1 – AND, OR, XOR Logic Gates. Each logic gate has the same inputs, but defines its own unique set of outputs.

AND Gate

Two Inputs	One Output
0 0	0
0 1	0
1 0	0
1 1	1

OR Gate

Two Inputs	One Output
0 0	0
0 1	1
1 0	1
1 1	1

XOR Gate

Two Inputs	One Output
0 0	0
0 1	1
1 0	1
1 1	0

A perceptron is a simple single-layer artificial neural network. It can learn to recognize the AND and OR situations, but is unable to recognize XOR because the inputs are not linearly separable (Bishop, 1995). It requires one internal configuration of weights to recognize 01, and a different configuration for 10, and cannot combine both configurations at the same time.

Bray (1995) pointed out that as with neurons, it is also "true that protein molecules are in principle able to perform a variety of logical or computational operations". He suggests one way in which a protein can "act somewhat like a boolean AND gate .. like an inclusive OR gate, and so on". In building a simple logic application using a kPBNN, I have used Bray's few examples as inspiration, but have not implemented these specific circuits.

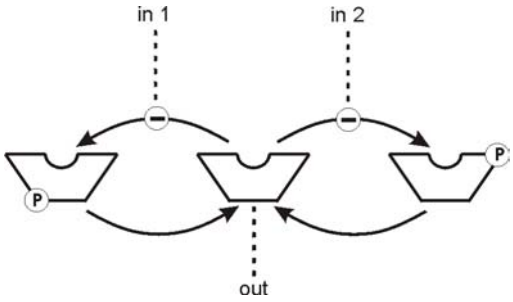


Figure 3 – "A reaction mechanism expected to function as an AND gate. Each of the two inputs ('in 1' and 'in 2') is assumed to inhibit one of the two protein kinases; when both are present, the concentration of A ('out') reaches its highest possible level. Given a suitable selection of rate constants and concentrations, a sharp ON response will be achieved only when both inputs are present. The addition of more enzyme cycles coupled to this one can create other logical devices, such as NOT and OR and XOR" (Bray, 1995, Fig. 2e).
A *kinase* is an enzyme that phosphorylates.

My kPBNN implementation of logic gates is organized into a different type of network than that proposed by Bray in Figure 3. This was done to reuse the piston binding mechanism employed in the bacterial chemotaxis application described later in this paper. Instead of using phosphorylation to change the binary state of a protein (phosphorylated or unphosphorylated), the kPBNN implementation uses a change in the binary state of a protein piston (extended or not extended).

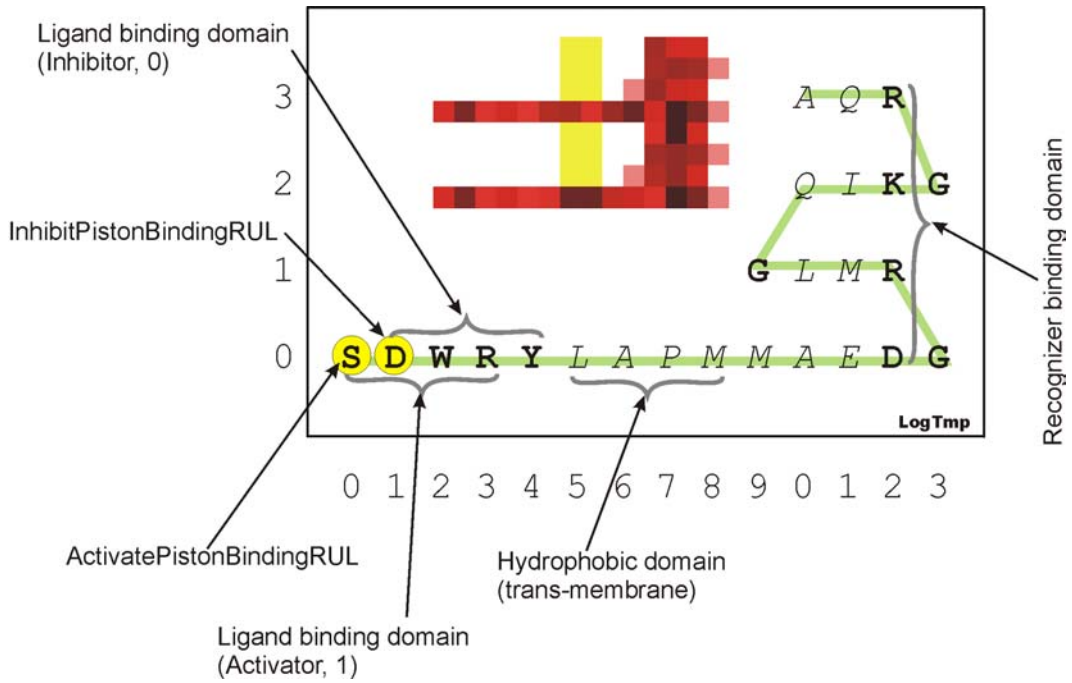


Figure 4 – Logic gates application. Transmembrane protein (TMP). The coloured inset shows two TMPs right up next to each other, with part of the lipid bilayer (yellow) that the TMP pistons extend through. The protein string is a sequence of monomer (amino acid) symbols starting with SD and ending with QA. The green line behind the monomers represents the covalent bonds. G at the end of each substring is the instruction to fold. The folded string forms a shape with 4 rows and 14 columns. The piston is the entire sequence in row 0. Monomer symbols in bold are explicitly specified, while those italicized are complete or partial (hydrophobic) wild cards. All binding domains are labelled. The locations of protein interaction rules are circled in yellow and are labelled. There are two ligand binding domains that overlap each other. The ligand activator SDWR binds to one of these, while the ligand inhibitor DWRY binds to the other. The two ligands share the symbols DWR. When one ligand binds the other one is prevented from doing so (competitive binding). The recognizer binding domain on the right (DRKR) is a column domain that crosses rows, and it may also continue into the adjacent second TMP. If the piston is extended when an activator ligand binds to it, then the recognizer binding domain will contain ERKR rather than DRKR. E moves to column 2 and D to 3.

In the kPBNN implementation, the two types of input (1 and 0) are represented by two different small molecules (ligands) in the local environment. The two input gates are represented by two separate transmembrane proteins (TMPs), placed adjacent to each other forming one continuous binding domain (Figure 4). The single output is represented as one or more instances of a second protein type that is free to move within the cytoplasm(Figure 5).

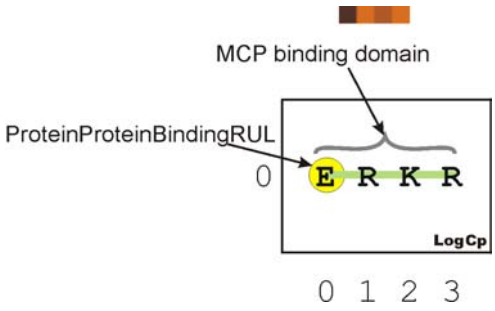


Figure 5 – Logic gates application. OR recognizer protein. If the piston in the MCP is extended, then the OR recognizer can bind with the MCP at its ERKR recognizer binding domain. The protein-protein binding rule contains the behaviour that searches for monomers already in the grid that match the four symbols starting from its position, which happen to be ERKR in this case.

Figure 6 shows a configuration in which the 0 and 1 input (each ligand is of a different type) produces a conformational change to one of the TMPs. This enables the OR recognizer protein to bind to the TMP complex. The OR recognizer can also bind if both input ligands are of type 1. The OR string, as shown in Table 2, is **ERKR** which matches **ERKR**DRKR (10 input), DRKR**ERKR** (01), or **ERKRERKR** (11), but not DRKRDRKR (00).

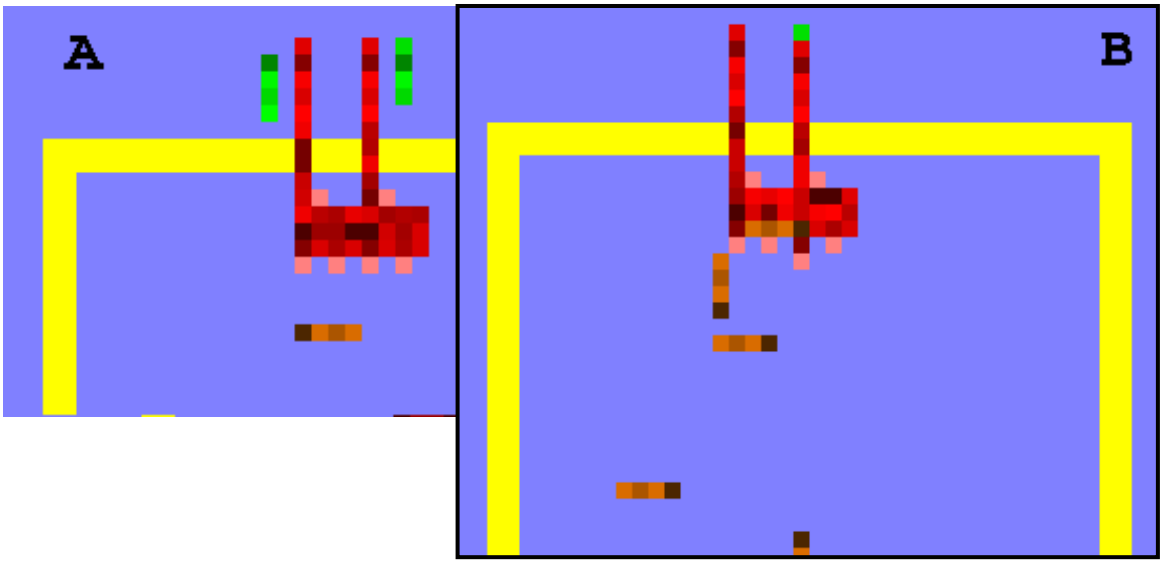


Figure 6 – Logic Gates application. OR recognizer with inputs of 01. **(A)** Time step 0. The two pistons are the long sections of the two transmembrane proteins (TMPs) (reddish) that extend from the environment, across the lipid bilayer (yellow). The green ligand on the left, in the environment, is a 0 and will bind with the leftmost TMP. As a 0 it will prevent any other ligand from binding to that TMP, and will inhibit the piston from extending. The green ligand on the right is a 1 that will bind with the other TMP and will activate its piston. The two ligands will only bind when their random movements place them directly under the TMP binding domain. However, in all experimental trials, both ligands were initially placed directly beneath the TMP binding domain, forcing a binding in time step 0. Both TMP pistons are initially not extended. Several instances of the OR recognizer, initially all in the same location, are shown (brownish). **(B)** Just after one of the OR recognizers has bound to the TMPs. The 0 ligand is beneath the non-extended left TMP. The 1 ligand has bound to the right TMP and caused the piston to extend. The altered monomer alignment of the TMPs has allowed one of the OR recognizers to bind. By extending one position, the very dark monomer in the right piston can now match the dark monomer at the end of the OR recognizer.

An XOR situation is handled in the same way. One input ligand must be of type 0, the other of type 1. One will bind to the left TMP and the other to the right TMP, producing, on separate trial runs, either 01 or 10. In either of these two cases, but not in any other, the XOR recognizer protein ERKRD will bind to the combination of TMPs. It can bind to 10 if unrotated, or to 01 if it becomes rotated 180° in which case it becomes DRKRE. But no possible rotation will allow it to recognize 11 as inputs.

Logic application experimental results

Table 2 summarizes the experimental results. Each trial was run for up to 200 time steps (negative success), or until the recognizer for that logic type found a match (positive success).

If achieved, positive success always occurred within 50 time steps. A plus (+) or minus (-) in each of the final four columns of the table, indicates that the recognizer either successfully found a match (+) or successfully failed to match (-). The program indicates a match by beeping and displaying either of two messages:

```
doRule_ProteinProteinBinding Matched: Recognizer to TmpA
doRule_ProteinProteinBinding Matched: Recognizer to TmpB
```

All tests were successful, as confirmed by the ⩾`symbol`.

Table 2 – Logic Types application– Details and Results. For each of the 16 possible logic gates (LogicType), the recognizer string and its graphical appearance are given. AND, OR and XOR are explicitly identified. Positive (+) and negative (-) success is shown, and a check mark indicates successful completion of each of the 00 01 10 and 11 trials for that logic gate. All trials were successful.

LogicType	RecognizerString	RecognizerShape	ResultsForInputs: 00	01	10	11	
0000	DDDDD		–	–	–	–	⩾
0001 AND	ERKRE		–	–	–	+	⩾
0010	wERKRD		–	–	+	–	⩾
0011	wERKR		–	–	+	+	⩾
0100	wDRKRE		–	+	–	–	⩾
0101	ERKRw		–	+	–	+	⩾
0110 XOR	ERKRD		–	+	+	–	⩾
0111 OR	ERKR		–	+	+	+	⩾
1000	DRKRD		+	–	–	–	⩾
1001	DRKRDqERKRE		+	–	–	+	⩾
1010	DRKRw		+	–	+	–	⩾
1011	DRKRDqRKREw		+	–	+	+	⩾
1100	wDRKR		+	+	–	–	⩾
1101	ERKREqRKRDw		+	+	–	+	⩾
1110	DRKR		+	+	+	–	⩾
1111	RKR		+	+	+	+	⩾

Learning in the kPBNN logic gates application

Currently the kPBNN is pre-configured and does not learn how to recognize trial sets of inputs. What mechanism might it use to learn? Bray (1990, p.226) suggests "that the closest analogy to the learning algorithm in a PDP network is evolution". He proposed an algorithm, essentially a hill climber, that randomly changes the network each generation, always retaining the better of the two latest networks tried. He seems unaware of previous work on evolutionary algorithms (Mitchell, 1997) that involve a population with crossover and/or mutation, although he does speculate that "networks might be allowed to 'propagate' until they form a small population of networks before selection" (Bray, 1990, p.228).

The kPBNN recognizer protein could be evolved using a genetic algorithm (GA). Protein strings would be generated randomly within a population of "organisms" each with its own

lipid bilayer, TMPs, and evolving recognizers. Each trial could run for 200 time steps, at which time crossover, mutation and selection of the next generation would take place. The recognizer is just a string of characters taken from an alphabet containing six symbols – D E K R q w. Strings can contain between three and eleven characters. The number of possible combinations of six symbols taken 3 to 11 at a time is $6^{11} + 6^{10} + 6^9 + 6^8 + 6^7 + 6^6 + 6^5 + 6^4 + 6^3 = 435{,}356{,}424$. The actual search space is at least several orders of magnitude smaller given that every recognizer, with the possible exception of that for 0000, must contain the substring RKR, and that the optional q and w monomers occupy specific positions.

An issue is what to use as a fitness function. The recognizer either matches an input configuration, or it doesn't. In this case only random mutation is effective, and selection is just the ability to pick out a successful member of the population. In the three cases in Table 2 where the recognizer strings are eleven symbols long, a type of crossover or concatenation genetic operator could be used to combine two partial solutions.

An evolutionary algorithm, constructed to take advantage of all the constraints built into this problem should be able to learn to solve each of the 16 logic gage problems quite quickly. But a simpler algorithm that generates random strings using the constraints and with the possibility of bolting together partial solutions, should do about as well. It is possible that either of these approaches would automatically discover better (shorter strings, or taking less time to match) solutions than I have been able to discover manually.

In summary, this implementation of a kPBNN is able to solve all 16 logic gates without making use of the hidden unit layer required in traditional artificial neural networks, through the use of two features that each combine two binding domains into one. It can rotate its string by 180°, and it can concatenate two substrings.

Bacterial Chemotaxis

E. coli bacteria can follow chemical gradients in their environment. The chemicals or ligands are either attractants (food) or repellents (noxious chemicals). The bacteria use receptors to sense the environment, rotary motors (6 to 8 per bacterium) to turn flagella that propel the bacteria, and an internal protein network that adaptively converts the sensed inputs into motor outputs. If all motors turn counterclockwise, the bacterium swims in a straight line, called a *run*. If at least one motor is turning clockwise, the bacterium *tumbles* and finds a new direction. If there is no ligand attractant in the environment, the bacterium exhibits run behaviour about 60% of the time, and tumble behaviour the other 40%. If there is ligand it will spend a greater percentage of time running, and will tend to move up the gradient toward the source of ligand. If the ligand exists for a short time in a constant concentration, the bacterium adapts to that constant higher level, and returns to the 60:40 ratio of runs and

tumbles. It moves up a gradient by continuously adapting to higher and higher levels allowing it to again sense a difference between a previous level and the current level of ligand. (Bray, 2001)

Falke *et al*. (1997) divide the internal chemosensory pathway into two branches, a phosphosignaling branch, and an adaptation branch. A network of proteins implements each of these branches (Figure 7).

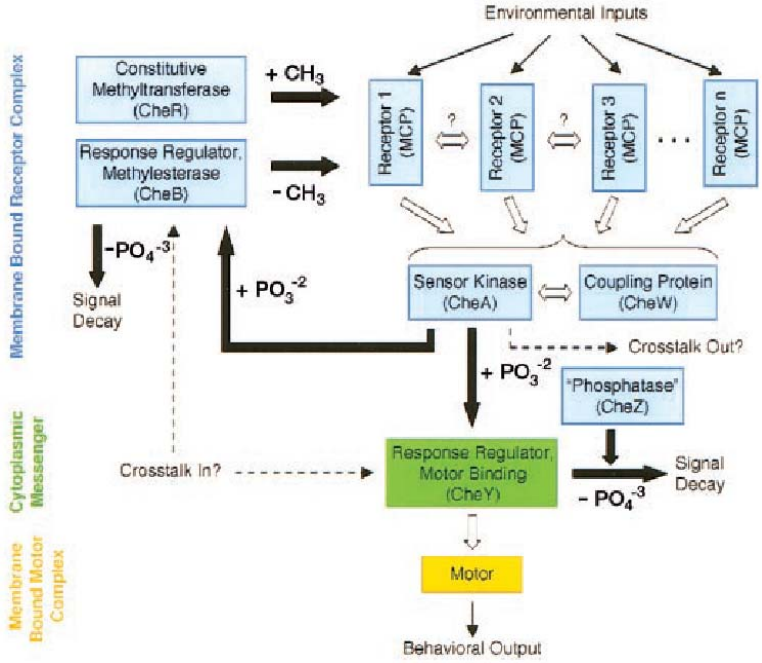


Figure 7 – Chemosensory signalling complex. (Falke, 1997, Fig. 6). This diagram shows the complete signal transduction pathway and all the proteins in the kPBNN system. The MCP receptor proteins bind to the environmental inputs (ligands). CheA and CheW are permanently bound to the MCPs. CheA phosphorylates (+ PO) CheY and CheB. The arrow leading from CheA to CheY is part of the phosphosignaling branch, while the arrow to CheB is part of the adaptation branch. CheY~P (phosphorylated CheY) can bind to a motor, while CheB~P can demethylate (- CH) the MCPs. CheR continuously methylates MCPs. CheB~P and CheY~P spontaneously autodephosphorylate (- PO). The protein enzyme CheZ also dephosphorylates free CheY~P.

Phosphosignaling branch

This section describes each polymer in the phosphosignaling branch.

Ligands

E. coli recognizes some 50 small molecules in its environment. Some of these ligands such as aspartate and glucose are attractants and provide food, while others such as phenol and nickel are repellents. The goal of chemotaxis is to swim towards concentrations of the attractants and away from the repellents. (Macnab, 1987)

The kPBNN simulation only models attractants. All attractants have the same effect, and at present the simulation has no way of distinguishing one type from another. The attractant used is nominally aspartate. Asparate and other ligands are all four monomers long in the simulation.

MCPs (Tar, Tsr, Trg, Tap)

E. coli have five or six different types of chemoreceptors called methyl-accepting chemotaxis proteins (MCP), all with very similar structures, especially the domains which are inside the cytoplasm (Figure 8). "Because each of the chemoreceptors binds the same CheW and CheA proteins, their cytoplasmic domains are highly conserved and exhibit pairwise sequence identities as high as 85%" (Falke *et al.*, 1997, p.482). Four of these have been implemented in the simulation, and only differ in the four monomers used to recognize ligands. The simulation trials all use a single MCP type Tar which recognizes aspartate (Figure 9).

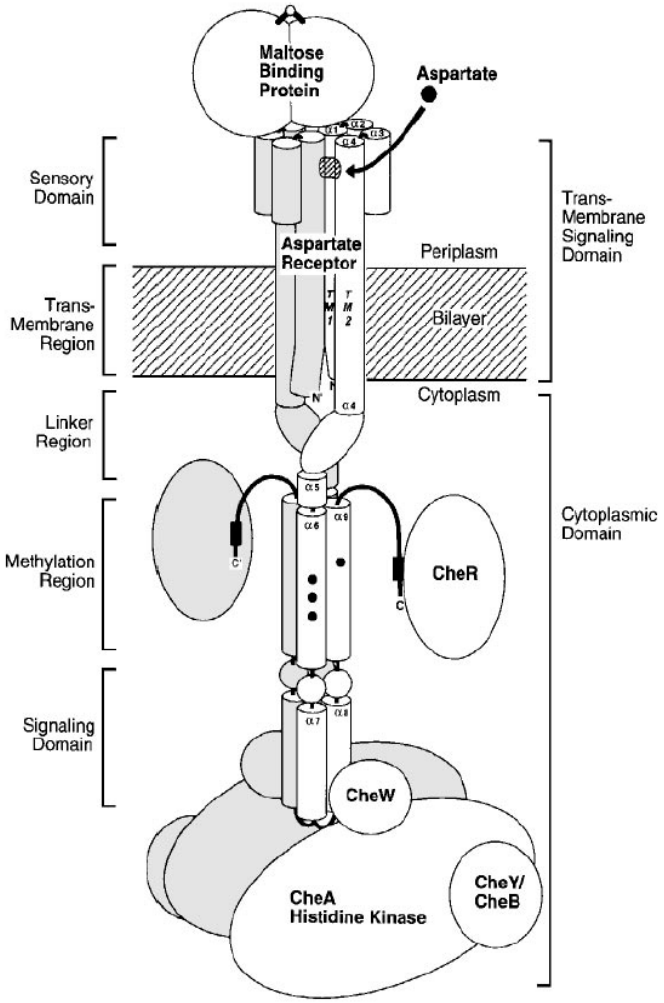


Figure 8 – Chemosensory signalling complex. (Falke, 1997, Fig. 6). The Tar MCP is the whole complex of tube like shapes in the figure. The piston is one of the long complexes on the MCP, extending through the bilayer from the sensory domain to the signalling domain. When an aspartate ligand binds at the sensory domain, the piston moves slightly downward, effecting CheA and its

ability to phosphorylate CheY and CheB. CheW and CheA are permanently bound to the MCP. CheY and CheB only bind until they become phosphorylated.

An MCP binds to a ligand in the environment. This causes its piston to extend, which causes a global conformational change to CheA bound to the MCP inside the cytoplasm. The ligand spontaneously unbinds which reverses the effect on CheA (Falke *et al.*, 1997). At the same time CheR and CheB~P respectively increase and decrease the number of methyl groups attached to the MCP, thus regulating the rate at which CheA autophosphorylation occurs (Bren & Eisenbach, 2000).

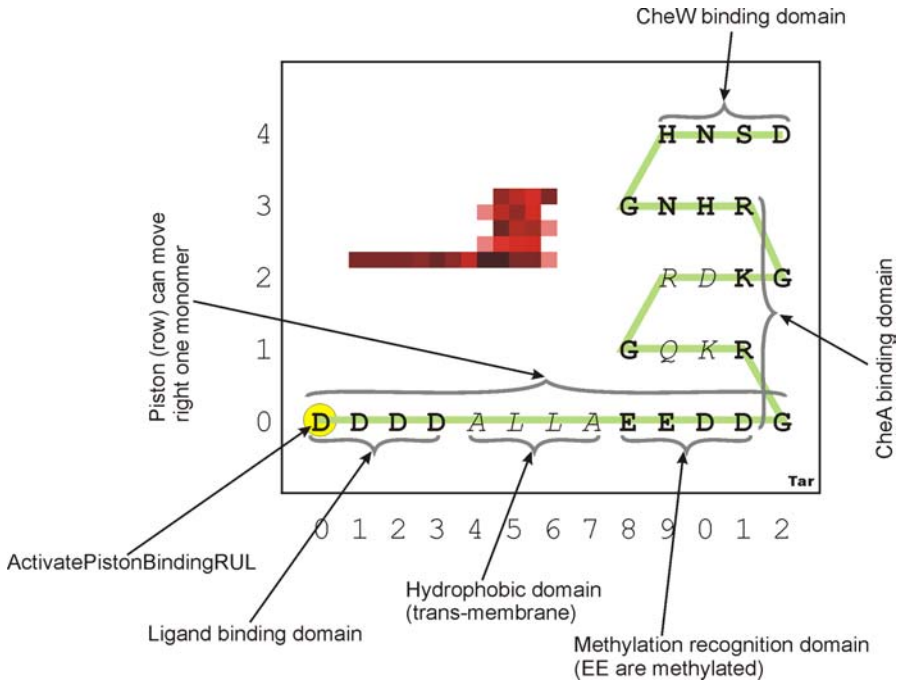


Figure 9 - Tar MCP. This is the kPBNN abstraction that corresponds with the Tar MCP in **Figure 8**. The ligand binding domain, hydrophobic domain, methylation recognition domain, and CheA binding domain correspond respectively with the sensory domain, trans-membrane domain, methylation region, and signalling domain in that previous figure. The piston can move one position to the right. The MCP only has one rule, the one that tries to bind with a ligand that matches DDDD (aspartate). The protein's monomer string, starting with DD and ending at SD, is folded four times, whenever there's a G monomer. The reddish coloured inset shows what an MCP looks like graphically while the kPBNN simulation is running.

CheW

CheW is a structural protein that has no real function in the kPBNN system, but that in a real bacterium binds CheA to an MCP.

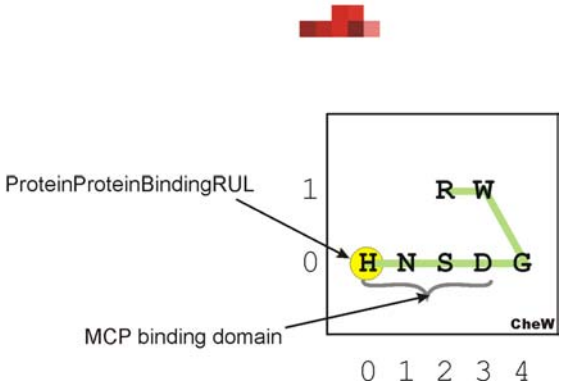


Figure 10 – CheW. This protein has one rule that tries to match the contents of its MCP binding domain with monomers in the grid. HNSD matches the CheW binding domain on the MCP.

CheA

CheA spontaneously autophosphorylates to become CheA~P by picking up a phosphoryl group from ATP small molecules diffusing in the cytoplasm. If a CheY (or CheB) protein is bound to it, the CheA~P very rapidly passes the phosphoryl group on to it resulting in the formation of ChY~P (or CheB~P) which then unbinds from the CheA. This cycle runs continuously (Falke, 1997). The autophosphorylation event is inhibited by ligand binding to the MCP, which results in less CheY~P, and a consequent higher rate of motor counterclockwise movement, and a higher proportion of runs (Bren & Eisenbach, 2000). Thus, "chemoattractants increase the likelihood of straight runs" (Rao *et al.*, 2004, p.240), and the rate of CheA autophosphorylation is a key factor in this.

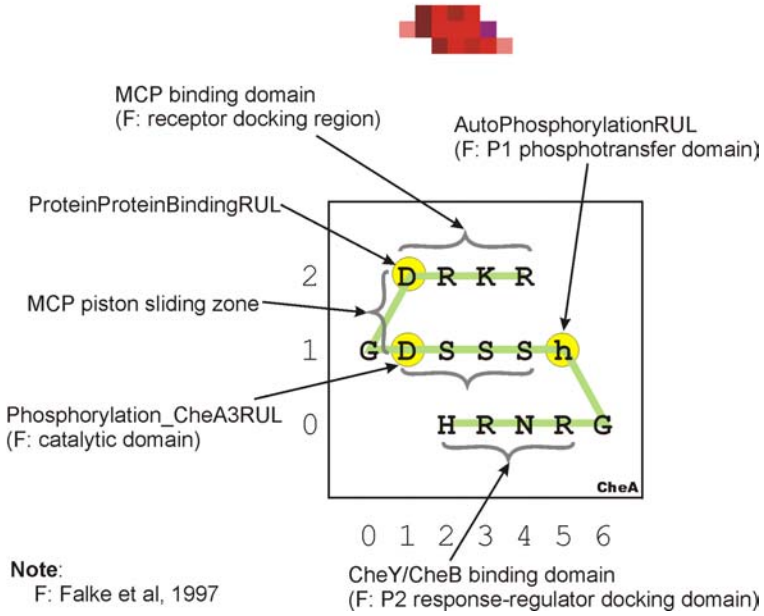


Figure 11 – CheA. Although this protein, both in a real bacterium and in the kPBNN simulation, is quite complex, the "architecture of CheA is highly modular" (Falke *et al.*, 1997, p.490). Falke *et al.* (1997) identify four binding domains, all four of which are included in the simulation. The Falke names are given in parentheses in the figure. The MCP binding domain is where it permanently binds to the MCP. The CheY/CheB binding domain is where unphosphorylated CheY and CheB competitively bind. The Falke P1 phosphotransfer domain is where the phosphoryl group binds and where the AutoPhosphorylationRUL rule is placed in the simulation. This rule only transfers a phosphoryl group if there is an ATP, ADP, or AMP small molecule colocated in the cytoplasm. The Falke catalytic domain is the site of the simulation Phosphorylation_CheA3RUL rule that transfers the phosphoryl group from the phosphotransfer domain to a bound CheY or CheB if any.

CheY

CheY is a messenger protein that shuttles between the chemosensory complex (MCP, CheA, CheW) and the motor complex (Bren & Eisenbach, 2000). When dephosphorylated it binds to CheA. When CheA phosphorylates it, the CheY unbinds, diffuses through the cytoplasm, and eventually binds with the motor complex thus delivering its message. When it spontaneously dephosphorylates, it unbinds from the motor complex and diffuses back to again bind with CheA (Falke *et al.*, 1997). CheY is also actively dephosphorylated by CheZ.

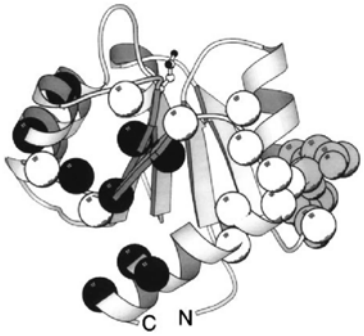


Figure 12 – The modular structure of CheY. Black spheres are the CheA binding domain. White spheres are the motor binding domain. Grey spheres are the CheZ binding domain. The ball-and-stick side chain is the phosphorylation site. (Falke *et al.*, 1997, Fig. 14c)

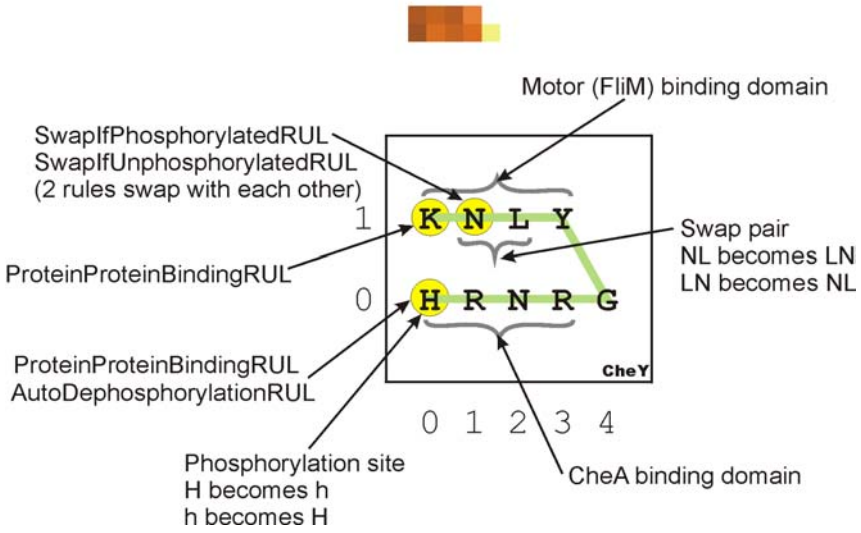


Figure 13 – CheY. This shows the folded string used in the simulation to represent CheY. Nine monomers sequenced as the string HRNRGKNLY are organized into two binding domains. The CheA binding domain (HRNR) binds to the CheA protein, and the Motor (FliM) binding domain (KNLY) binds to the motor complex. The phosphorylation site H at position 0,0 is phosphorylated (H → h) by CheA, and dephosphorylated (h → H) by CheZ and through temporal decay. The rules are shown in the figure as circles around the monomer to which they are associated with RUL at the end of their names. The current state of the CheY protein is captured by swapping the two rules, by reversing the order of NL, and by interconverting h and H. This allows each binding domain to operate independently, each a module with its own instance of the protein-protein binding rule.

An individual instance of a CheY protein knows exactly three different things, each of which it represents by the binary state of a specific domain through the monomers in that domain. It knows if it is phosphorylated or not. It knows if it is bound to a CheA protein in a receptor complex. It knows if it is bound to a FliM binding domain of a motor complex. Of the eight (2 × 2 × 2) potential combinations, only four are actually possible, those shown in Table 3 with an asterisk (*).

Table 3 - Valid CheY states.

	Phosphorylated	Bound to CheA	Bound to Motor	Notes
0*	0	0	0	freely moving in cytoplasm
1*	1	0	0	freely moving in cytoplasm
2*	0	1	0	OK
3	1	1	0	-----
4	0	0	1	-----
5*	1	0	1	OK
6	0	1	1	-----
7	1	1	1	-----

Figure 14 shows each of these four valid combinations as a separate state, and shows the various transitions between states.

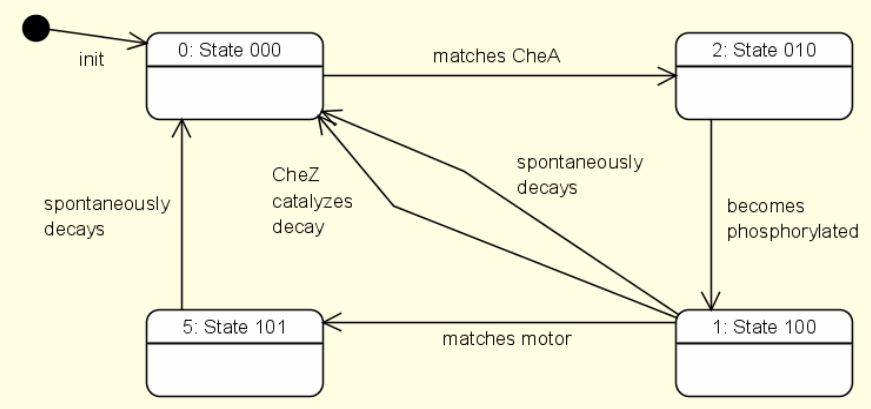


Figure 14 – CheY. State diagram at the protein level.

Nature is unable to directly implement this state diagram, because of the requirements that everything be physically instantiated and adjacent, and that there be no meta-level (protein) states beyond those inherent in the monomers that make up the protein. Note that this state diagram is for the protein as a whole.

There are many ways to produce a concrete instantiation of the abstract behaviour shown in Figure 14. A protein-based neural network must be a physical instantiation that meets the physical, chemical and biological requirements found in real proteins. All states must be explicitly coded within the monomers that make up each individual protein. In the real world there are no meta-states, or at least so it is assumed here. If a protein is phosphorylated, this is so precisely and only because it contains a monomer that has a phosphoryl group attached to it. It does not have a separate state at the protein level called "phosphorylation state". A kPBNN implementation should by definition operate this way. This is true of my C++ simulation program in which the isPhosphoryl() member function on the Protein class returns true if the specified monomer has a phosphorylated group attached to it, and false otherwise.

The kPBNN physical instantiation also requires direct physical effects. An event at one monomer or domain (a short sequence of adjacent monomers) has an effect by altering some physical attribute of monomers that are in some sense adjacent to it. Disconnected action at a distance is not possible, such as is standardly done in conventional software using variables.

CheZ

CheZ proteins diffuse freely in the cytoplasm. They bind to free CheY~P proteins in the cytoplasm, but not those bound to a motor complex (Bren & Eisenbach, 2000). CheZ removes phosphoryl groups from CheY~P as a way of regulating the amount of free CheY~P in the

cytoplasm. There is a "high probability that CheY~P will be dephosphorylated before it travels far from its origin. … Perhaps only a sustained signal can produce enough CheY~P to overwhelm CheZ and get through to the motors" (Bourret & Stock, 2002, p.9627).

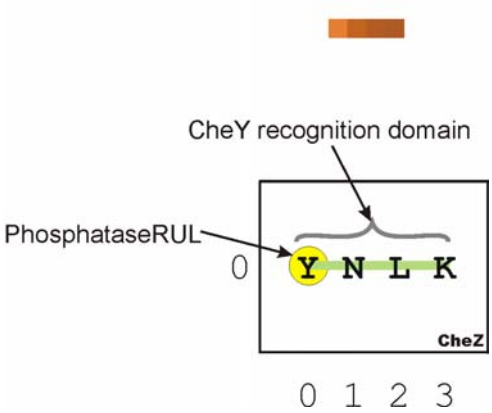


Figure 15 – CheZ. In the simulation, the CheY recognition domain matches the YNLK motor (FliM) binding domain of CheY~P (phosphorylated CheY). The PhosphataseRUL rule transfers the phosphoryl group from the CheY~P to a ADP, AMP or adenosine small molecule, but only if one of these molecules is colocated with the CheZ and CheY~P.

Motor Complex

The motor complex consists of multiple copies of six different protein types, in total over 200 different molecules (Bren & Eisenbach, 2000). Among these are about 35 copies of the FliM protein to which CheY~P binds. In the kPBNN simulation, the motor complex is treated as if it were one single protein, but with binding sites for up to three copies of CheY~P.

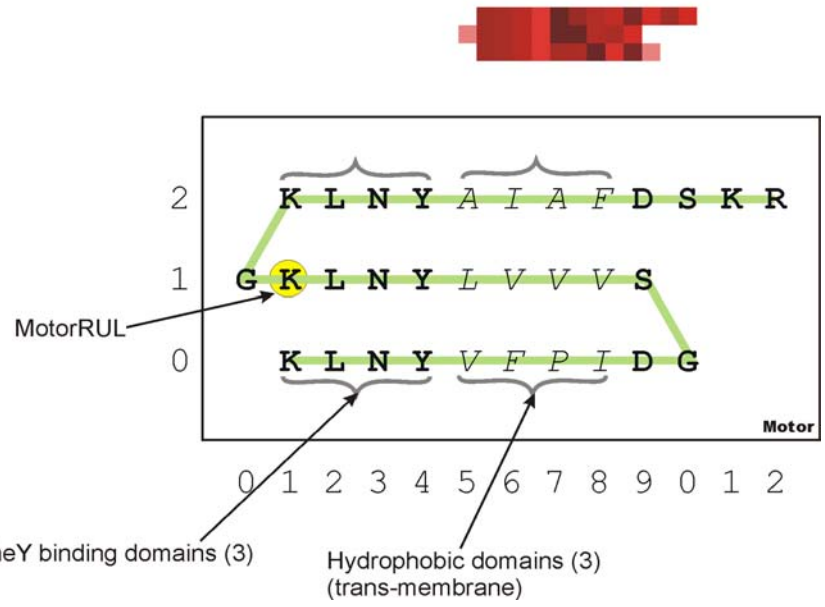


Figure 16 – Motor complex. Up to three CheY~P proteins can each bind to one of the three CheY binding domains. The three parallel hydrophobic domains allow the complex to span the lipid bilayer membrane. The DSKR domain that sticks out into the environment currently has no role, but in a complete simulation it would be connected to a long flagellum that would be turned by the motor to propel the bacterium.

Adaptation branch

The adaptation branch of the chemotaxis pathway involves many of the same proteins as the phosphosignaling branch.

MCP methylation binding domains

Figure 8 and Figure 9 show the MCP methylation binding domains, recognized by both CheR and CheB proteins. In the simulation, this domain is implemented as EEDD. Four methyl groups can be added to each E monomer, for a total of eight. The level of methylation effects the rates of CheA, CheB~P, and CheR activity (Rao *et al.*, 2004, p.240-241). Also, a "high methylation level decreases the affinity of the receptor supramolecular complex to attractants" (Bren & Eisenbach, 2000, p.6870). Methylation provides feedback enabling the bacterium to adapt to a higher level of ligand in the environment, "so that it can chemotax even in a concentration gradient superimposed on a large constant level of attractant or repellent. … The receptor methylation level also provides a simple chemical memory used to ascertain whether the current direction of swimming is favourable or unfavourable" (Falke *et al.*, 1997, p.465).

CheB

The CheB protein competes with CheY for the same binding domain on CheA. If phosporylated by CheA, it can subsequently recognize the methylation binding domain on MCP proteins, and can transfer a methyl group to the MCP.

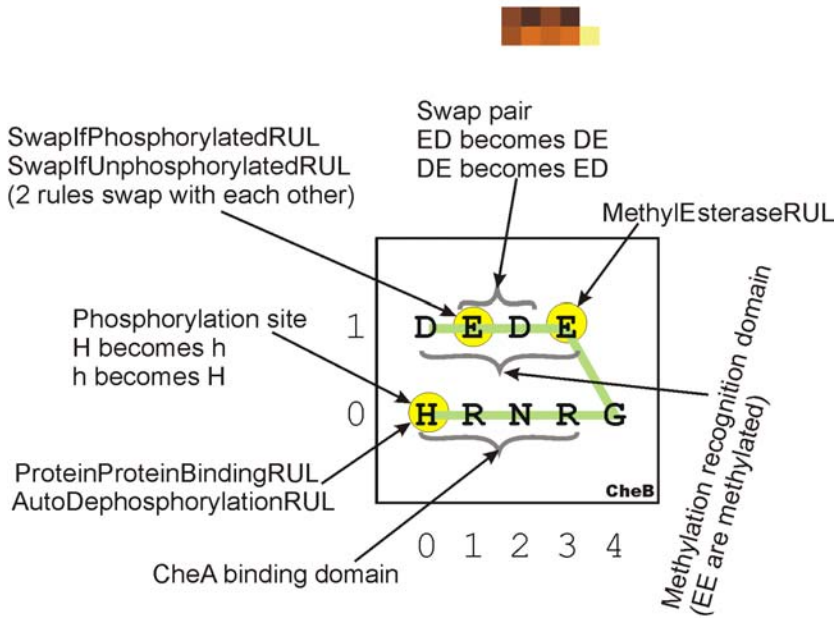


Figure 17 – CheB. In the simulation, this protein contains some of the same rules and concepts that are used with the CheY protein, because they both bind to the same CheA binding domain. CheB~P recognizes and demethylates the methylation domain on MCP proteins. When phosphorylated by CheA, it's methylation recognition domain undergoes a conformational change (a swap) that converts DEDE into DDEE which then matches the EEDD domain on the MCP. CheB~P spontaneously dephosphorylates back into CheB.

CheR

The CheR protein is always present in the cytoplasm, and continuously methylates the methylation domain of MCP proteins. It only has one state and cannot be deactivated. Its effects are regulated by the rate at which CheB~P is formed and demethylates the MCPs.

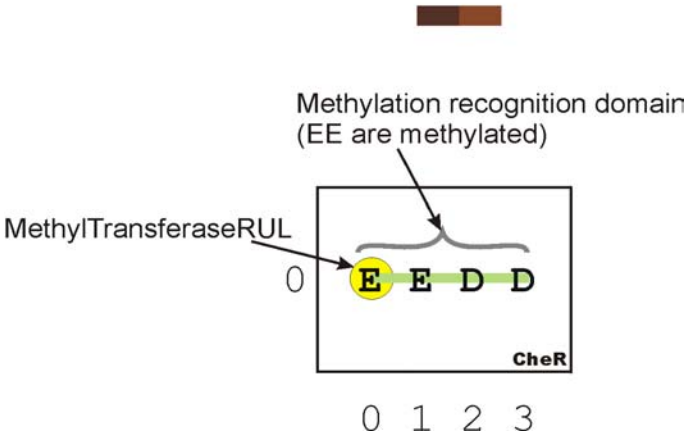


Figure 18 – CheR. Its methylation recognition domain matches the EEDD on the MCP.

Knock-out experiments

Biologists use gene knock-out experiments to determine which proteins are essential for specific cell functions. When a gene is removed the protein it codes for will no longer be expressed and the cell will lack copies of that protein. This section describes what happens when the bacterial chemotaxis simulation is run with the quantity of one specific protein or ligand set to zero, effectively knocked out, while all others are maintained at the default levels shown in Table 4.

Table 4 - Parameters used during the knock-out experiments. The amount of ligand is actually 20 times the specified value or 400. The other quantities are the number of units of each protein used in the simulation.

Ligand	MCP	CheW	CheA	CheY	Motor	CheB	CheR	CheZ
20 (400)	4	4	4	100	6	10	13	30

No ligands

Ligands are not proteins so this is not a true knock-out experiment. With the number of ligands set to 0, the bacterium swims through an environment consisting exclusively of water. Under these conditions, *E. coli* in the laboratory exhibit their default behaviour of about 60% runs, and 40% tumbles.

When calibrated (see below under *Motor calibration*), the kPBNN simulation exhibits this characteristic behaviour (Figure 20). The measured quantities maintain a mean constant level once they have reached a steady state (Figure 19).

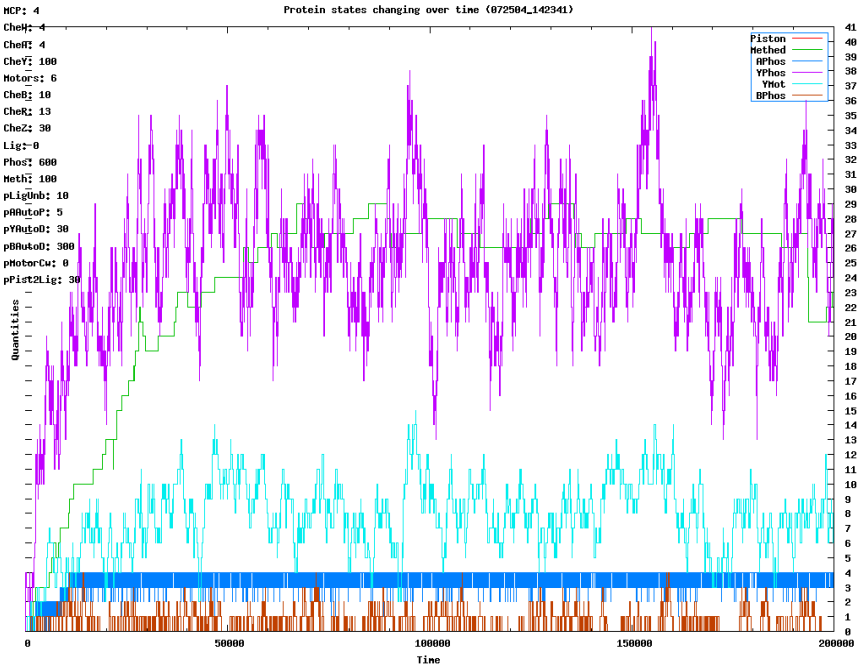


Figure 19 – Pseudo knock-out experiment in which the number of ligands is 0 (Lig: 0).

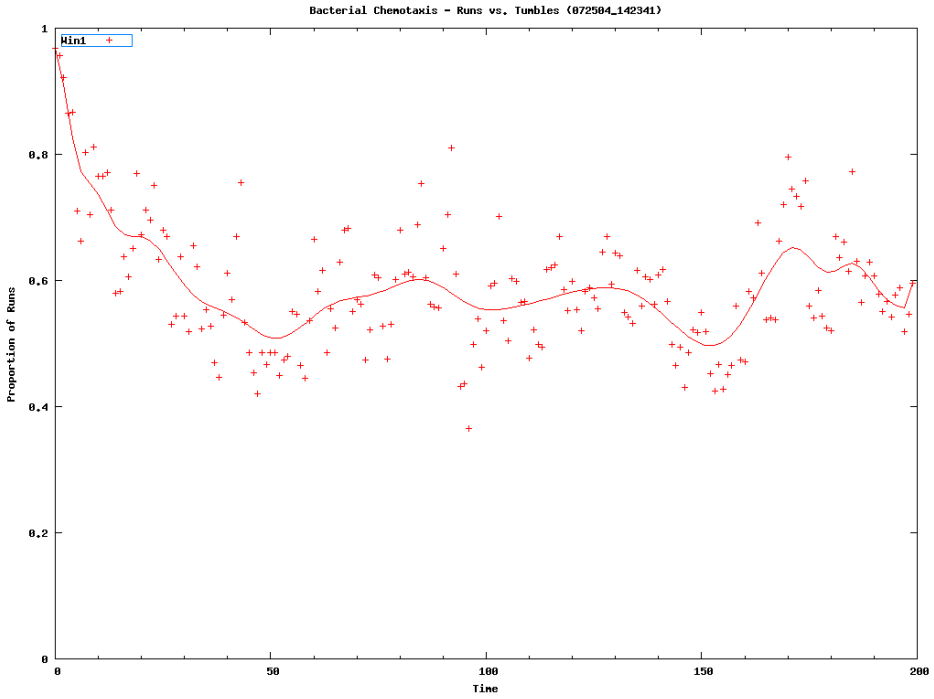


Figure 20 – Pseudo knock-out experiment in which the number of ligands is 0 (Lig: 0).

MCP knock-out (Tar, Tsr, Trg, Tap)

A bacterium with no chemoreceptors (MCPs) cannot detect conditions in its environment. It is unable to form chemosensory complexes because CheW, CheA, CheY, CheB, and CheR proteins have nothing to bind to. This completely disrupts the entire chemotaxis pathway.

When all four MCPs are knocked-out in the simulation, no CheY~P forms, the motors turn CCW 100% of the time, and the bacterium runs continuously in a straight line. The CheW and CheA proteins move randomly within the cytoplasm.

Figure 21 and Figure 22 show that the bacterium was in runs mode 100% (1.0) of the time, and that all measured quantities had constant values of zero.

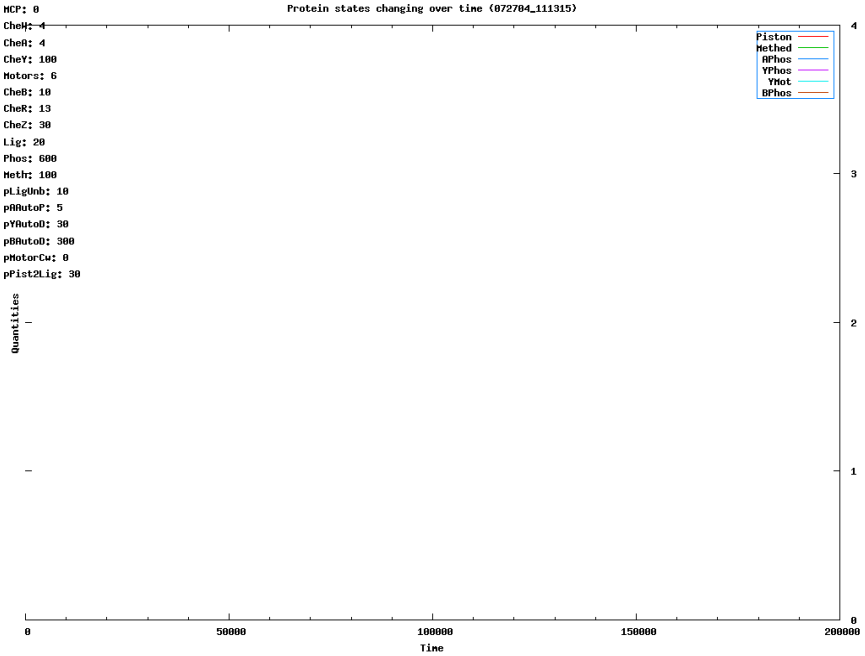


Figure 21 – Knock-out experiment with quantity of MCP set to 0. All measured quantities are 0.

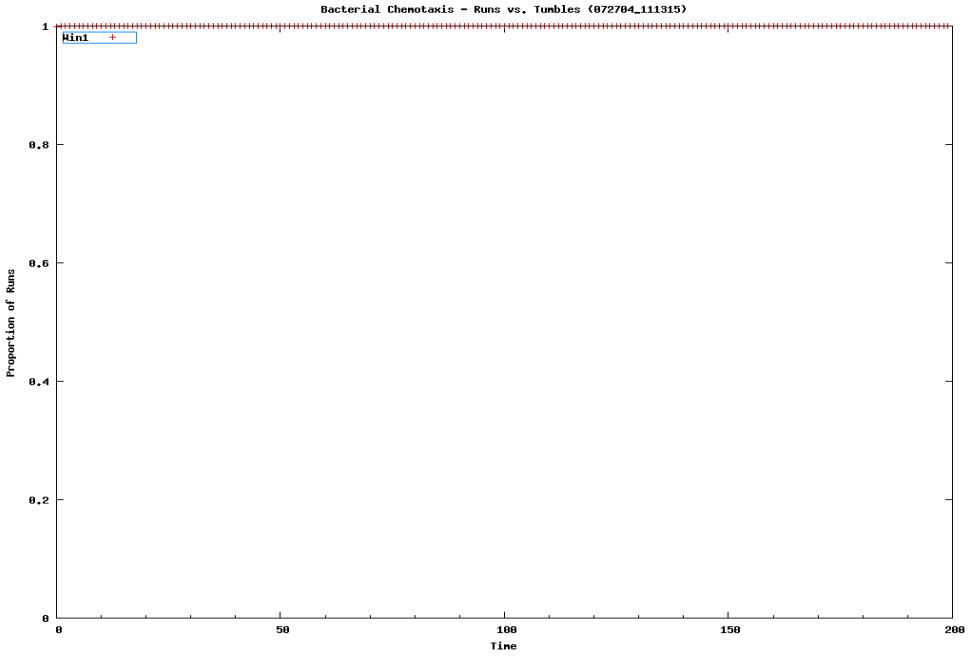


Figure 22 – Knock-out experiment with quantity of MCP set to 0. The red crosses at the top of the graph show that the bacterium spends 100% of its time in runs behaviour.

CheA knock-out

If CheA is knocked out there can be no phosphorylation of either CheY or CheB, thus disrupting both the phosphosignaling and adaptation branches of the chemosensory pathway. In the simulation, the motors should turn CCW 100% of the time, and the methylation level

should climb to 100% of its maximum value 32. This is because there will be no CheY~P to bias the motors, and no CheB~P to counteract the methylation activity of CheR.

Figure 23 also shows that there is less piston activity as the level of methylation increases. Fewer ligands bind to the MCP chemoreceptor once the methylation level moves above 24 (around ts = 52000). This result is predicted by Bren & Eisenbach (2000, p.6870) statement that a "high methylation level decreases the affinity of the receptor supramolecular complex to attractants … up to 10,000-fold for serine … suggesting that the methylation level regulates ligand binding to receptor supramolecular complexes".

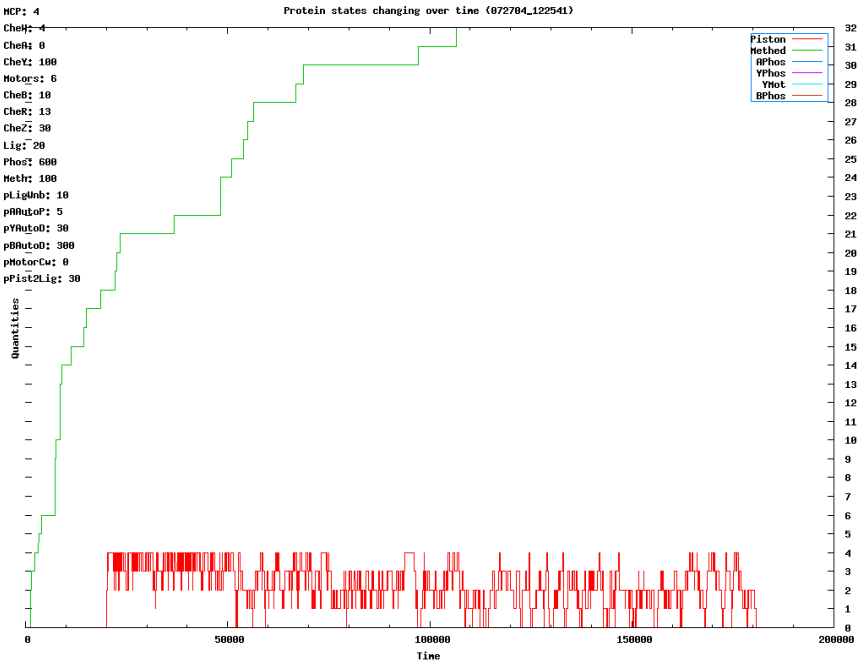


Figure 23 – Knock-out experiment with quantity of CheA set to 0.

In the simulation, this happens because ProbPistonBindToLigandPRI (pPist2Lig on graph) has a value of 30. If this has a value of zero, simulating a simultaneous mutation in the gene for the MCP protein as well as a knock-out of CheA, then the rate of piston binding is consistently high and no longer depends on the amount of methylation. If pPist2Lig has its maximum value of 256, then the rate of piston binding is at a much lower level throughout of the simulation run, except at the beginning when the methylation level is still fairly low (Figure 24).

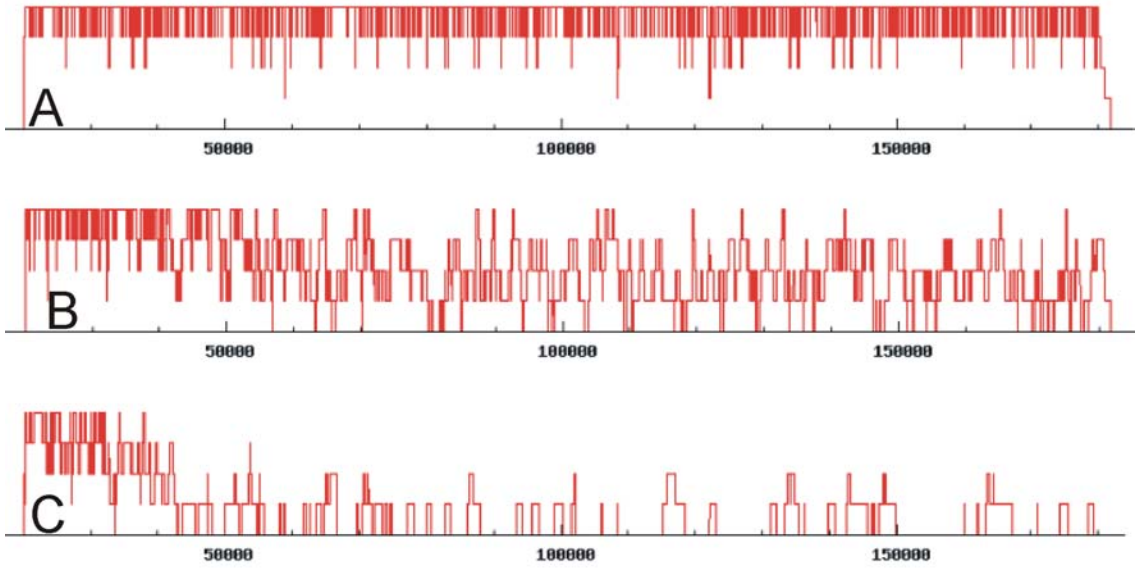


Figure 24 - Extent to which the MCP piston binds to ligand for three values of pPist2Lig. Time runs from left to right (ts=20,000 to 180,000 in graphs). In all three cases the methylation level rises during this period of time. **(A)** pPist2Lig = 0. **(B)** pPist2Lig = 30. **(C)** pPist2Lig = 256.

CheY knock-out

If CheY is knocked out, then no CheY~P can be produced and no biasing of the motor can take place. The motor will turn CCW 100% of the time, and the bacterium will always be in run mode. "Deletion of the CheY response regulator causes *E. coli* to run exclusively" (Rao *et al.*, 2004, p.239). There will also be no competition with CheB for CheA binding domains, and possibly this will tip the scales slightly in the favour of CheB as it reaches a methylation level balance with CheR.

When the simulation is executed, the motor turns CCW 100% of the time, and there is no freely moving CheY~P, or CheY~P bound to motors. As expected, the MCP pistons continue to bind to ligands, and CheA phosphorylates CheB. There is a moderate amount of CheB~P but it is hard to tell by inspection if there is more CheB~P than there would be if it were competing with CheY.

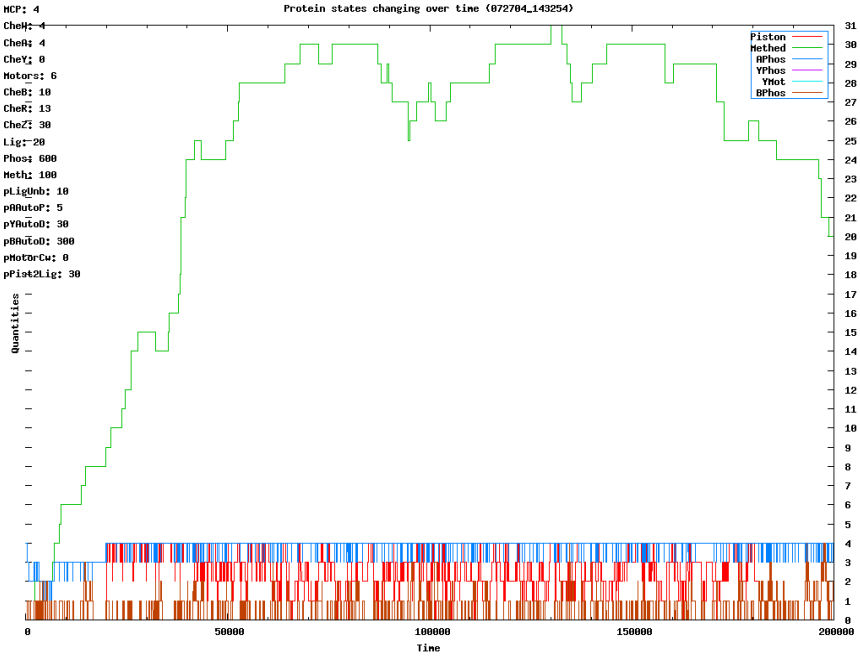


Figure 25 – Knock-out experiment with quantity of CheY set to 0.

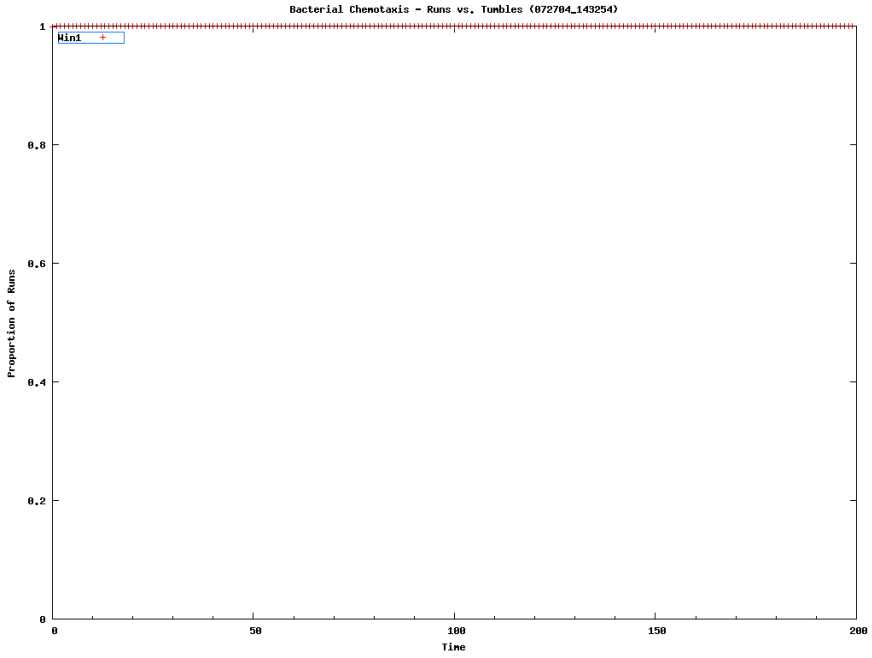


Figure 26 – Knock-out experiment with quantity of CheY set to 0. Note the red crosses at the top. The bacterium was in run mode 100% (1.0) of the time during this simulation.

Motor complex knock-out

If there are no motors then the bacterium will not be able to move, assuming that the motors are attached to flagella. The bacterium will move only subject to the movement of the liquid it is contained within.

In the simulation, if the motors are knocked-out, then the bacterium should neither run nor tumble, but just stay in one place. Everything else should work normally. Possibly there will be a somewhat lower level of CheY~P because CheZ can only dephosphorylate CheY~P that is not bound to a motor.

In Figure 27, the values for the proportion of runs are all zero, indicating a lack of any movement. As expected, everything else looks to be within the normal ranges.

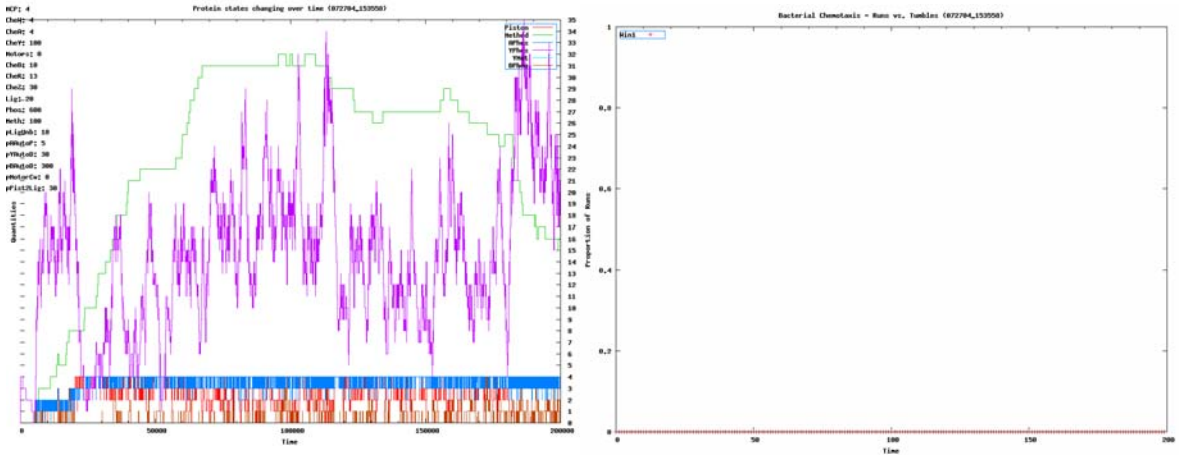


Figure 27 – Knock-out experiment with quantity of motor complexes set to 0.

Motor calibration

The fact that, in laboratory experiments under two special conditions bacteria exhibit well-documented constant behaviours, was used to calibrate my motor rule. In the absence of CheY and thus of CheY~P, the motor turns CCW 100% of the time, and the bacterium therefore only runs in a straight line with no tumbles. In the presence of CheY/CheY~P but in the absence of ligand, the motor turns CCW 92% (0.92) of the time, and if there are six motors will therefore run about 60% ($0.92^6 = 0.606$) of the time. These results were replicated in the simulation knock-out experiments as documented above.

The simulation makes use of a bias calibration term that specifies the additive extent to which each individual CheY~P perturbs the rotation behaviour of the motor to which it binds. This was given a value such that, averaged over many runs, the motor will in fact turn CCW 92% of the time, but only if the starting number of CheY is 100, and assuming that everything else is working with its defaults and otherwise as expected. The use of a bias calibration term was necessary because the simulation does not include any detail about the different types of proteins in the motor complex, but uses a simple aggregate rule to transform number of CheY~P into motor behaviour. See appendix F for details on the calibration process.

Shimizu *et al.* (2003) use an equation to calculate actual motor bias, which they define as "the proportion of time that it [the flagellar motor] spends rotating in the counter-clockwise direction." The equation calculates motor bias from the current CheY~P concentration, and makes use of a constant called a Hill coefficient. The equation also needs to know the steady-state concentration of CheY~P when there is no ligand present. It is difficult to directly compare my approach with that of Shimizu *et al*. because theirs is a differential-equation approach while mine is individual-based. Shimizu uses a special power term, while I use a special additive term.

CheB knock-out

If CheB is knocked out, there can be no production of CheB~P, and no demethylation of methylated MCP proteins. The balance between CheR and CheB~P will be lost, and methylation will increase to its maximum level. The bacterium will be unable to adapt.

In the simulation, the methylation level rapidly increases to its maximum value of 32. It was almost 31 at around ts = 42500, and reached the maximum by ts = 80,000.

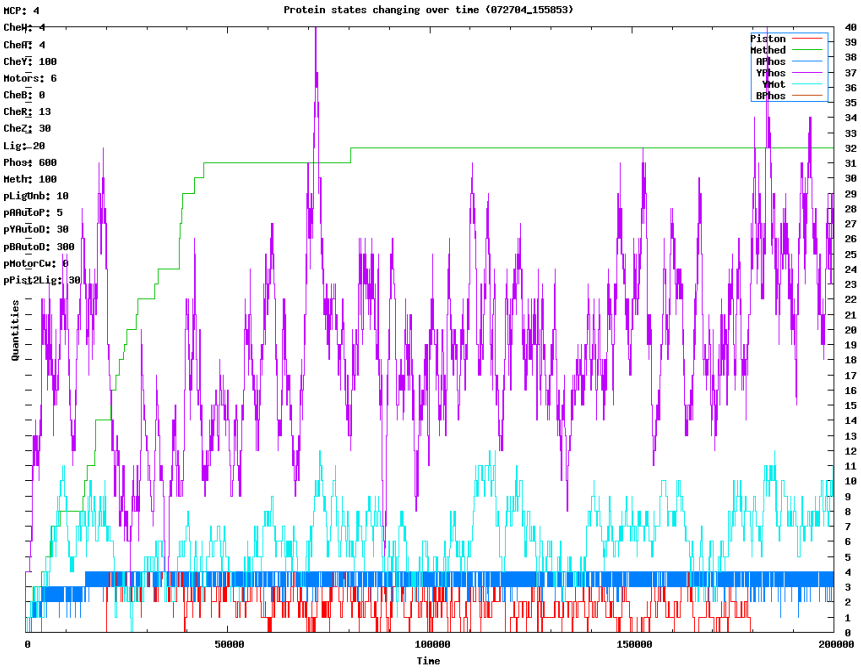


Figure 28 – Knock-out experiment with quantity of CheB set to 0.

CheR knock-out

Rao *et al.* (2004, p.239) state that when "the CheR methyltransferase is deleted in *E. coli*, the cells are incapable of tumbles and only run".

In the simulation, the bacterium runs 100% of the time, except for the first 10,000 timesteps before it reaches a steady-state. The piston binds with ligand, but all other quantities remain at zero.

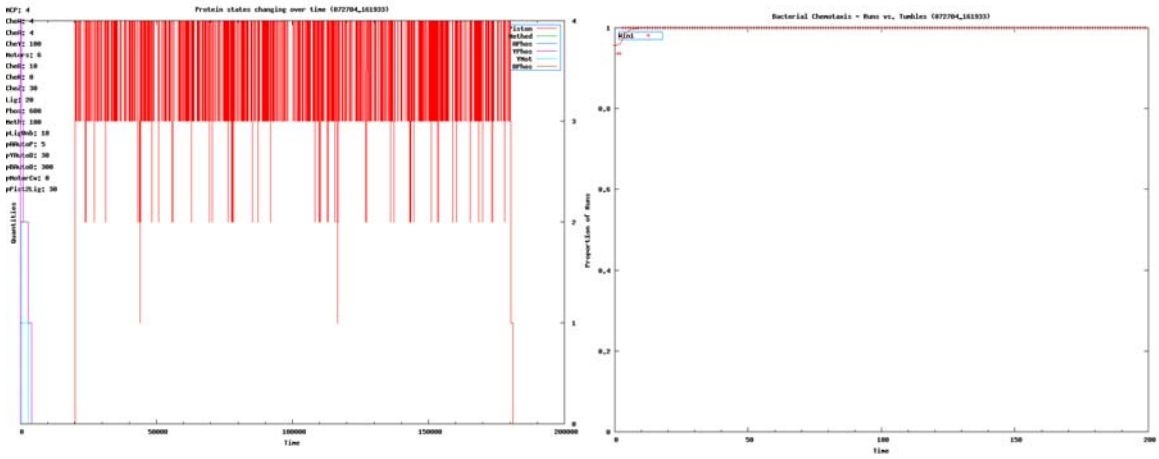


Figure 29 – Knock-out experiment with quantity of CheR set to 0.

CheZ knock-out

CheZ is an enzyme that dephosphorylates CheY~P. If CheZ is knocked out, then CheY~P will only decrease through auto-dephosphorylation.

In the simulation, this should produce some small increase in the quantity of CheY~P. In Figure 30 the proportion of runs appears smaller and more constant than in other runs that had the default value for CheZ. To confirm this tentative observation, additional special trials would need to be run.

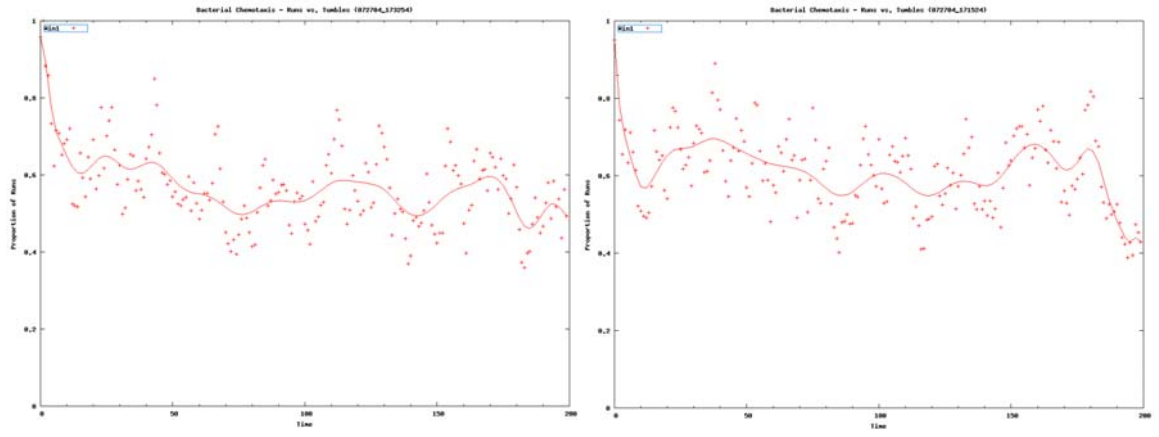


Figure 30 – Knock-out experiment with quantity of CheZ set to 0.

Whole bacterium experiments

A series of experiments was performed on the simulated bacterium, with all proteins and internal small molecules present. The actual quantities and probabilities used, which are similar across experiments, will be provided in the text or figures. The experiments build on each other from reaching internal stability, to responding to ligand in the environment, to adapting to ligand, to attempts at movement along a ligand gradient (chemotaxis).

Bacterium reaches a steady state in the absence of ligand

The simulated bacterium was run many times for 200,000 time steps in the absence of any ligand small molecules in the environment. Across trials, the bacterium consistently increased its levels of phosphorylated CheA, CheY and CheB (CheA~P, CheY~P, CheB~P), and the level of methylation on the Tar MCP. The quantity of CheY~P bound to motors (YMot) also increased. In all trials the quantities increased from 0 at the start, and reached a stable (but noisy) steady state. The specific steady state values depend upon the parameters used. A typical trial with typical parameters is shown in Figure 31.

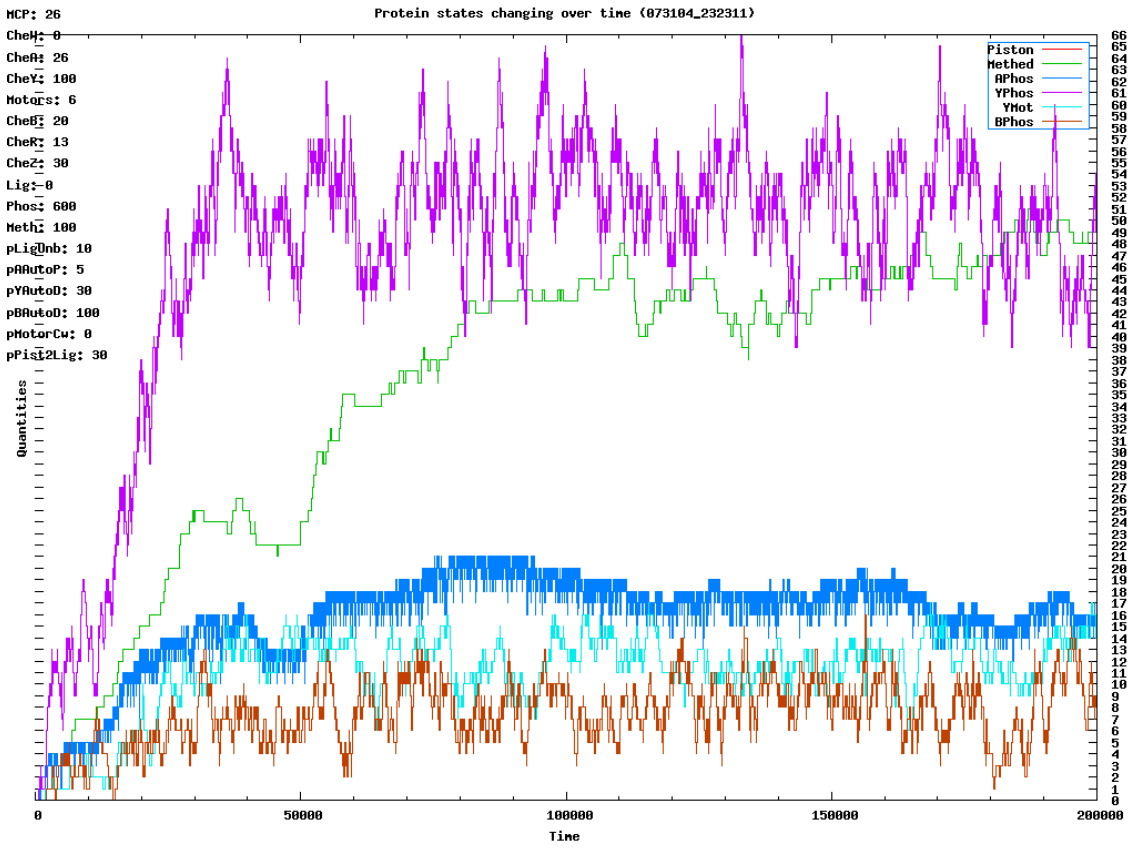


Figure 31 – Reaching a steady state in the absence of ligand (Lig: 0). At time step 0, the levels of all tracked quantities are 0. In this trial there were 26 MCP Tar proteins. The levels of CheA~P (APhos in the figure legend) and CheB~P (BPhos), the total amount of CheY~P (YPhos) and the amount of CheY~P bound to motors (YMot), stabilized by about 40,000 time steps. The methylation level of the MCPs continued to increase until around 80,000 time steps. All levels oscillate considerably until the

end of the simulation run at time step 200,000 due to inherent noise, but the mean levels tend to remain relatively constant.

The binding of the CheY~P to the motors caused the motors to sometimes rotate clockwise. This resulted in a decreased motor bias, and decreased proportion of times that the bacterium is in run rather than tumble mode. The mean proportion of runs decreased from 1.0 (100% of the time) to a steady state which should be around 0.6 if the motors' *bias calibration term* is properly set for the particular quantity of CheY used in the experiment. The decrease in the proportion of runs over time to a steady state is shown in Figure 32. For this and several other experiments the motors were not recalibrated to take account of the changed parameters, and the proportion of runs is around 0.5 rather than 0.6.

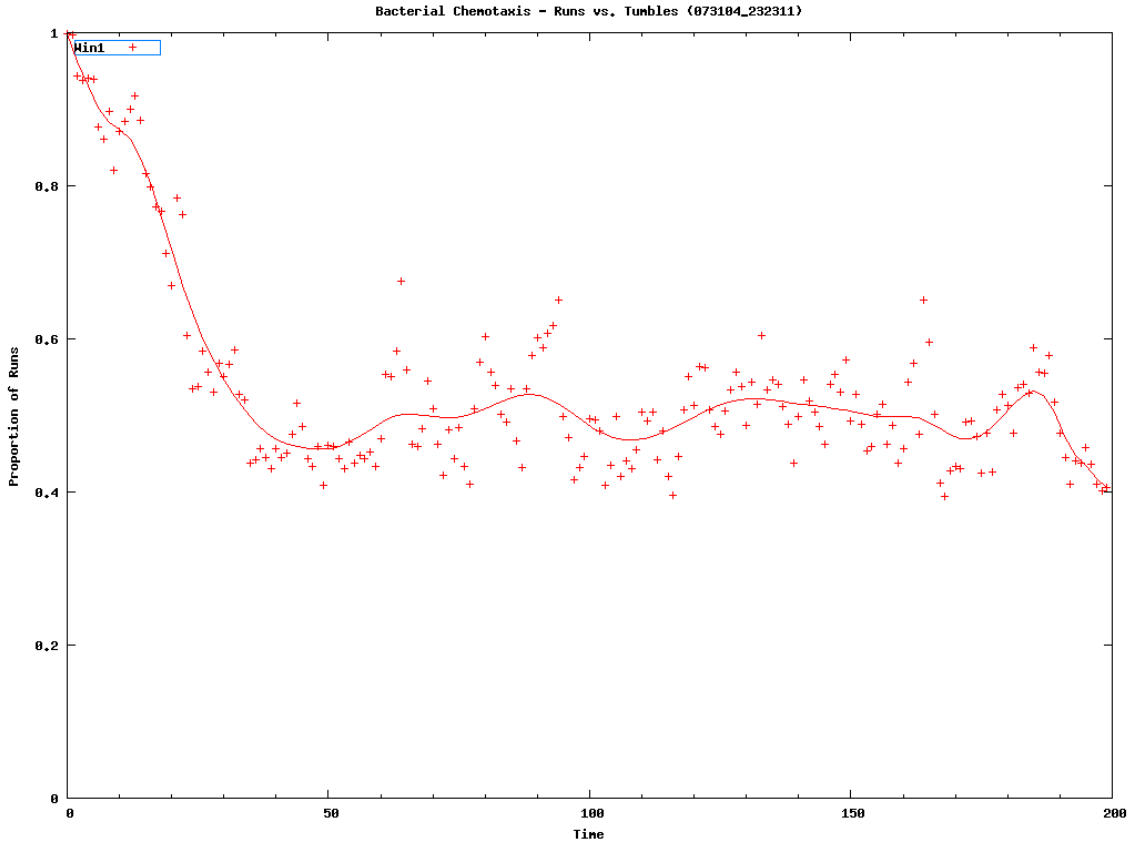


Figure 32 – Reaching a steady state in the absence of ligand (Lig: 0). The proportion of runs decreases rapidly from 1.0 (100%) and stabilizes by time step 50,000 (50 on the figure) at around 0.5, again with oscillations due to noise.

Note that in this as well as in other *Runs vs. Tumbles* figures the very rightmost part of the curve that's been fit to the data points is often misleading. This is an artifact of the gnuplot curve-fitting algorithm that always draws a line segment directly to the final point in the graph. Thus, in Figure 32, the mean proportion incorrectly appears to significantly decrease just before Time=200.

The results obtained in the steady-state experimental trials agree with expectation. In all subsequent experimental trials reported on in this paper, the bacterium was run for at least 50,000 time steps in the absence of ligand to allow it to initially reach an internal steady state.

Bacterium responds to introduction of ligand

In another set of experimental trials, a measured quantity of ligand was released into the simulated environment once the bacterium had reached a steady state. The ligand was removed after a further 10,000 time steps, in real time on the order of one second or so. This type of *impulse* experiment is commonly reported in the literature, both for real bacteria and for computer models.

When all proteins and probabilities have their default values, the simulation should exhibit a chemotaxis response. Segall *et al.* (1986, p.8988) report a characteristic response observed experimentally when wild-type *E. coli* are subjected to a brief (0.02 second) pulse of aspartate attractant. As shown in Figure 33, at steady state the motor bias is about 0.6. An attractant pulse pushes this rapidly and sharply higher, followed by a slower decline back to the original level. There is typically an undershoot before it returns to the original level.

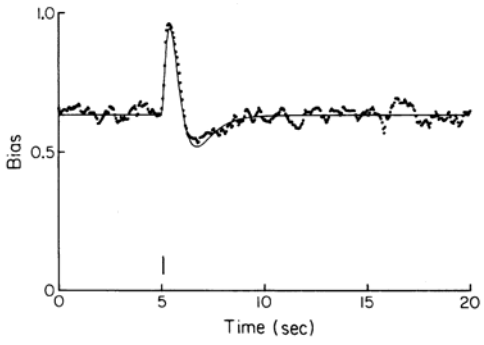


Figure 33 - "Impulse response to attractant in wild-type cells. The dotted curve is the probability, determined from repetitive stimulation, that tethered cells of strain AW405 spin CCW when exposed to pulses of L-aspartate or a-methyl-DL-aspartate beginning at 5.06 sec (vertical bar). The smooth curve is a fit to a sum of exponentials." (Segall *et al.*, 1986, Fig. 1).

Shimizu *et al.* (2003, p.296) describe how their chemotaxis computer model replicates this, as shown in Figure 34.

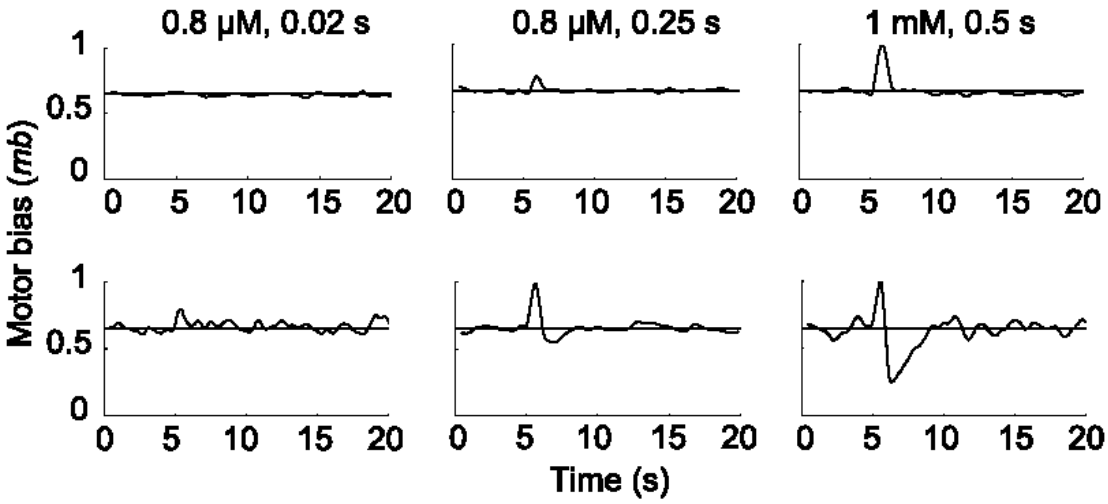


Figure 34 - "simulated impulse responses. All panels show the response in motor bias to a brief pulse of aspartate at five seconds." (Shimizu *et al.*, 2003, Fig. 3).

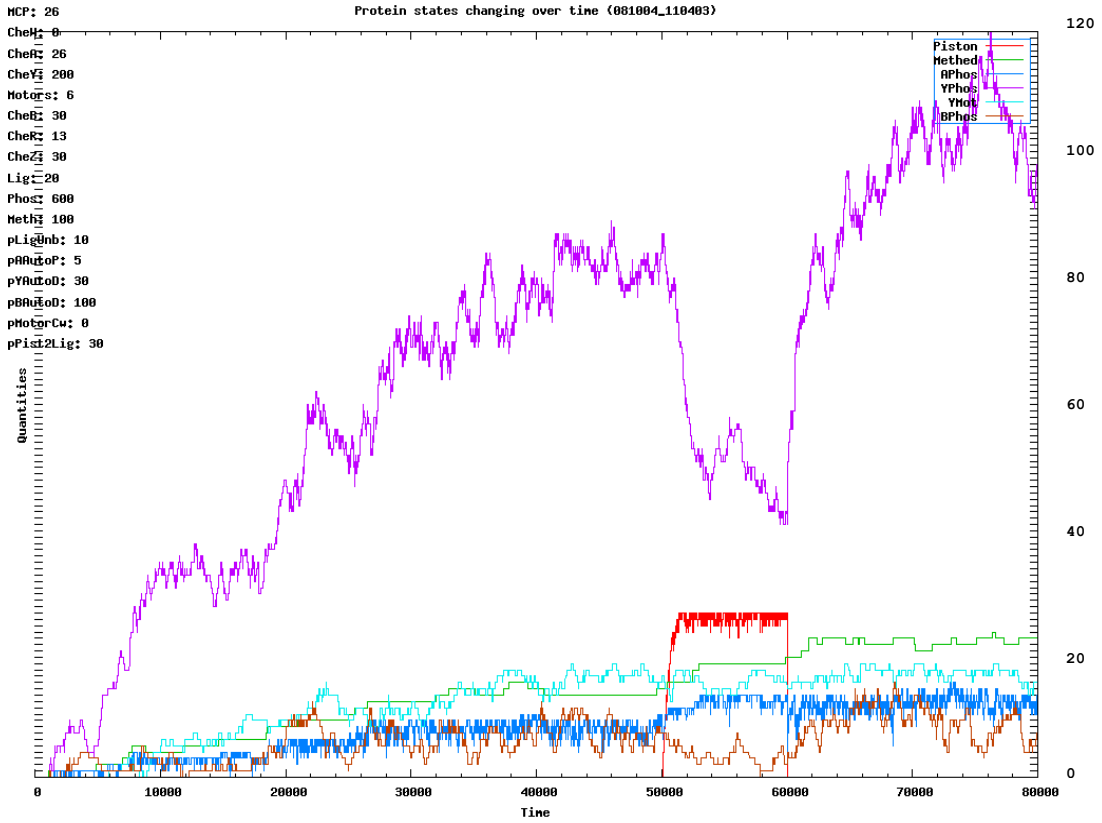


Figure 35 – Results from typical impulse experimental trial. In these trials there were 26 MCP proteins. The simulation was allowed to run for 50,000 time steps to reach a steady state. 400 units of ligand (Lig: 20) were introduced at time step 50,000 and removed at time step 60,000. There was a sharp rise from 0 in the number of MCPs bound to ligand and the consequent number of extended pistons (Piston: red). This resulted in sharp decreases in the number of phosphorylated CheY (YPhos: magenta) and CheB (BPhos: brown).

Figure 35 shows a typical response, in the kPBNN simulation, to the introduction of 400 ligand molecules, released simultaneously at both the top centre and bottom centre of the local environment adjacent to the Tar MCP receptor proteins of the bacterium. The response is strong, quite rapid, and highly reproducible across trials. These trials all ran for 80,000 time steps, an initial 50,000 to reach a steady state, 10,000 for the impulse, and a further 20,000 to return to a steady state. The trials all use the same default parameter values as documented on Figure 35.

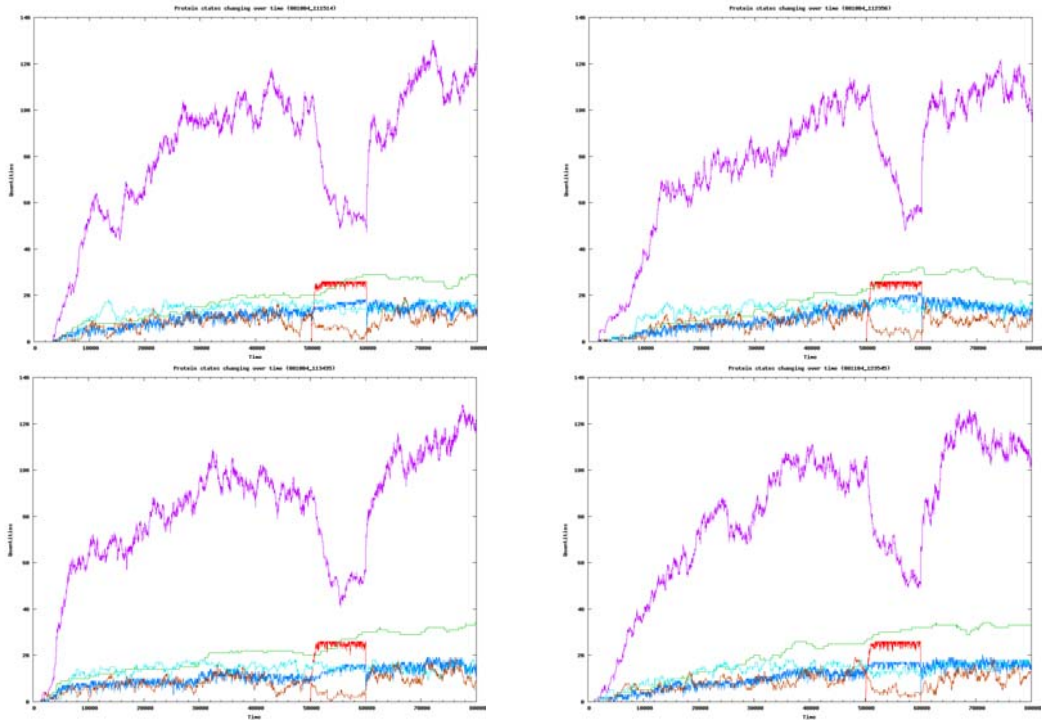


Figure 36 – Results from additional impulse experimental trials, all using the same parameters as in **Figure 35**. These are included to substantiate the claim that the results shown in **Figure 35** are typical.

In many of the trials the steady state reached after the removal of ligands appears to be at a higher level than it was immediately before introduction of the ligand. Whether this is a statistical anomaly, is related to the undershoot seen in Figure 33, or has some other origin, has not been investigated in this dissertation.

Figure 36 shows the results for a further four trials.

Figure 37 shows that these trials produced little if any discernible difference in the proportion of runs.

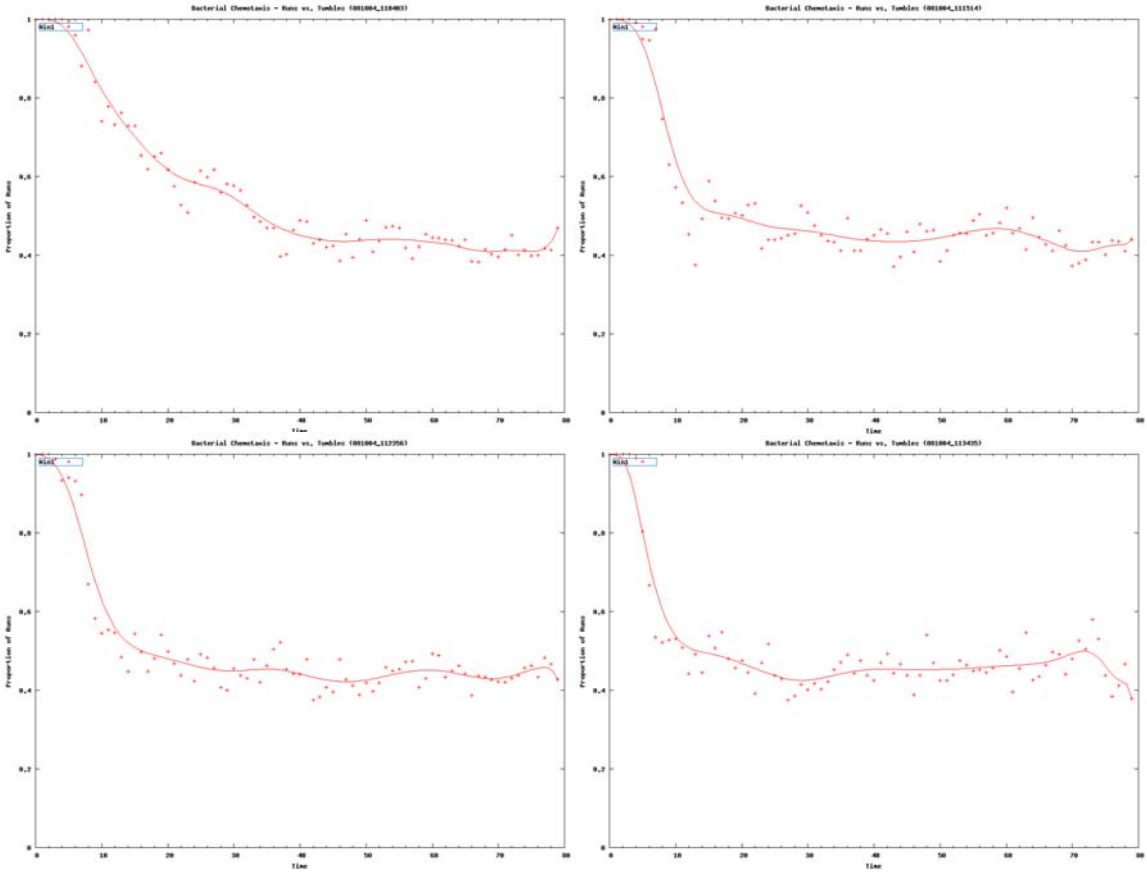


Figure 37 – Impulse experimental trials, with default values, showing the proportion of runs. Note the non-existent or very weak response between 50,000 and 60,000 time steps. The large change in the quantity of CheY~P (YPhos in **Figure 35**) has not translated into a change in the overall behaviour of the bacterium.

Figure 38 and Figure 39 show the result of a set of control trials. All parameters are the same as for the impulse trials, except that no ligand is introduced into the environment. Note the lack of change between time steps 50,000 and 60,000.

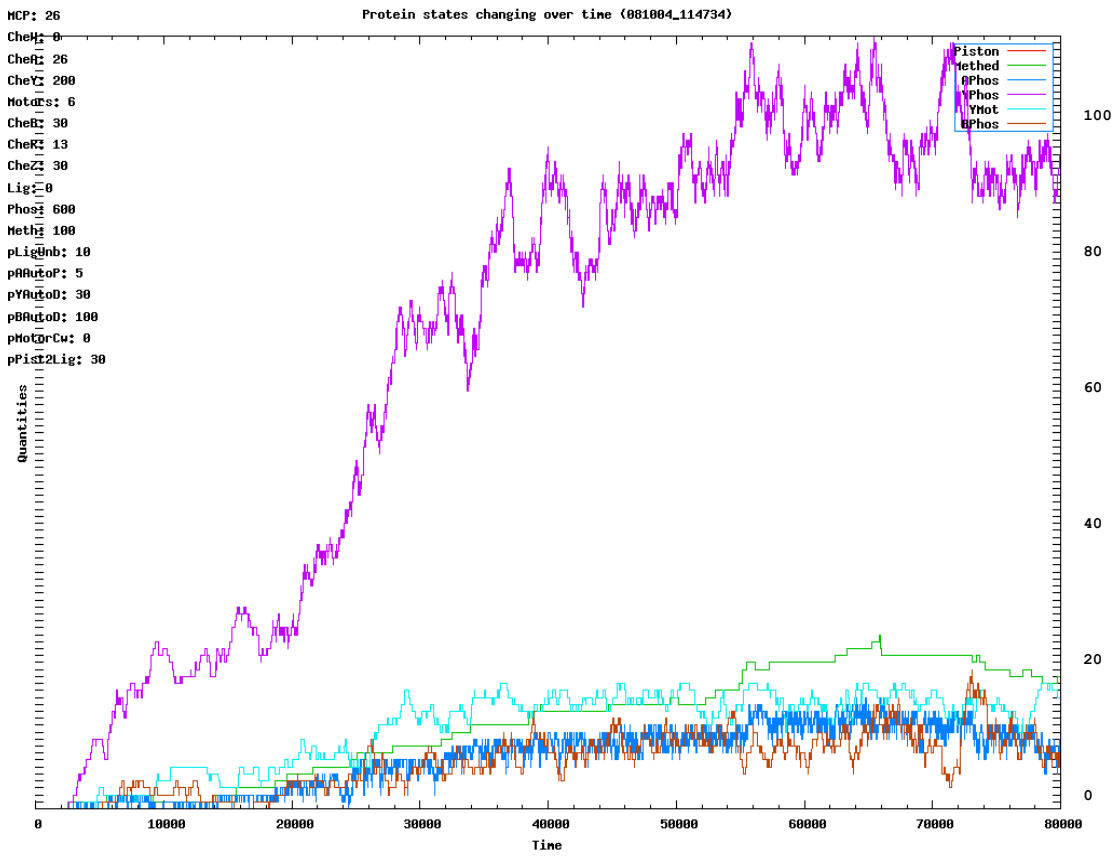


Figure 38 – Typical control trial. The level of ligand is kept at 0 for the entire 80,000 time steps. There is no change between time steps 50,000 and 60,000 in the quantities of extended MCP (Piston), phosphorylated CheY (YPhos), and other quantities as there is in **Figure 35**.

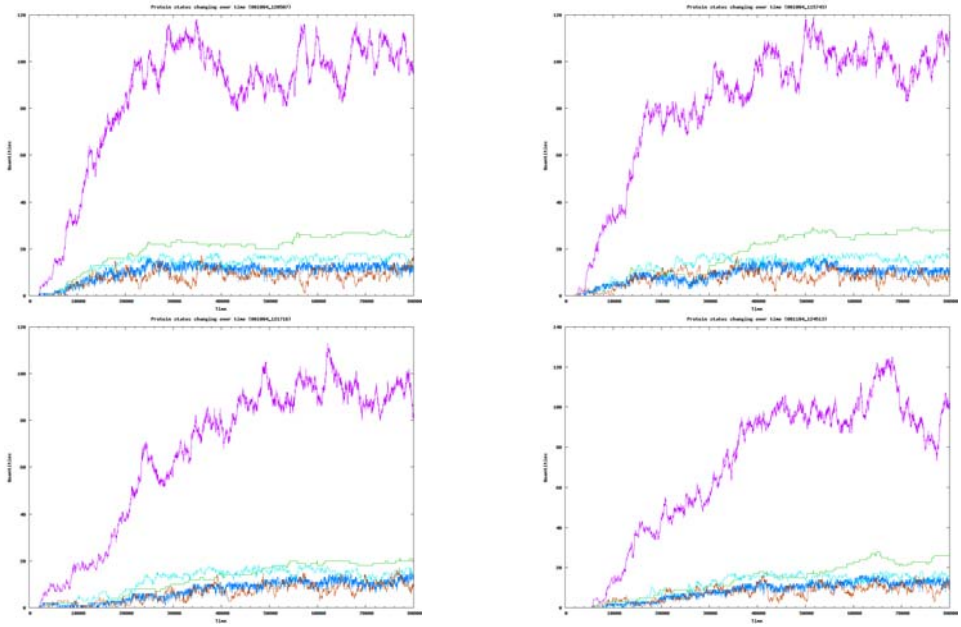


Figure 39 – Additional control trials. These are included to substantiate the claim that the results shown in **Figure 38** are typical.

Although the introduction and subsequent removal of ligand in the simulation produces a strong effect on the quantity of CheY~P, it only very weakly changes the amount of CheY~P bound to motors. This accounts for the lack of result in Figure 37.

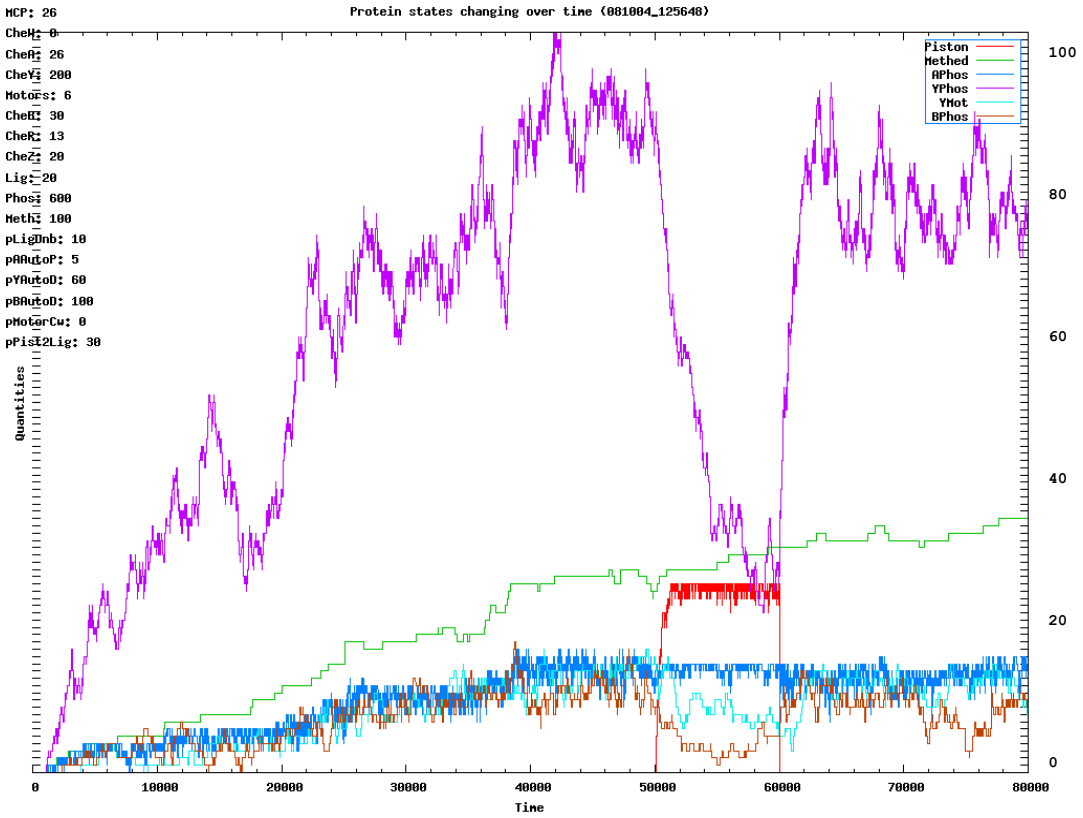


Figure 40 – A new set of trials with CheZ = 20 and pYAutoD = 60. Compare this with **Figure 35**. Note especially the substantial decrease in the amount of CheY~P bound to motors (YMot: cyan) after the introduction of ligand at time step 50,000, and its increase back to a steady-state level after time step 60,000 when the ligand is removed.

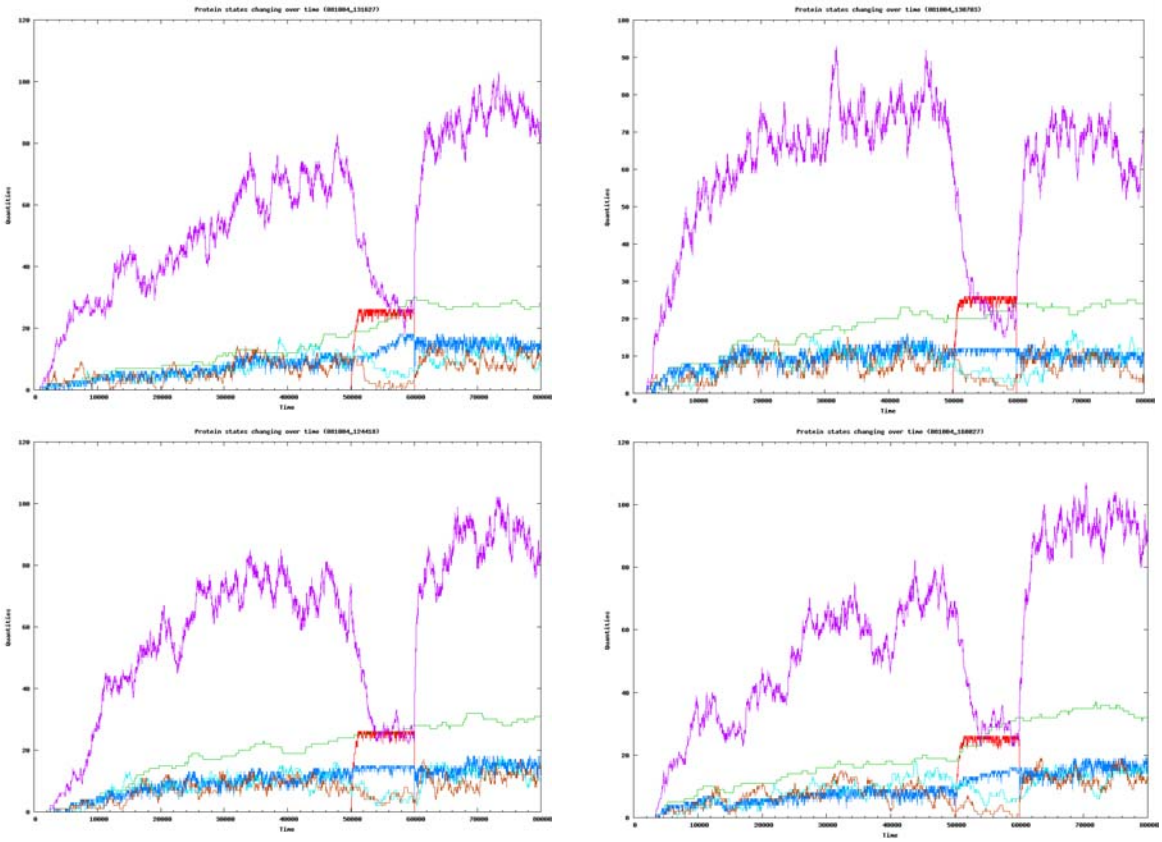


Figure 41 – Additional trials with CheZ = 20 and pYAutoD = 60. These are included to substantiate the claim that the results shown in **Figure 40** are typical.

A further series of trials was run in which two of the initial parameters were changed. The enzyme protein CheZ moves freely through the bacterium cytoplasm, and dephosphorylates CheY~P that is also free and not currently bound to a motor. The parameter pYAutoD determines the probability at each time step that an individual CheY~P will auto-dephosphorylate back into CheY, whether it is free in the cytoplasm or bound to a motor.

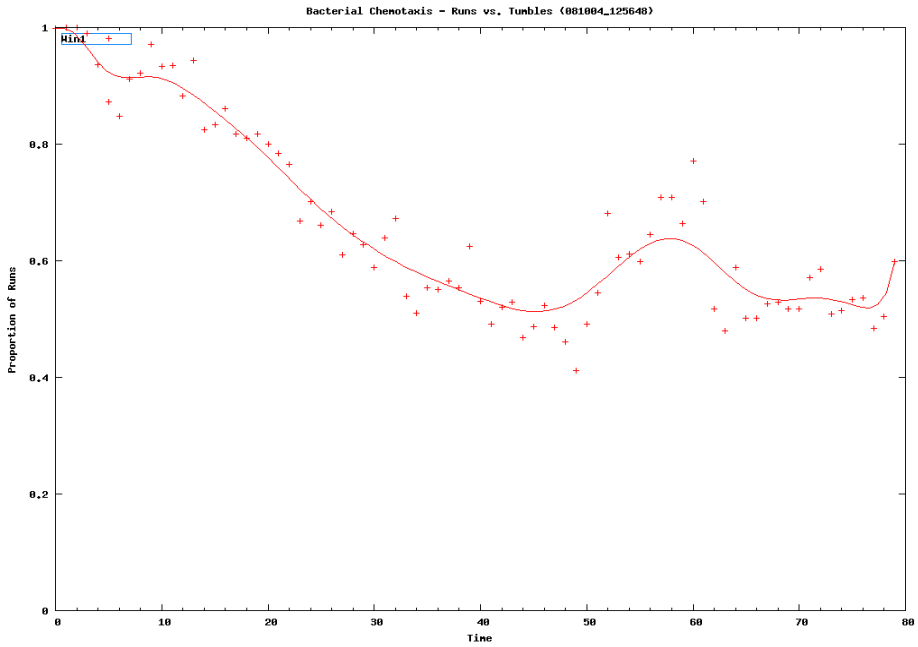


Figure 42 – Proportion of runs when CheZ = 20 and pYAutoD = 60. Compare this with **Figure 37**. There is a substantial increase in the proportion of runs to tumbles between time steps 50,000 and 60,000 (shown in the figure as 50 and 60).

For this new series of trials, the amount of CheZ was decreased from 30 to 20 molecules, and the value of pYAutoD was increased from 30 to 60 (out of a maximum value of 10,000). These changes increased the rate at which all CheY~P proteins would dephosphorylate, but even more so for free CheY~P than for bound CheY~P. The result was that CheY~P remained bound for less time to motors, thus increasing the time responsiveness of the motors to changes in the amount of ligand in the environment. The overall amount of CheY~P remained high enough that it could continue to respond sharply to changes in ligand amount. Figure 40, Figure 41, Figure 42, and Figure 43 show the results. Note especially the much more pronounced decline in the quantity of YMot (CheY~P bound to a motor), and the resulting increase in proportion of runs between time steps 50,000 and 60,000, as compared with the earlier figures.

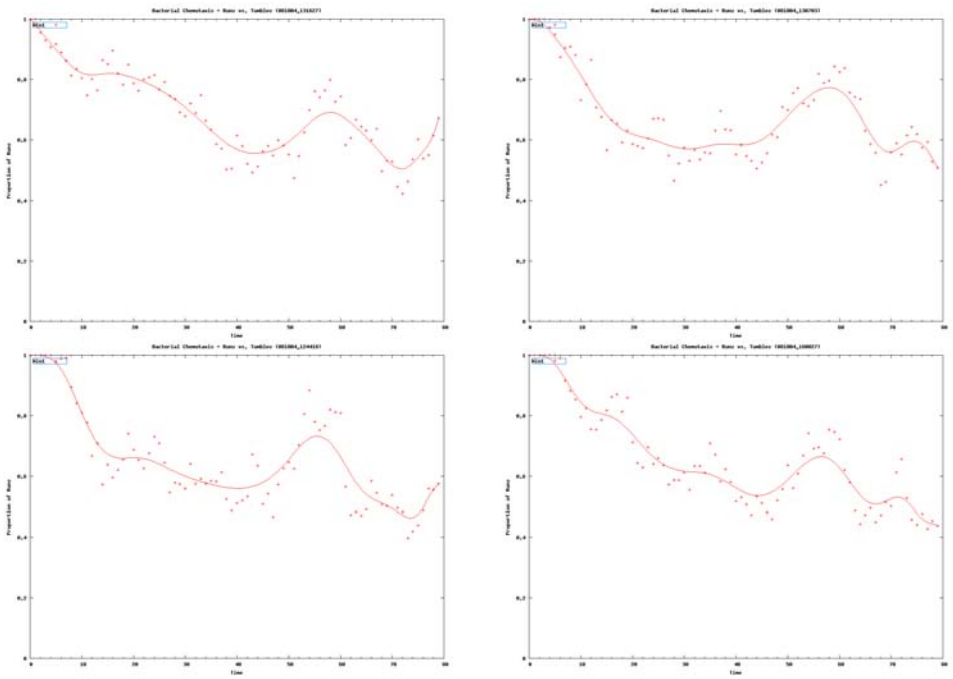


Figure 43 – Proportion of runs when CheZ = 20 and pYAutoD = 60. This figure is included to substantiate the claim that the results shown in **Figure 42** are typical.

Figure 44 compares the change in YMot between using the default parameters and using the new values for CheZ and pYAutoD.

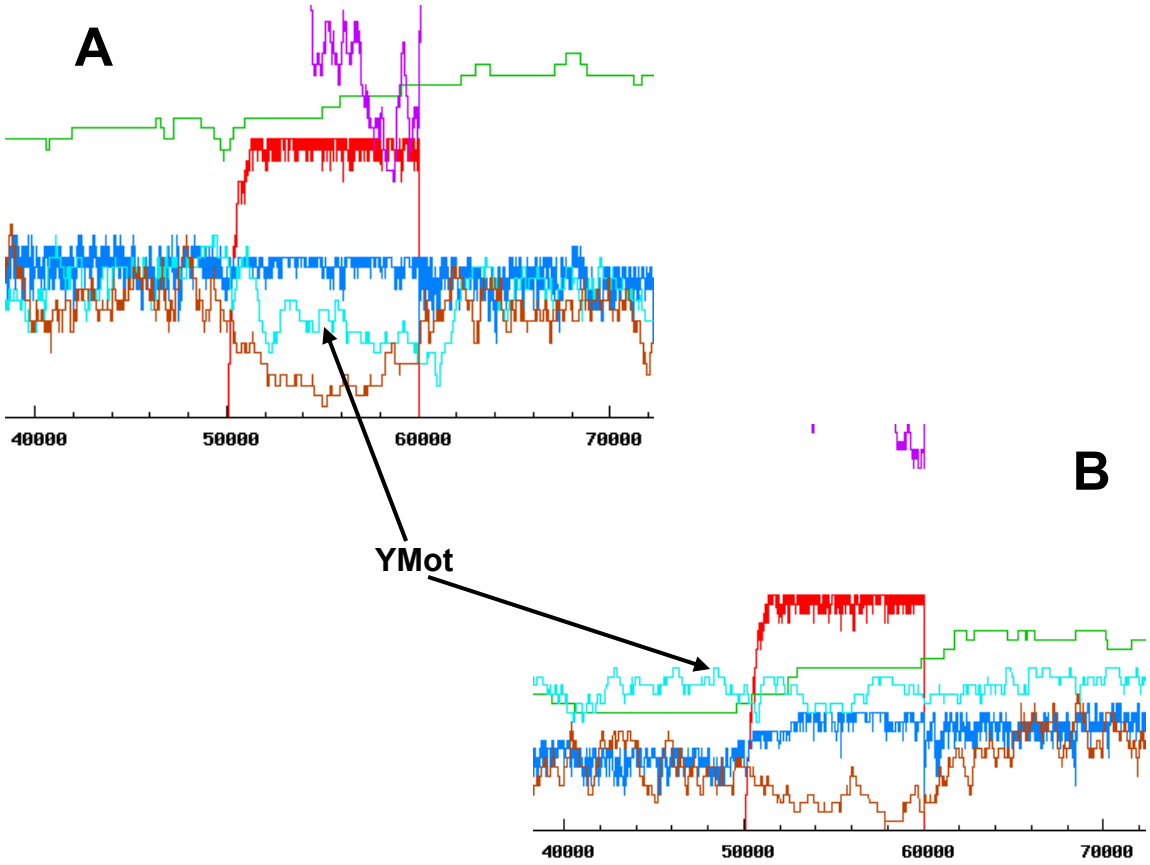


Figure 44 – Close-up comparison of the amount of phosphorylated CheY bound to motors (YMot) when: **(A)** CheZ = 20 and pYAutoD = 60, and **(B)** CheZ = 30 and pYAutoD = 30. In both graphs YMot is the light blue (cyan) line. Experimental trials consistently showed this type of substantial decrease in YMot which translated into the type of change in bacterial behaviour shown in **Figure 42** and **Figure 43**.

Bacterium adapts to sustained level of ligand

A simulated bacterium was run for 450,000 time steps with the default parameters, but using 200 (10 * 20) rather than 400 ligand molecules. In each trial, after reaching a steady state, the 200 ligand molecules were introduced at time step 50,000. Half of these were placed in the top centre of the simulated local environment, and the other 100 in the bottom centre, close to the 26 Tar MCP receptor proteins. All ligand was removed from the environment at time step 180,000.

Rao *et al*. (2004) describe their results from running a deterministic differential-equation based model of bacterial (*E. coli*) adaptation, which uses quantities and kinetic values that are close to those determined in the lab for real bacteria. They introduced an attractant ligand at 500 seconds and removed it at 1000 seconds. Figure 45 shows how the quantities of CheY~P and the level of methylation varied during the course of their model run.

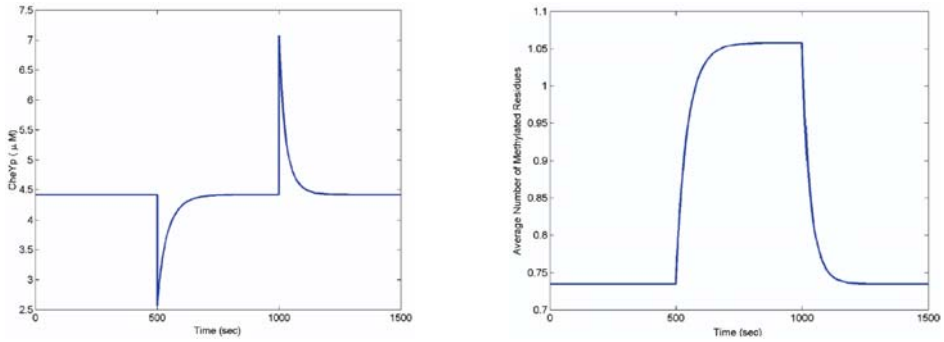


Figure 45 – "Simulation of Adaptation in *E. coli* … Attractant (10 μM) is added at 500 s and removed at 1,000 s. Timecourse simulation of phosphorylated CheY (left) and receptor methylation (right) in *E. coli*." (Rao *et al*., 2004, Fig. 4A)

Figure 46 shows the results from one trial of the kPBNN simulated bacterium. Superimposed on the experiment results, I have drawn lines that visually and approximately correspond to the lines from Figure 45. My claim is that there is a pattern that comes through the noise in my results, and that that pattern is qualitatively similar to the deterministic results reported by Rao *et al*.

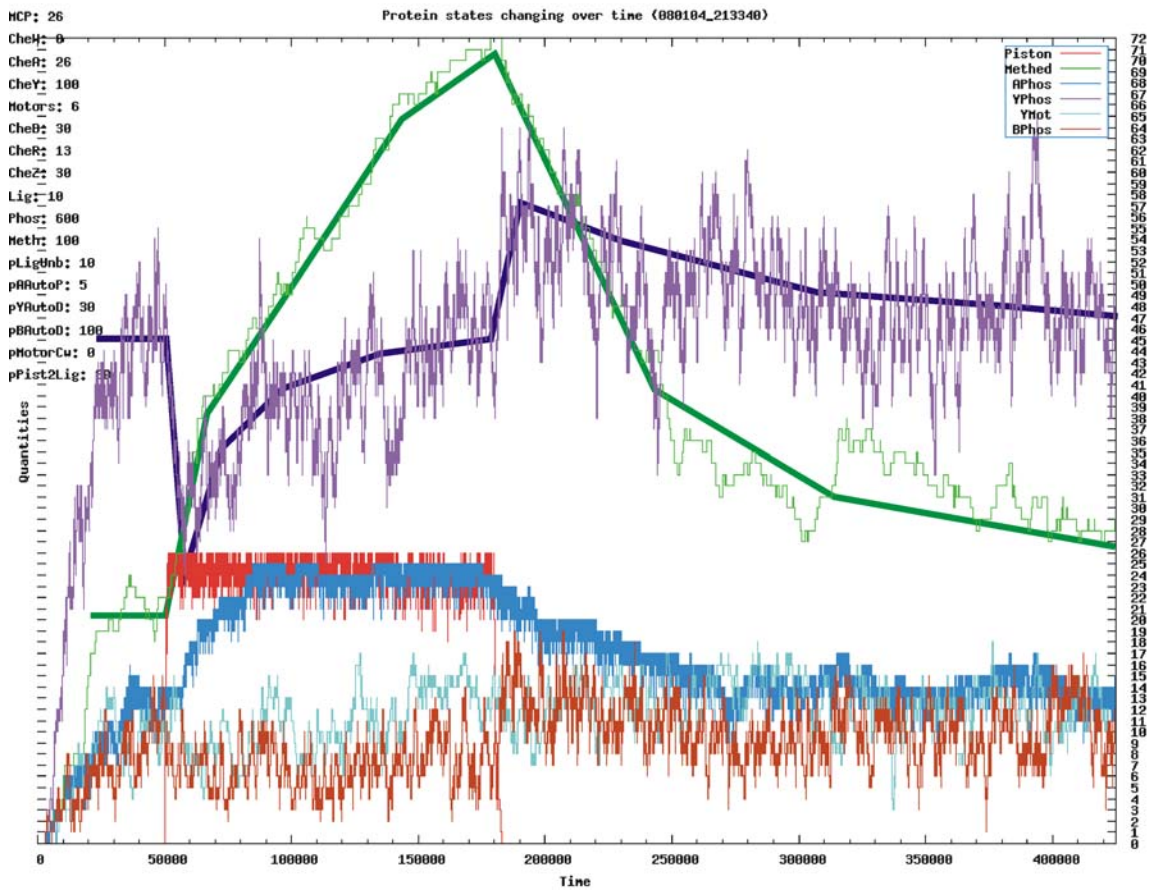


Figure 46 – My results with those of Rao *et al*. very roughly superimposed, in support of the claim that my simulation produces behaviour that is qualitatively similar to that produced by the Rao *et al*. (2004) deterministic model.

The darker (purplish) line shows the quantity of CheY~P. It rapidly increases and then reaches a steady state. With the introduction of ligand at time step 50,000, it decreases sharply, and then begins to more gradually and asymptotically increase, approaching but perhaps not quite reaching its steady state level by time step 180,000. When the ligand is removed, the amount of CheY~P rapidly increases, and then falls very gradually and asymptotically towards its steady state value.

The lighter (greenish) line shows the level of methylation for the Tar MCP protein receptors. It rapidly increases and then reaches a steady state. When ligand is introduced, it increases quite rapidly and then less so, seemingly tending toward an asymptote. Note that initially the methylation level moves in the opposite direction to the quantity of CheY~P, and then the two both increase together. When the ligand is removed, the methylation level decreases rapidly and then gradually settles toward a steady state level.

Figure 47 shows the results of additional runs using the same parameters.

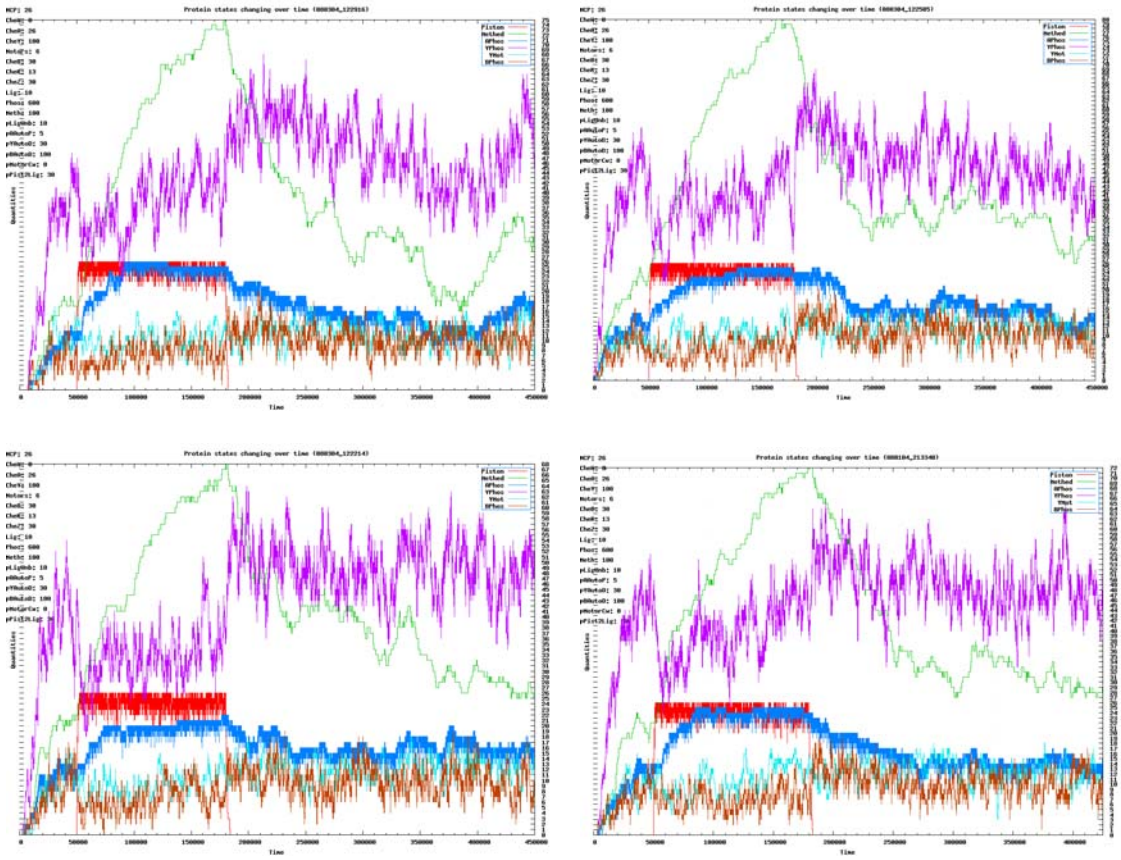


Figure 47 – Additional trials all using the same parameters. The lower right graph is the same as **Figure 46**, reproduced here for ease of comparison. The additional three results are included in support of my claim that the results of **Figure 46** are typical.

It is important to note that the quantities in my simulation are operating somewhere in the middle of their theoretical ranges, with two exceptions. The number of activated (piston extended) Tar MCP is consistently at or close to the maximum value of 26, indicating that the quantity of ligand is close to saturation level. The quantity of CheA~P also approaches its upper limit of 26. The amount of CheY~P bound to motors (YMot) does just reach its theoretical maximum of 18, but is usually well below that value. Table 5 shows the *theoretical maximum* and *actual maximum reached* for each quantity tracked during the simulation runs.

Table 5 - How close simulation quantities are to their theoretical maxima.

	Tar Piston	Tar Methed	CheA~P (APhos)	CheY~P (YPhos)	CheY~P bound to motor (YMot)	CheB~P (BPhos)
Theoretical Maximum	26	208	26	100	18	30
Actual Max. Reached	26	72	25	65	18	19

By operating within the middle portion of its range, a quantity is always free to move either up or down in response to events in the simulation.

In Figure 48, Lauffenburger (2000) distinguishes three types of adaptation in bacterial chemotaxis. Using his criteria, the kPBNN simulation appears to be an example of partial adaptation.

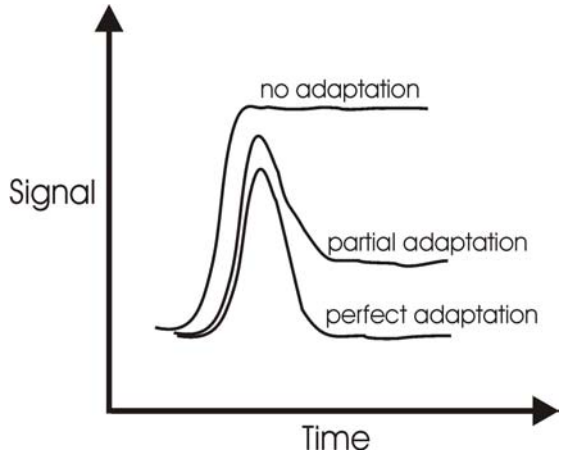


Figure 48 – "Illustration of dynamic responses of a system to a change in input. In the total absence of adaptation, a new steady-state output is attained; partial adaptation reduces the 'offset' of the new steady-state output value compared with the original prestimulus value, whereas perfect adaptation brings the output back to the original value. Alon *et al*. [1999] previously found that the bacterial chemotaxis system … adapts perfectly …" (Lauffenburger, 2000; graph redrawn from Fig.2)

Bacterium is unable to follow a ligand gradient

A series of experiments was run to see if the bacterium can adapt fast enough to consistently move toward a region of higher ligand concentration. These experimental trials used a client-server configuration. The client is the same bacterium as in the previous experiments. Instead of having a concentration of ligand in its local environment that can only be changed by pre-specified amounts at pre-specified time steps, the server now manages a much larger environment that may contain a ligand gradient. At each time step the bacterium client tells the server whether its current behaviour is a run or a tumble. The server updates the position of the bacterium in its grid and tells the bacterium the ligand concentration at the new location.

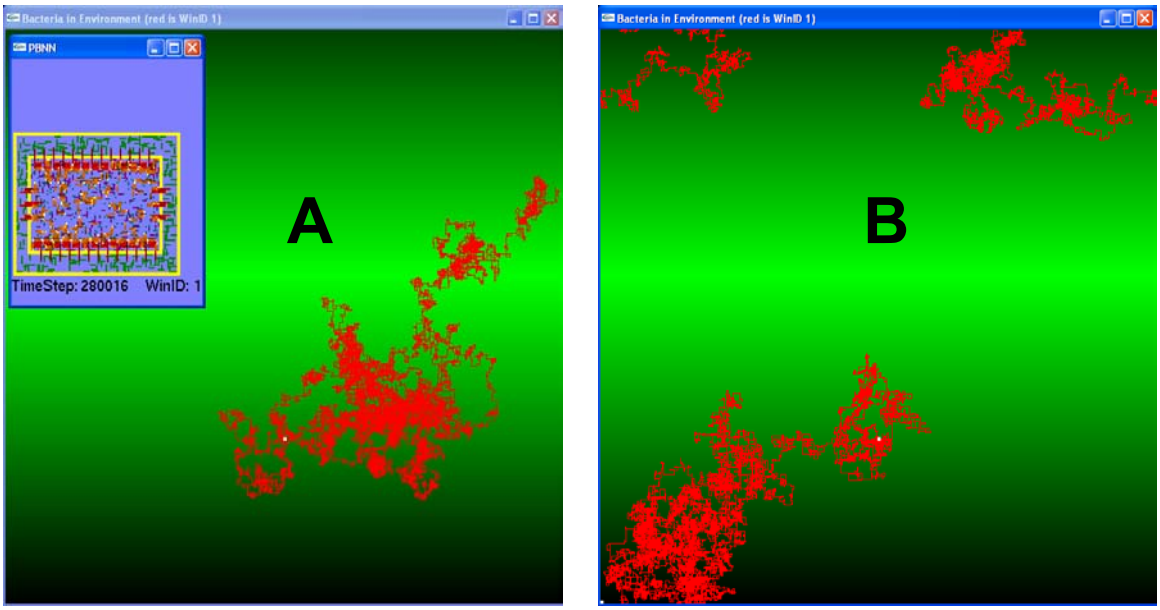


Figure 49 – Movement of bacterium (red) in a 700 by 700 environment grid. Its initial position is shown by the small white square in the lower middle. The ligand concentration (shades of green) increases from 0 at the bottom, peaks at row 400, and then decreases again. **(A)** The bacterium has tended to move toward and stay within a region of higher ligand concentration. The visual appearance of the client bacterium at time step 280,016 are included with the server grid at the same time step. **(B)** The bacterium has tended to move toward a region of minimal ligand concentration.

In the trials, the server manages a 700 by 700 grid configured to have a smooth gradient with zero ligand (NumLigands: 0) at the bottom of the grid, and the highest concentration of ligands in row 400 (NumLigands: 400). From row 400 to 699 the amount of ligand decreases again. The bacterium is initially placed in row 199 exactly half-way up the gradient, facing in a neutral direction along the row. The hypothesis is that it will move toward the higher concentration of ligand rather than toward the zero level.

Several dozen trials were run. On average, the bacterium moves toward the zero concentration about as often as it moves toward a higher concentration. See Figure 49. As expected and consistent with previous experiments, its runs tend to become longer as it begins to move into regions of higher concentration, and shorter when it begins to move in the other direction. This asymmetry should bias it in favour of moving up the gradient. However, it normally overshoots the region of maximum ligand. And if it happens to reach the region of very low ligand concentration it takes a long time to get out because of its increased rate of tumbling and consequent short runs.

Noise

There is considerable noise in the simulation, which is apparent in all of the graphs, especially with the quantity of CheY~P. Despite this, the signal is strong enough for patterns to get through. One cause of the noise, suggested by Morton-Firth and Bray (1998) (Figure 50), is

the relatively small number of instances of each type of polymer used in the simulation. A real bacterium may have thousands of proteins of each type, while the simulation uses between 6 and 100 (MCP: 26, CheW: 0, CheA: 26, CheY: 100, Motors: 6, CheB: 30, CheR: 13, CheZ: 30).

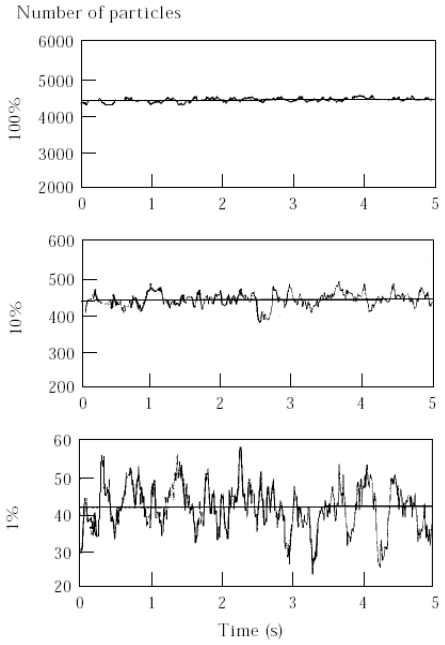


Figure 50 – "Variation of fluctuations with the number of particles. CheYp [CheY~P] fluctuations are shown for simulations containing 22000, 2200 and 220 particles, corresponding to approximately 100, 10 and 1% of the bacterial volume respectively. As mentioned in the text, the amplitude of the fluctuations grows as the square root of the number of particles so that their relative size becomes less as the volume increases." (Morton-Firth and Bray, 1998, Fig. 5)

Another cause is the stochastic nature of many of the calculations performed each time step. Most things occur with some probability rather than strictly deterministically.

Time

The amount of time taken up by a time step is left unspecified for the simulation. It is less than one millisecond.

In a real bacterium it takes about a tenth of a second for a signal to progress from a newly-bound MCP to the motor. Almost all of this time is spent by the CheY~P as it unbinds from the MCP sensory complex, diffuses randomly through the cytoplasm, and eventually binds with the motor complex. On average, in the simulation, this takes about 1000 time steps. This is the amount of time it takes on average for the first CheY~P to become bound to a motor, the YMot quantity on the graphs. Therefore one (preliminary and not very accurate) measure of time is that one second takes roughly 10,000 time steps, and that an entire simulation trial represents about 20 seconds of real time.

An accurate and consistent measure of time would require a more accurate simulation. The relative sizes of polymers are one important factor, to ensure correct diffusion distances.

Competitive binding

In the simulation, a polymer binding domain may only bind to another binding domain if some third polymer is not already bound to that location. Polymers compete for access to binding domains. This introduces a subtle realistic influence in which for example the amount of CheB can have a significant indirect influence on the amount of CheY~P and thus of the whole phosphosignaling branch even though it only has a direct effect on the adaptation branch. The ability to explore the effects of competitive binding is a benefit of the kPBNN simulation system.

At present, CheR and CheB~P do not actually bind to the methylation domain in an MCP. Their respective methylation and demethylation behaviours take place within a single time step with only an implied binding and unbinding. If these two actually bound for some finite amount of time, then there would also be competitive binding of CheR and CheB~P which would introduce additional subtle effects.

The Computer System

The kPBNN system has been written in C++ and compiled using Microsoft Visual C++ (version 6.0). It has been implemented as a console application and does not make use of any Microsoft-specific features. It has intentionally been conservatively written using generic C++, to enable porting to other non-windows platforms. It has been successfully compiled using the MinGW GNU compiler as distributed with the Dev-C++ development environment (version 4.9.8.0) on a windows platform. It should therefore also compile on Unix and Linux platforms although this has not yet been attempted. The 2D graphics output uses OpenGL which is also compatible with a wide range of platforms.

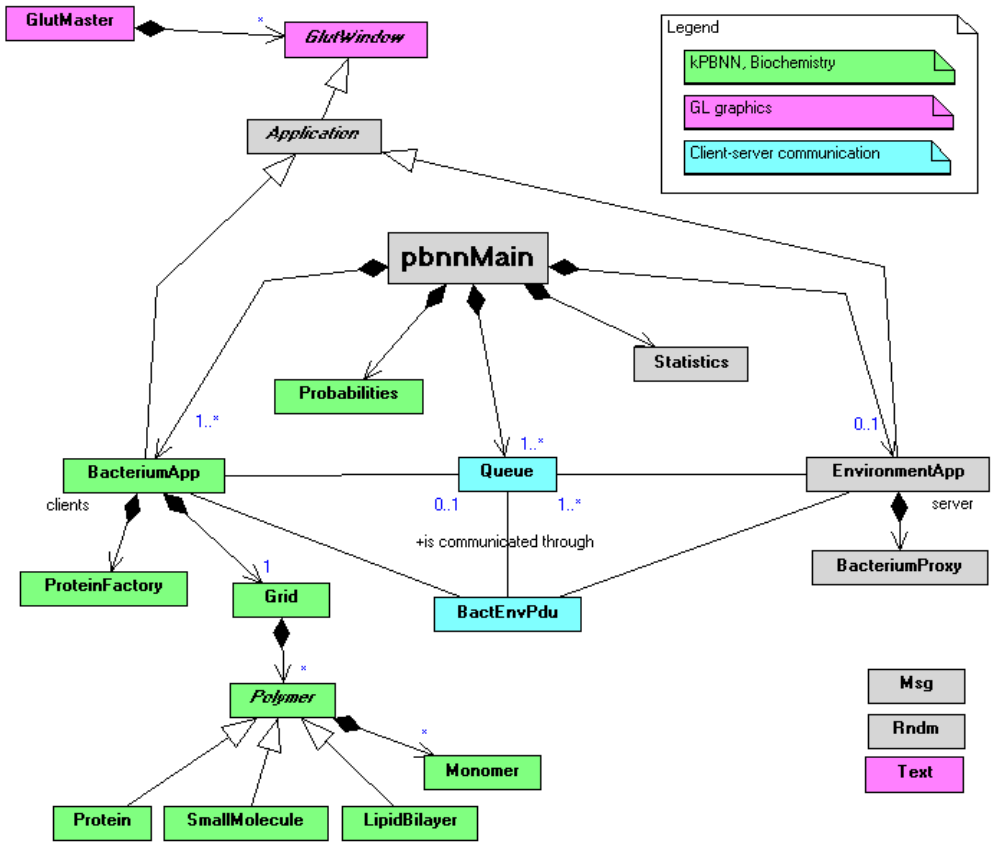


Figure 51 – UML class diagram showing all classes in the PBNN system.

The C++ code has been modularized into 20 classes, shown in the UML class diagram of Figure 51. A brief description of each class is provided in appendix H. The classes that implement the actual functionality of a kPBNN are those highlighted in green in Figure 51. Instances of these classes are first initialized, and then at each time step collaborate to implement the algorithm described in the pseudocode below.

```
Initialize the grid and all polymers
Initialize the time step counter to 0
At each time step
    Clear the grid
    For each active polymer in the system
        Traverse that polymer's string looking for rules
        When a rule is found
            Execute the rule
        Probabilistically break existing bindings (entropy)
        Place the polymer into the grid above any other polymers already there
    For each active polymer in the system
        Randomly:
            move one position right, left, up or down, or
            rotate 90° clockwise, or
            flip horizontally or vertically
    Increment the time step counter
```

Thus, at each time step, two complete passes are made through the list of polymers. The grid is completely recreated at each time step. At present only proteins have rules. A detailed

description of each rule is included in appendix B. The C++ code for each rule is implemented as a separate *doRule_* function in the Protein class. Rules implement various interactions such as bindings between monomers and binding domains within the current polymer, and monomers that are already in the grid at the same positions.

Z ordering and grid levels

Although the simulation is played out in a 2D grid with explicit x and y dimensions, it is essential that it also have a pseudo z dimension. Polymers and their monomers are placed into the grid in a certain constant order, each on top of whichever ones have already been placed there. At present the order cannot be changed. Polymers must be created in the correct order if they are to be able to detect each other.

Every polymer must be able to detect the lipid bilayer(s) that encloses it, and during the simulation must always be guaranteed access to whatever portion of the lipid bilayer it is adjacent to. Ligands must be located underneath a protein for it to recognize and be able to bind with it. Protein A can only bind with protein B if B is below A in the grid. These issues have been resolved for now by using four different grids. A polymer in a grid can see polymers and monomers in grids at a lower level unobstructed by any other polymers within its own grid.

Grid layer	Contents
1 MembraneLY	all lipid bilayers
2 SmallMolLY	all ligands and other small molecules
3 Protein1LY	proteins that bind to a lipid bilayer to form a stationary complex
4 Protein2LY	proteins that are free to move but may bind to a protein in Protein1LY

A more biologically plausible way of allowing access might be to have each polymer randomly jump up or down in the z dimension each time step, although this would prolong the time required for a simulation.

Discussion

kPBNN configurations are able to solve the 16 logic gate problems, reach a steady state, respond to the introduction of ligand, and adapt to a sustained level of ligand. But as yet no configuration has succeeded robustly at moving along a ligand gradient.

To get more insight into this negative result, I reran a simple simulation that was part of my Adaptive Systems project (Webb, 2004). In the simulation, written in NetLogo (NetLogo itself, 1999), a bacterium is an agent moving within a grid that contains patterns of ligand concentration. The bacterium has no internal structure other than a single variable that stores the ligand concentration from the previous time step. At each time step it compares this "remembered" concentration with the concentration at its new position, and uses this to bias

its normal rate of runs and tumbles which is based solely on probabilities. In the NetLogo simulation, the bacterium is robustly able to follow a ligand gradient and remain close to the maximum ligand concentration indefinitely, as shown in Figure 52.

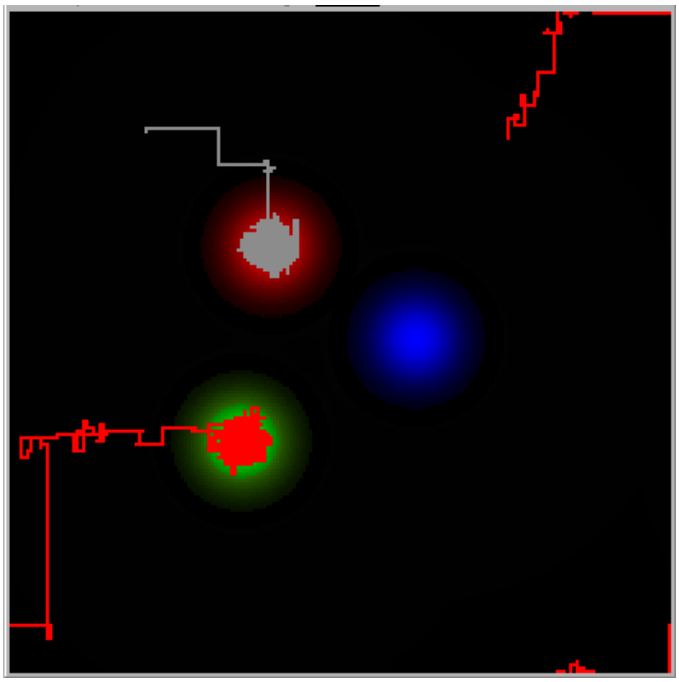


Figure 52 – NetLogo bacterial chemotaxis simulation with a delay of 0 time steps. Two bacteria, starting at random locations, successfully follow a ligand gradient and remain within the region of highest concentration. The red and green circles represent the highest concentrations of the ligand attractants (aspartate and glucose), while the blue circle is a ligand repellent (phenol). The black background contains low levels of all three ligands. All three gradients continuously increase with proximity to the circle. The tumble behaviour uses successive 90° rotations as in the kPBNN simulation. The simulation was run for 10,000 time steps.

This demonstrates that a sequence of runs and tumbles biased by the ligand concentration in the environment is sufficient to allow a simulated bacterium to move up a gradient. Why is it that the simple NetLogo simulation is able to achieve this, but the much more complex kPBNN simulation is unable to?

I introduced a variable delay into the NetLogo simulation to explore what happens as the bacterium's information about the environment becomes more and more out of date. At each time step, the moving bacterium stores the current concentration at one end of a queue, but uses the older value stored at the other end. If the queue contains 100 entries then the bacterium will use a value representing the state of the environment corresponding to its position 100 time steps ago. Figure 53 shows the much reduced ability of the bacterium to find a gradient under these conditions.

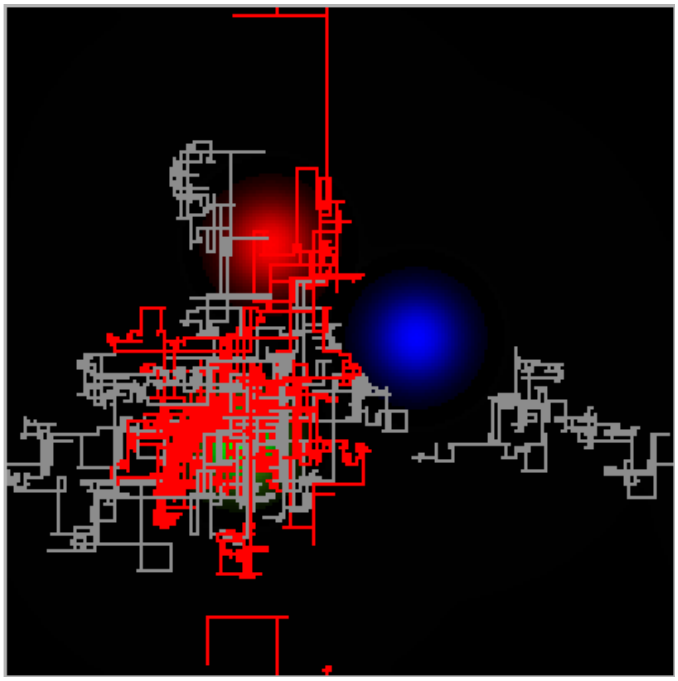


Figure 53 – NetLogo bacterial chemotaxis simulation with a delay of 100 time steps. Two bacteria, starting at random locations, follow a ligand gradient and remain very roughly within regions of highest concentration. The simulation was run for 10,000 time steps.

When the delay (and queue size) is increased to 1000 time steps, the bacterium is completely unable to follow a gradient. See Figure 54. This corresponds with the situation in the kPBNN simulation where it takes a finite amount of time, on the order of 1000 time steps, for the binding of an MCP to a ligand to have an effect on the motor. The main part of this delay is the time required for phosphorylated CheY (CheY~P) to diffuse from the chemosensory complex to the motor complex.

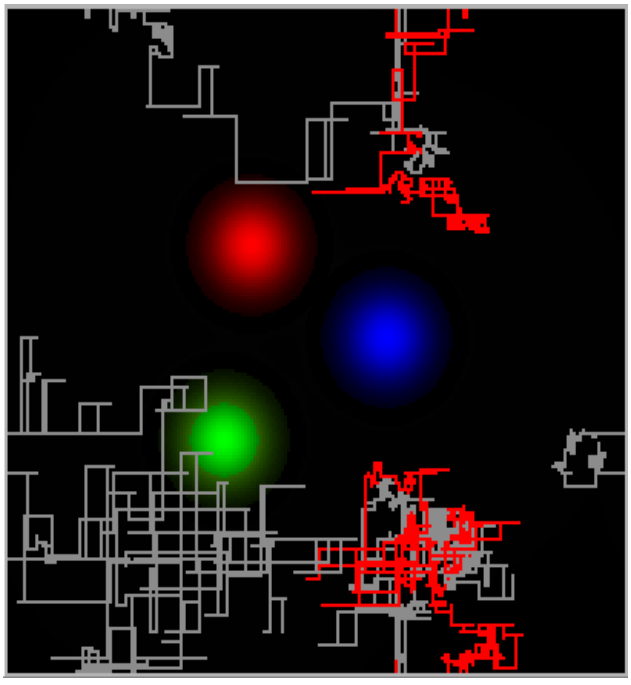


Figure 54 – NetLogo bacterial chemotaxis simulation with a delay of 1000 time steps. Two bacteria, starting at random locations, are unable to follow a ligand gradient and instead move quite randomly. Note the combination of long runs (straight lines) with other periods with more tumbles (lots of turning). The simulation was run for 10,000 time steps.

My hypothesis is that in the kPBNN simulation, the bacterium moves too rapidly for the space available. Within 1000 time steps it can easily move completely out of the region that it is in the process of becoming adapted to, especially during periods of relatively long runs. As currently implemented it does not appear that the bacterium can even in theory robustly follow a ligand gradient.

Within the NetLogo simulation I also tested various ways of implementing the bacterium tumble. In the kPBNN simulation, a tumble behaviour consists of rotating 90° to the right, while in the NetLogo simulation a tumble consists of rotating a random number of degrees to the right (from 0° to 359°). There was no change in the NetLogo results when I modified its tumble behaviour to be the same 90° rotation used in the kPBNN simulation. This suggests that the rotation algorithm used in the kPBNN simulation is not an important factor in its inability to follow a gradient.

On the complex behaviour of CheR and CheB~P

Alon *et al*. (1999, p.170) claim "feedback in which CheB [CheB~P] demethylates receptors only when they are in their active conformation" as the "molecular mechanism that leads to robust exact adaptation". Conversely, Shimizu *et al*. (2003, p.307) state that "methyltransferase CheR binds only inactive receptor". Rao *et al.* (2004, p.243) concur and stress the importance of "activity-dependent methylation". In the kPBNN simulation I assume that an MCP receptor is in an "active conformation" when its piston is extended.

In the kPBNN simulation, demethylation of MCP by CheB~P was originally much more probable if the MCP piston was extended through binding with a ligand, and CheR-induced methylation was much more probable if the MCP piston was not extended. I encountered a major problem using this configuration. When there is no ligand in the environment, the piston is never extended and only CheR is able to perform its function to any appreciable extent. The level of methylation increases continuously to a maximum value with little chance for CheB~P to balance the level through demethylation. This is at odds with the principle that biological pathways are carefully regulated to achieve balanced levels of molecules. I therefore reversed the dependencies described by Alon *et al*., Shimizu *et al*., and Rao *et al.*, and then adjusted the two probabilities. CheB~P in the kPBNN simulation has *less* probability of demethylating when the piston is extended, and CheR has *less* probability of methylating when the piston is not extended. This simple change had a large effect on the simulation results, which seems to confirm the suggested importance of these mechanisms in achieving

adaptation, while calling into question either the details of the mechanisms reported in the literature or my interpretation of these.

Future work

An immediate project is to enhance the software so that it is capable of following a ligand gradient. The hypothesis proposed in the discussion section is that the bacterium moves too rapidly within the limited space available to it. A careful analysis should be made of various time and space parameters in the simulation to ensure a consistency with those found in real bacteria and their environments. Each bacterium should have a larger local environment in which it can move and adapt to the current concentration of ligand.

A project that would assist in this effort, would be to combine the kPBNN system with previous work on CellAK (Webb & White, 2003; Webb & White, 2004a; Webb & White, 2004b), based partly on component hierarchies especially hierarchies of lipid bilayer compartments. The lack of hierarchical structure in the kPBNN simulation at the organism level makes it difficult to move the bacterium through a local environment. In a flat space each polymer must be individually moved. By contrast, in a hierarchical space, only the lipid bilayer would need to be explicitly moved because all the other polymers are inside the bacterial compartment that it defines and would automatically move along with it.

In addition to such static hierarchies, largely imposed from the top down, I would like to use an enhanced kPBNN system to explore the creation of more dynamical hierarchies (Rasmussen *et al*., 2001), which evolve largely from the bottom up. The current kPBNN system has a *monomer level*, a *binding domain level*, a *polymer level*, and a *bacterium (organism) level*. A start has also been made on defining an *environment level* that can contain multiple bacteria. Real bacteria can detect and migrate towards small molecules released by other bacteria, and can use this signalling to self-aggregate into higher level structures (Falke *et al*. 1997). All of these levels depend on the monomers, and are built from the same fundamental parts.

The current kPBNN system has a limited ability to self-assemble or self-organize. If the collection of proteins and small molecules from the bacterial chemotaxis application are placed randomly inside a lipid bilayer at the start of a simulation, they organize themselves into a partially functioning system. The proteins with hydrophobic binding domains insert themselves into the lipid bilayers. Other proteins then bind to these transmembrane proteins. Very little work has so far been done exploring this natural ability of a kPBNN to self-assemble. In the few trials that have been done with the current system, proteins usually fail to locate themselves in the most optimal configurations, and some transmembrane proteins find

themselves inserted with their intended inside part on the outside of the bacterium. But the basic capability for self-assembly is there and is a starting point for future work.

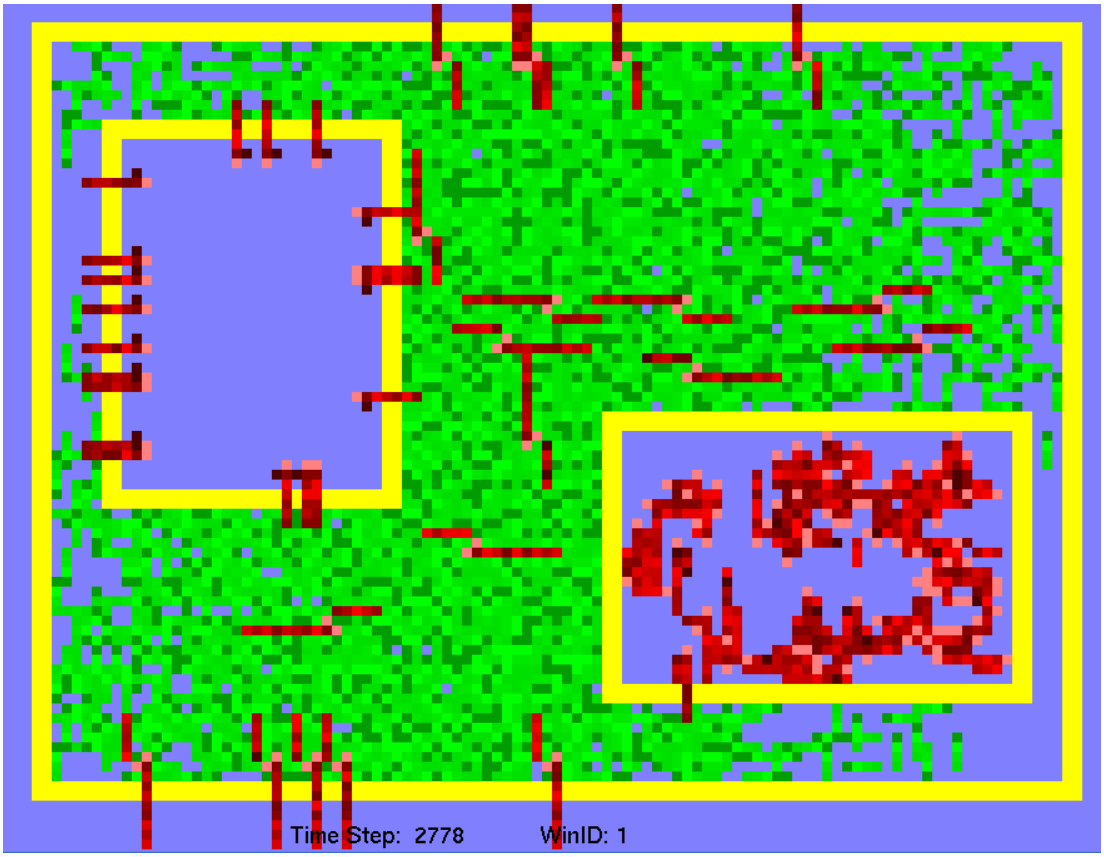


Figure 55 – A kPBNN configuration with three nested lipid bilayer compartments. The proteins in the leftmost compartment all have hydrophobic domains and have inserted themselves into the bilayer. One of the randomly generated proteins in the rightmost compartment happens to have a hydrophobic domain and has inserted itself into the bilayer. The ligands and hydrophobic proteins in the larger compartment have diffused from the centre, and some of these proteins have inserted themselves into the outer bilayer.

In other trials, random protein strings were generated and allowed to fold. When placed inside a lipid bilayer compartment, those few that happened to have hydrophobic domains inserted themselves into the bilayer. See Figure 55.

Structure, relationships between structural elements, and behaviour are fundamental concepts in a kPBNN. Each polymer has a structure consisting of a possibly-folded monomer string, relationships with other polymers implemented as multi-monomer binding domains, and behaviour provided as rules attached to specific monomers and operating on specific binding domains. These three concepts are also very important in the object-oriented (OO) approaches that dominate software development today. In previous work (Webb & White, 2003; Webb & White, 2004a; Webb & White, 2004b), it has been suggested that biological simulations can be advantageously and naturally constructed using OO approaches and tools, using the structure, relationship and behaviour capabilities these provide as fundamental building

blocks. It would be interesting to explore in detail how these OO concepts of structure, relationship and behaviour map onto those used in a kPBNN.

Enhancements that would enable use of a GA

The present software implementation has a variety of parameters that need to be optimized in order for the kPBNN system to behave adaptively. These parameters are the quantities of each type of protein and small molecule in the simulation, and the probabilities of certain key interactions occurring on each time step. One such set of parameters has been discovered through a time consuming process of trial and error. This is exactly the type of exercise that should be performed by an evolutionary or genetic algorithm (GA) (Mitchell, 1997), that would be able to search the entire parameter space in parallel for optimal solutions. Use of a GA has not been practical because the current prototype implementation runs too slowly. It takes about 15 minutes to complete a single full simulation trial of 200,000 time steps. A 2.2 GHz windows PC, simulating two bacteria at the same time, runs at 100% of its capacity.

The main reason for this was a conscious desire to not compromise the design goal of having all processing performed by individual agents acting on other individuals. For example, most of the individual objects in the system are small molecules which are indistinguishable from each other except for their location and orientation, and which could just as readily be represented as symbolic quantities with probabilities determining their positions. Also, the prototype implementation spends most of its time matching monomers, and little effort has been made to optimize these algorithms.

The architecture of the prototype implementation uses a client-server approach (Webb & Lafreniere, 1991) that can readily (although not without a considerable amount of work) be extended to run as a distributed system across multiple computers. The single server (EnvironmentApp class) currently communicates with multiple clients (BacteriumApp class) using a pair of queues. These queues are designed to be replaced by TCP/IP sockets.

An additional server could control a distributed population of bacteria clients. A GA running as part of this server could randomly create a set of parameters for each bacterium in the initial simulation generation, collect the results after 15 minutes, do selection, cross-over, and mutation to determine a new better set of bacteria for the next generation, iteratively running new generations with parameter sets that should gradually improve the ability of the bacteria to find food in the environment. A precise fitness function would need to be determined, probably quantity of ligand in the bacterium's local environment.

Some Last-Minute Results

A major contributing factor in the inability of the bacterium to follow a gradient, is the high level of random variation in the simulation. When the amount of ligand is increased or decreased by a small amount, as in a gradient, noise completely obscures the signal. See Appendix Ga for details on a series of experiments completed at the conclusion of the project after the rest of the dissertation had been printed. The noise issue, quite natural for this type of system, has been discussed in the section on Noise. The practical significance of this became especially apparent after performing additional experiments which once again failed to achieve chemotaxis. Future work on this issue must begin by first carefully exploring each contributing factor, and only then extending the prototype. The achievement of chemotaxis, in the visualizable sense of my earlier simplistic model in NetLogo, was not the major goal of this dissertation, in part because none of the models or simulations reported in the biology literature that I have seen have attempted to do this (Bray et al., 1993; Mello & Tu, 2003; Rao *et al.*, 2004; Shimizu et al., 2003; Spiro *et al*., 1997), and because there is still considerable uncertainty in the literature on what are the critical factors that allow for bacterial chemotaxis (Bray, 2002; Sourjik & Berg, 2002). Only some of these factors have been discussed in this dissertation and have been included in the simulation.

Conclusion

This dissertation has explored the concept of a protein-based neural network (PBNN), has described one particular prototype implementation, and has used this kPBNN simulation software to build and experiment with two different application domains. The intention was to offer this as a proof-of-concept that a simulated PBNN can perform some of the neural network-like tasks that Bray ascribed to them. I submit that the prototype kPBNN is able to perform such tasks, and is a potentially useful approach for modelling and simulating biological and other adaptive systems.

Bibliography

Agutter, P.; Wheatley, D. (1997). Information processing and intracellular 'neural' (protein) networks: considerations regarding the diffusion-based hypothesis of Bray. *Biology of the Cell 89*, 13-18. [PBNN hypothesis]

Alberts, B. (2002). *Molecular Biology of the Cell, 4th ed*. New York: Garland Science. [bacterial chemotaxis, p.890-892]

Alon, U.; Surette, M.; Barkai, N.; and Leibler, S. (1999). *Robustness in bacterial chemotaxis*. Nature 397, 168-171.

Becker, W. (1996). *The World of the Cell*. Menlo Park, CA: Benjamin/Cummings. [p. 706-709 has an introductory section on The Bacterial Flagellum]

Beer, R. (1995). On the Dynamics of Small Continuous-Time Recurrent Neural Networks. *Adaptive Behavior 3*, 469-509.

Bishop, C. (1995). *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Bourret, R.; and Stock, A. (2002). Molecular Information Processing: Lessons from Bacterial Chemotaxis. *Journal of Biological Chemistry 277*, 9625-9628.

Branden, C.; and Tooze, J. (1999). *Introduction to Protein Structure, 2nd ed*. New York: Garland.

Bray, D. (1990). Intracellular Signalling as a Parallel Distributed Process. *J. Theor. Biol. 143*, 215-231. [PBNN hypothesis]

Bray, D.; Bourret, R.; and Simon, M. (1993). Computer Simulation of the Phosphorylation Cascade Controlling Bacterial Chemotaxis. *Molecular Biology of the Cell 4*, 469-482. [bct 1.1]

Bray, D. (1995). Protein molecules as computational elements in living cells. *Nature 376*, 307-312. [PBNN hypothesis]

Bray, D. (2001). *Cell Movements – From Molecules to Motility, 2nd ed*. New York: Garland.

Bray, D. (2003). Molecular Networks: The Top-Down View. *Science 301*, 1864-1865. [PBNN hypothesis]

Bray, D. (2002). Bacterial chemotaxis and the question of gain. *Proc. Nat'l. Acad. Sci. 99*, 7-9.

Bren, A.; and Eisenbach, M. (2000). How Signals Are Heard during Bacterial Chemotaxis: Protein-Protein Interactions in Sensory Signal Propagation. *J. of Bacteriology 182*, 6865-6873.

Falke, J.; Bass, R.; Butler, S.; Chervitz, S.; and Danielson, M. (1997). The Two-Component Signalling Pathway of Bacterial Chemotaxis: A Molecular View of Signal Transduction by Receptors, Kinases, and Adaptation Enzymes. *Annu. Rev. Cell Dev. Biol. 13*, 457-512.

Falke, J.; and Hazelbauer, G. (2001). Transmembrane signaling in bacterial chemoreceptors. *TRENDS in Biochemical Sciences 26*, 257-265.

Hancock, J. (1997). *Cell Signalling*. Longman.

Hellingwerf, K.; Postma, P.; Tommassen, J.; and Westerhoff, H. (1995). Signal transduction in bacteria: phospho-neural network(s) in Escherichia coli? FEMS Microbiology Reviews 16, 309-321. [PBNN hypothesis]

Lauffenburger, D. (2000). Cell signaling pathways as control modules: Complexity for simplicity? *PNAS 97*, 5031-5033.

Macnab, R. (1987). Motility and Chemotaxis. In Neidhardt, F. (ed). *Escherichia coli and Salmonella typhimurium*. Washington, D.C.: American Society for Microbiology.

Mello, B.; and Tu, Y. (2003). Quantitative modeling of sensitivity in bacterial chemotaxis: The role of coupling among different chemoreceptor species. *PNAS 100*, 8223-8228.

Mitchell, M. (1997). *An introduction to genetic algorithms*. Cambridge, MA: MIT Press.

Morton-Firth, C.; and Bray, D. (1998). Predicting Temporal Fluctuations in an Intracellular Signalling Pathway. *J. Theor. Biol. 192*, 117-128. [Fig. 5 showing how fluctuation increases as the number of CheY~P in a simulation is decreased]

NetLogo itself: Wilensky, U. 1999. NetLogo. http://ccl.northwestern.edu/netlogo/. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL.

Rao, C.; Kirby, J.; and Arkin, A. (2004). Design and Diversity in Bacterial Chemotaxis: A Comparative Study in Escherichia coli and Bacillus subtilis. *PLOS Biology 2*, 239-252.

Rasmussen, S.; Baas, N.; Mayer, B.; Nilsson, M.; and Olesen, M. (2001). Ansatz for dynamical hierarchies. *Artificial Life 7*: 329-354.

Segall, J.; Block, S.; and Berg, H. (1986). Temporal comparisons in bacterial chemotaxis. *Proc. Natl Acad. Sci. USA 83*, 8987–8991.

Shimizu, T.; Aksenov, S.; and Bray, D. (2003). A Spatially Extended Stochastic Model of the Bacterial Chemotaxis Signalling Pathway. *J. Mol. Biol. 329*, 291-309.

Sourjik, V.; and Berg. H. (2002). Bacterial Chemotaxis: Resolving the Gain Paradox. *ICSB 2002*, Stockholm, Sweden.

Spiro, P.; Parkinson, J.; and Othmer, H. (1997). A model of excitation and adaptation in bacterial chemotaxis. *Proc. Nat'l. Acad. Sci 94*, 7263-7268.

Webb, K. (2004). *Adaptive Systems Project - Architectural Models of Bacterial Chemotaxis*. Brighton: University of Sussex.

Webb, K.; and Lafreniere, L. (1991). *Oracle Distributed Systems – A C Programmer's Development Guide*. Blue Ridge Summit, PA: McGraw-Hill.

Webb, K.; and White, T. (2003). UML as a Cell and Biochemistry Modeling Language. *Carleton University Cognitive Science Technical Report 2003-05*. http://www.carleton.ca/iis/TechReports . Accepted for publication in *BioSystems* journal.

Webb, K.; and White, T. (2004a). Combining Analysis and Synthesis in a Model of a Biological Cell. *SAC '04*, March 14-17, 21004, Nicosia, Cyprus.

Webb, K.; and White, T. (2004b). Cell Modeling using Agent-based Formalisms. *AAMAS 2004,* July 20-23, 2004, New York, U.S.A.

Wolf, B.; Kraus, M.; Brischwein, M.; Ehret, R.; Baumann, W.; and Lehmann, M. (1998). Biofunctional hybrid structures – cell-silicon hybrids for applications in biomedicine and bioinformatics. Bioelectrochemistry and Bioenergetics 46, 215-225. [p.219-220 on PBNN hypothesis]

Glossary

The glossary contains a number of specialized terms. Most are defined in terms of how they are used in the PBNN system.

amino acid: A monomer out of which protein polymers are constructed. There are 20 types of amino acid.

aspartate: A type of ligand found in the bacterial environment. Aspartate is also a type of amino acid.

bind: Two polymers bind when the monomers in a binding domain match. Once bound they move, or don't move, as a single unit.

binding domain: A sequence of monomers that could bind with another compatible sequence of monomers. A binding domain is usually four monomers. They may occupy adjacent positions within one row, or within one column of a polymer. A binding domain may include monomers from two or more different polymers if these are immediately adjacent to each other.

chemotaxis: Chemical-seeking behavior in bacteria and other organisms. Chemotaxis results from a combination of runs and tumbles.

covalent bond: The strong chemical bond that holds the atoms of a molecule together.

cytoplasm: The water, polymers and monomers that make up the inside of bacteria and other cells.

enzyme: A protein that actively changes the molecular properties of specific proteins or other polymers that it comes into contact with.

hydrophobic: Water hating. Some monomers and binding domains of polymers are repelled by water and seek the interior of lipid bilayers and other locations that lack water.

kinase: An enzyme that phosphorylates another protein.

ligand: A small molecule that acts as a signal in a signal transduction pathway.

lipid: A monomer that aggregates to form lipid bilayers (membranes).

lipid bilayer: A double layer of lipids that surround some space, such as the cytoplasm, and keep that internal space separate from the external space. A lipid bilayer with proteins embedded within it is called a membrane.

methanol: A small molecule that is enzymatically attached to a protein during methylation.

methyl-accepting chemotaxis protein (MCP): A chemosensory receptor protein, of which Tar, Tsr, Trg, and Tap are specific types. These proteins can all be methylated as a way of regulating their activity level.

methylation: The addition of a methyl group, derived from methanol or similar molecules, to an amino acid in a protein. Methylation, and demethylation which is the removal of methyl groups, are under the control of enzymes. Methylation is a way of regulating molecular processes.

monomer: The fundamental building block of all polymers in the PBNN system described in this dissertation. All such simulation monomers are the same size.

motor: *E. coli* bacteria contain a small number (typically 6 to 8) of rotary motors that spin some 100 times or so per second, either clockwise or counter-clockwise. This motion turns flagella, long strings that extend behind the bacteria and that provide mobility. The motors are under adaptive control provided by a protein-based neural network inside the bacteria.

OpenGL: Platform-independent software that provides 2D and 3D graphics capabilities on a computer.

parameter: In the PBNN system described in this dissertation, the quantity of a polymer or a probability value.

pathway: A sequence of protein-protein and protein-small molecule interactions that combine to convert some initial chemical or signal into some product or result.

phosphorylation: The addition of a phosphoryl group, typically derived from ATP, ADP, or AMP, to an amino acid in a protein. Phosphorylation, and dephosphorylation which is the removal of phosphoryl groups, are under the control of enzymes. Phosphorylation is a very common way of regulating molecular processes. Many proteins are only active when phosphorylated.

piston: A subsequence (single row in the PBNN system) of an MCP or other transmembrane protein (TMP) that changes its state by becoming extended when it binds with a ligand.

polymer: A sequence of covalently-bound monomers that may be folded into a 2D shape. Proteins, lipid bilayers and small molecules are the three types of polymer found in the PBNN system.

run: A bacterial behavior that involves a series of forward movements.

signal transduction: A sequence of molecular events inside the bacterium that result when an MCP or other TMP binds to a ligand signal such as aspartate.

string: A sequence of symbols, such as the sequence of amino acid monomers that make up a protein. A string is one-dimensional (1D) and has a single starting point and single end point. A string can be folded into a two-dimensional (2D) shape.

tumble: A bacterial behavior. A break in a run, in which a bacterium rotates to face in a new direction.

Acronyms and Abbreviations

ADP	adenosine diphosphate
AMP	adenosine monophosphate
ANN	artificial neural network
ATP	adenosine triphosphate
CCW	counter clockwise
Che	chemotaxis
CW	clockwise
FliM	flagellum i motor (gene and protein)
GA	genetic algorithm
kPBNN	Ken's concept of a protein-based neural network
MCP	methyl-accepting chemotaxis protein
PBNN	protein-based neural network
PDP	parallel distributed processing
Tap	transducer for dipeptides (gene and protein)
Tar	transducer for aspartate (gene and protein)
TMP	transmembrane protein
Trg	transducer for ribose and galactose (gene and protein)
Tsr	transducer for serine (gene and protein)
UML	Unified Modeling Language
XOR	exclusive or

Appendices

The appendices are in separate files.