

# Potts Model 2.0 Collaboration Requirements

Chris Mueller

September 20, 2003

## Abstract

This document is a high-level requirements overview for the Potts collaboration. Its main purpose is to define the scope of the project, standardize terminology, and identify starting points for the research and development phase. In its current draft form, it is meant to stimulate discussion and provide a point of reference for terminology.

## 1 Overview

The Biocomplexity group is developing a Potts model-based simulation of chick limb development. Their simulation environment currently consists of a C++ program that implements the Potts model and a very basic OpenGL viewer used to visualize the results. This environment has a few shortcomings that make experimentation with the Potts model challenging:

The C++ implementation is single threaded, making experimentation on large systems impractical.

Adjusting the parameters and model requires writing code and rebuilding the application.

There is no version tracking in place to enable auditing and recreating past experiments.

The visualization environment does not show the entire system accurately. For example, chemical concentrations, which play an important role in the model, are not visible.

The initial goal of the collaboration with the CS department is to address these immediate problems. The remainder of this document details the Potts model, how it is used as an experimental tool, and what new tools will be developed to support the project.

## 2 Potts Model Overview

The Potts Model is a statistical mechanics model based on the Ising Model. It solves energy functions in systems consisting of many elements arranged on an n-dimensional lattice. The lattice is set up with an initial (possibly random) configuration of values at each lattice site. Each lattice site contains an element that belongs to a biological cell. By convention, these elements are called spins,  $\sigma^1$ . Each element that is part of an individual cell has the same spin; within a cell, all elements are indistinguishable from one another, save for their location.

In an implementation of the Potts Model, there are many cell types that supply terms for the energy calculations. By convention, the type of each spin is referred to as  $\tau(\sigma)$ . Each  $\sigma$  can only have one type, but each type can have many  $\sigma$ 's. This is essentially saying that there can be many different cells that have the same type. In the chick limb model, there are two main types of cells: biological cells and ECM (extra-cellular material). The biological cells are further divided into different types.

---

Version 1.0 is the current implementation

<sup>1</sup>The name spin is a holdover from the original use of the Ising Model to model ferromagnetic material

The Potts Model works by randomly selecting lattice sites and flipping the spin of a neighboring site to the spin of the random site if the energy for the flip is favorable. In other words, cells attempt to grow into new lattice points. To help avoid local minimums, the Metropolis algorithm is used to allow a temperature that can encourage things to flip when they wouldn't otherwise.

The conditions that affect the energy are the terms of the Hamiltonian. For any potential flip, the contributions of each term for the cell type are summed to determine the total energy change for the flip. Examples of things that affect the energy are cell volume, cell area, and adhesion interaction energy, or stickiness. If the sum of all conditions leads to a lower energy state, then the flip is more likely to occur.

Monte Carlo steps are executed to update the lattice sites. Each step performs the following:

1. Choose a random lattice site that has neighboring sites with different spins.
2. Pick a random neighbor
3. Solve the Hamiltonian for the new site based on flipping it to be part of the original cell.
4. If the energy change is favorable, flip
5. Diffuse any chemicals
  - (a) Choose a set of lattice sites at random
  - (b) For each site, diffuse the chemical concentration across the neighboring sites.

## 2.1 Aside: Chemical Diffusion

The diffusion step is currently not implemented directly in the Potts Model. Chemical concentration movement is external but does take place on the lattice. The concentration at any given lattice point affects the energy function but flipping has no effect on concentrations. It is hoped that with a finer grid and new Hamiltonian terms, chemical diffusion can be moved into the model.

## 2.2 Hamiltonians

The Hamiltonian is the sum of individual terms from the randomly selected site,  $\sigma_i$ , and its neighbors,  $\sigma_j$ . The terms of the Hamiltonian in the current implementation model volume, area, chemical energy, and adhesion. The overall  $\Delta E$  is the sum of these terms. They are defined as:

**Adhesion**  $\sum_{i,j} J(\sigma_i, \sigma_j)(1 - \delta(\sigma_i - \sigma_j))$  where  $J(\sigma_i, \sigma_j)$  is the adhesion coefficient between  $\tau(\sigma_i)$  and  $\tau(\sigma_j)$ .

**Volume Constraint**  $\sum_{i,j} \alpha(\sigma_i)(V(\sigma_i) - V_T(\sigma_i))^2$  where  $\alpha(\sigma_i)$  is the volume coefficient of  $\tau(\sigma_i)$  and  $V_T(\sigma_i)$  is the ideal volume for  $\tau(\sigma_i)$

**Surface Area Constraint**  $\sum_{i,j} \beta(\sigma_i)(A(\sigma_i) - A_T(\sigma_i))^2$  where  $\beta(\sigma_i)$  is the area coefficient of  $\tau(\sigma_i)$  and  $A_T(\sigma_i)$  is the ideal area for  $\tau(\sigma_i)$

**Chemical Energy** (see code)

## 3 Potts 2.0

The next version of the Potts Model simulation environment will provide a new implementation of the model described above that can take run in multi-processor environments.

Discussions with the Biocomplexity group yielded a few major use cases that will drive some of the design choices. These are:

Adjust the coefficients in the Hamiltonian over many runs, based on results of previous runs. Note that the adjustments are such that they preclude systematic sweeps across parameters.

Define initial conditions. (not necessary now, but duplicating (ie, setting the random seed) should be supported right away.)

Adjust global parameters (eg, temperature, number of Monte Carlo steps)

Define the different Hamiltonians

Define cell types and their Hamiltonian coefficients.

Visualize results as a 3D volume that shows cells along with chemical gradients.

Visualize 2D slices of the 3D volume.

Note that the following sections are just placeholders. The information is in no way meant to be complete or final. (what's the code for smiley in LaTeX?)

### **3.1 Parallel Implementation**

The first version of the parallel code will subdivide the grid across processors, with each processor performing N Monte Carlo steps. The main type of communication is to preserve the values of boundary sites (sites that have one or more neighbors on other processors).

Open Issues:

I'm assuming we'll be using MPI. I need to get a development environment setup for MPI.

I'll also need to learn how to use the cluster resources (do we have a small cluster for development?).

### **3.2 Initialization and Parameters**

The first version of the code will most likely use linked object files to supply all parameters and Hamiltonians. Another option that is being considered is to wrap the main program in a scripting language and define the parameters in that environment. Wrapping it up in a scripting language has the benefit of removing the compile step from successive experiments and provides a nice separation of input parameters and the simulation code.

### **3.3 Visualization**

The new visualization system will be a VTK-based volume and image viewer, possibly just Mayavi ([mayavi.sourceforge.net](http://mayavi.sourceforge.net)). VTK provides a nice abstraction on top of OpenGL and implements many high level routines for scientific visualization.

### **3.4 Version Tracking**

Version tracking will be handled using CVS. There will also be a SourceGrid project setup to manage the repository and other administrivia.

The CS group will also hold a few tutorials to get the Biocomplexity group up to speed with CVS, Makefiles and other common software engineering tools.

## **4 Development Iterations**

First iteration should be more or less a straight parallel port of the current code. This will get us up and running quickly and give us a foundation for improvements. Next, a first pass at the visualization system will be implemented.

## A Some (not all) Definitions

**Cell** 1) A biological cell. 2) Occasionally in conversation, this will be used for lattice site

**Cell Type (or just type) (t)** Cells of the same type have the same properties. The properties may have different values for each instance, but the overall set of properties is the same. In OO terms, this is the Class.

**Hamiltonian** The energy function. The goal of the Potts simulation is to minimize the global energy function across all lattice points. Each lattice point contributes coefficients to the global hamiltonian.

**J** The adhesion interaction energy, or stickiness, between cells.

**Lattice** The uniform nxn or nxnxd grid used as the physical framework for the model.

**Lattice Site** A distinct point on the lattice. Also referred to as pixel or voxel.

**Spin (sigma)** A single instance of a cell occupying one or more lattice sites. In OO terms, this is the instance of a class.

**Spin Flip** Changing the spin of a given lattice point. This is analogous to the cell extending into a new physical space.

## B Current Program Structure

The current implementation of the Potts Model for single threaded systems has the following structure:

```
InitializeLattice()
```

```
Do for n Monte Carlo steps:
```

```
  while (site has no alien neighbors):  
    site = ChooseRandomLatticeSite()
```

```
  neighbor = ChooseRandomNeighbor(site)
```

```
   $\Delta E$  = CalculateEnergyForFlip(neighbor, type(site))
```

```
  if  $\Delta E < 0$ :
```

```
    flip(site, neighbor)
```

```
  elif rand() <  $e^{-B \Delta E}$ : // Metropolis step
```

```
    flip(site, neighbor)
```

```
if diffusion is calculated on the grid (current implementation):
```

```
  Do for n diffusion steps:
```

```
    diffusionSite = ChooseRandomLatticeSite()
```

```
    CacluateDiffusion(diffusionSite)
```