1 Memory usage

In this section, I will discuss the design pattern used in the new system to cover the flaws in memory usage in the original model. The new design aims to further reduce the unnecessary memory requirement for better space efficiency. While space efficiency is an important factor in system design in general, it is particularly important in our system because less memory requirement for each individual simulation will allow us to run a larger number of simulations in parallel(or only CPU). Smaller memory requirement for each simulation also means less data transfer between GPU and CPU, which is another bottleneck identified during the preliminary stage of the project. Given those two factors, it is easy to see that space efficiency is important to performance of the new system.

1.1 Methodology

One of the main data structure in the system is a large three-dimensional array named *baby_cl*, which holds concentration levels over several delayed time steps for all species and all cells. In parameter input, the length of delay is directly associated with each individual reaction. Since concentration level in *baby_cl* is stored for each species, however, there is no direct attribute in model information that will determine the delay size in *baby_cl*. To handle this problem, original systems decides the number of time steps to keep based on the maximum delay size across all reactions to ensure that each species is kept in the system long enough for possible reactions. However, not all species are related in a reaction with the maximum delay. The maximum delay can be as long as $1,200$ time steps for some reactions such as gene transcription and much shorter for reactions such as mRNA translation; some reactions in the system may not even have a delay. The original system does not fully utilize memory allocated not all species need to be stored for maximum delay to provide history data for future simulations.

To decrease the size of allocated memory for *baby_cl*, I redesigned data structures so that the system will now only allocate necessary space to hold enough history data for future usage. I first added a related species section in the data structure ($reaction.cpp$) describing relations between species and reactions. Next, the system iterates through all reactions to find the species according to reaction.cpp and add delay size of the reaction to a set specific to each species. After all iterations, the maximum delay size needed for a species is the maximum delay within the subset. Based on this information, the system will then create

a large one-dimensional array to hold all concentration levels and place different wrappers according to its maximum delay size for each species. For the system to access concentration levels of each species, I created another much shorter one-dimensional array to hold the pointers to the beginning point of each species in the larger array. Each concentration level access starts by locating subsection of *baby_cl* through species id and completes through cell number and time step.

1.2 Results

This design will reduce size of *baby_cl* on running environment greatly and thus allow simulation of more parameter sets at the same time. Using the segmentation clock project for example, the new *baby_cl* data structure uses 70% less memory than the original one. Through this improvement, the number of simulations that can stay on the GPU at the same time increases more than two folds and consequentially, the average runtime for each of the parameter sets will be reduced to one third of its original runtime.