12-1-2015

Geometric Algorithms and Data Structures for Simulating Diffusion Limited Reactions

Shaun Bloom

Shaun Bloom

Candidate

Computer Science

Department

This dissertation is approved, and it is acceptable in quality and form for publication:

Approved by the Dissertation Committee:

Dr. Shuang Luan

Dr. Lawrence Williams

Dr. Trilce Estrada

Dr. Yin Yang

Geometric Algorithms and Data Structures for Simulating Diffusion Limited Reactions

BY

SHAUN BLOOM

B.S. PHYSICS, UNIVERSITY OF NEW MEXICO, 1989

DISSERTATION

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Doctor of Philosophy

Computer Science

The University of New Mexico
Albuquerque, New Mexico

December, 2015

Dedication

I dedicate my dissertation work to my wife, family and many friends. A special thanks goes to my wife whose encouraging words helped me along the way. I will always appreciate hers and their patience with me during this time.

Acknowledgements

I would like to express my deep appreciation and gratitude to my advisor, Dr. Shuang Luan, for the patient guidance and mentorship he provided to me, from the application process into the doctoral program through to completion of this degree. My meetings with Dr. Luan have been both educational and inspirational. With regard to geometric data structures, he’s a deep pool of knowledge and I found our meetings to be full of creativity and genuinely fun. I am truly fortunate to have had the opportunity to work with him and I thoroughly enjoyed having discussions in such a creative atmosphere. I would also like to thank my committee members, Drs. Lance Williams, Trilce Estrada and Yin Yang for their friendly guidance, thought provoking suggestions, and the general collegiality that each of them offered to me over the years. In a similar vein, I’d like to recognize Dr. Dorian Arnold for providing help and direction with the parallel implementation of my algorithm.

Geometric Algorithms and Data Structures for Simulating Diffusion Limited Reactions

by

Shaun M. Bloom

B.S. Physics, University of New Mexico, 1989

Ph.D. Computer Science, University of New Mexico, 2015

ABSTRACT

Radiation therapy is one of the most effective means for treating cancers. An important calculation in radiation therapy is the estimation of dose distribution in the treated patient, which is key to determining the treatment outcome and potential side effects of the therapy. Biological dose – the level of biological damage (e.g., cell killing ratio, DNA damage, etc.) inflicted by the radiation is the best measure of treatment quality, but it is very difficult to calculate. Therefore, most clinics today use physical dose - the energy deposited by incident radiation per unit body mass - for planning radiation therapy, which can be calculated accurately using kinetic Monte Carlo simulations. Studies have found that physical dose correlates with biological dose, but exhibits a very complex relationship that is not yet well understood.

Generally speaking, the calculation of biological dose involves four steps: (1) the calculation of physical dose distribution, (2) the generation of radiochemicals based on the physical dose distribution, (3) the simulation of interactions between radiochemicals and bio-matter in the body, and (4) the estimation of biological damage based on the distribution of radiochemicals. This dissertation focuses on the development of a more efficient and effective simulation algorithm to speed up step (3). The main contribution of this research is the development of an efficient and effective kinetic Monte Carlo (KMC) algorithm for simulating diffusion-limited chemical reactions in the context of radiation therapy. The central problem studied is - given n particles distributed among a small number of particle species, all allowed to diffuse and chemically react according to a small number of chemical reaction equations - predict the radiochemical yield over time. The algorithm presented makes use of a sparse grid structure, with one grid per species per radiochemical reactant used to group particles in a way that makes the nearest neighbor search efficient, where particles are stored only once, yet are represented in grids of all appropriate reaction radii. A kinetic data structure is used as the time stepping mechanism, which provides spatially local updates to the simulation at a frequency which captures all events - retaining accuracy. A serial and three parallel versions of the algorithm have been developed. The parallel versions implement the kinetic data structure using both a standard priority queue and a treap data structure in order to investigate the algorithm's scalability. The treap provides a way for each thread of execution to do more work in a particular region of space. A comparison with a spatial discretization variant of the algorithm is also provided.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

According to the American Cancer Society, about 13 million Americans today have cancer and one million more are being diagnosed every year [1]. One out of every two men and one out of every three women will be diagnosed with cancer in their lifetimes. According to a recent report from CNN [2], cancer will become the most deadly disease in the U.S. by 2030. Thus, it is important to develop effective methods for treating cancer.

A cancerous cell is a mutation of a healthy cell and mutations are, by their nature, random. To date, there are more than two hundred classes of cancer and within those classes there are many variations – usually classified by the type of cells they affect. There are common characteristics among them, but they are all significantly different from one another. They grow at different target depths within the body, affecting tissues of varying density and afflicting differing biological systems. One of the few treatments we know of today that is effective against most types of cancer is radiation therapy – a process of delivering lethal doses of ionizing radiation into tumorous regions of living tissue in an effort to kill the tumorous mass. Since radiation also damages the healthy and normal tissues along its beam path, the goal of radiation therapy treatment planning is to maximize the dose to the target tumors while minimizing the dose to the nearby healthy tissues and structures. Therefore it is important to accurately calculate the dose distribution for any designed radiation treatment. Generally speaking, there are two types of dose that are of interest clinically - the physical dose and the biological dose. The physical dose refers to the energy absorbed per unit mass in a biological medium following irradiation (measured in Joules per kilogram). Biological dose refers to the biological effectiveness (i.e., cell killing probability) of the beam irradiation. Living cells are very likely to be rendered inoperative when the helical strands of DNA in the nucleus are broken into two pieces (a double strand break) because it is very difficult for the cell to repair [3]. Double strand breaks are caused directly by the impact of high energy particles or ions, or indirectly by reaction with free radicals that have been generated from other radiochemical reactions. If one knows the density of these particles, ions and free radicals – at a sufficiently small scale - an accurate estimate of biological dose (cell damage) can be made. Such an estimate would be superior to an estimate based solely on physical dose, but the physical dose is much easier to calculate and can be obtained using Kinetic Monte Carlo simulation. Kinetic Monte Carlo simulations arise in many areas of study including chemistry, traffic networks, ecology, surface diffusion and growth, and evolutionary game problems in epidemiology. KMC methods provide a time evolution of random processes occurring in nature. They are a simple and accurate means of modeling real world physical problems, albeit at the expense of time and often significant computer resources. Various KMC engines providing this type of simulation are in use today, such as Geant4-DNA [4] [5] [6] [7], Fluka [8], and MCNP [9].

Most clinics today use physical dose as a measure of treatment quality. Even though physical dose correlates with biological dose, the two types of dose exhibit a complex non-linear relationship that has yet to be determined and remains an active area of study.

The biological dose deposited by therapeutic radiation takes place in the following setting within living tissue. First, all physical interactions take place, where the molecules in the cell undergo elastic and inelastic physical interactions. This stage – the physical stage - is characterized by energy deposition and is extremely fast, lasting only about 1 femtosecond (10^{-15} second). The physico-chemical stage follows this, lasting roughly from 1 femtosecond to 1 picosecond (10^{-12} second). This short-lived stage involves the transfer of energy to the biological environment and is characterized by thermalization and solvation of subexitation electrons, electronic hole migration, and electronic recombination. The result of the physico-chemical stage is an array of radiolytic species which are thermalized and subject to the diffusion equation. Finally, the chemical stage starts and lasts from 1 picosecond to 1 microsecond (10^{-6} second). In this stage, Brownian motion governs particle movements, where the thermalized radiolytic species diffuse in the medium reacting with each other and with the biomolecules. Kinetic Monte Carlo simulations calibrated with experimental data provide the most accurate means for simulating these three stages, however Monte Carlo simulation of the third stage is particularly demanding on computational resources. This dissertation focuses on simulation of the third stage, i.e., the chemical stage.

Simulation of the chemical stage means tracking the locations of all radiolytic particles, simulating chemical reactions as the reactants drift within reaction range of one another. At any single instant in time, if the reactants are of the same chemical species, this is a description of the well-known geometric closest pair problem, which may be stated as: given the locations of N points in a metric space, return the pair of points that are closer to one another than any other pair. This is an apt description of what it means to simulate a chemical reaction because knowing the closest pair of particles at any instant means knowing the participants of the next reaction. If the reactants are not of the same species, this is an instance of the variant known as the bi-chromatic closest pair problem where we require the solution to contain points of differing colors (i.e., species). Because there are many reaction types to simulate and each one has a different reaction radius, there are many definitions of closeness that must be used. As time progresses, the particles move to new locations by diffusion and we say the simulation is kinetic. As reactions occur, reactants are removed from the simulation and products are added, so we say that the simulation is dynamic. We can therefore think of these simulations as solving M simultaneous kinetic and dynamic closest pair or bi-chromatic closest pair problems given M reaction types and N particles. Given the assumption that the human body is composed mostly of water, this leads us to a statement of the task at hand which may be stated as, given n radiolytic particles arranged in a radially symmetric, but random, pattern (a particle beam), and a variety of molecular species in a bath of H_2O molecules all subject to diffusion over a time period of $1\ \mu s$, observe and process all chemical reactions, reporting the radiochemical yield over time.

An algorithm and results are presented here that shows how the simulation times can be shortened. The key to the improved run times is a new algorithm for solving the dynamic and kinetic closest and bi-chromatic closest pair problems.

1.1 Executive Summary of Work Presented

This dissertation lays the ground work for the accurate calculation of biological dose and provides an efficient means for doing so. Efficient serial and parallel kinetic Monte Carlo algorithms for estimating dose are presented.

Description of the relevant physics. The physics of radiation chemistry, as pertinent to the presented algorithm, are presented. Areas covered are Compton scattering, the photo-electric effect, and elementary particle pair production. Definitions of stopping power, linear energy transfer, and relative biological effectiveness are given.

Kinetic Monte Carlo. The kinetic Monte Carlo method used to model the physics of the problem at hand is described.

Nearest Neighbor Search. The optimization problem known as the bi-chromatic nearest neighbor search is at the heart of the algorithm presented here. The best known algorithm for computing the static version of the nearest neighbor search is summarized. It is based on the use of grids at multiple resolutions and this provides helpful insights to understanding the new proximity grid method. An efficient mono-chromatic dynamic near neighbor search using multiple binary trees in a wegde structure is also summarized. It is an example of a dynamic nearest neighbor search for moving points in a plane.

Kinetic Data Structure. Traditional time domain codes advance time in their simulations in small, globally constant time steps. In cases when the simulated particles number in the tens or hundreds of thousands, the simulation can become bogged down by superfluous simulation activity that does not contribute to the final result. To mitigate this effect, the time step size can be increased, but increasing the size of the time step can cause important events to be missed, so this is not always the best solution to the problem. A kinetic data structure provides an alternative to the traditional constant time step method and it is used in the proximity grid algorithm. We describe this data structure in detail.

Parallelization. The proximity grid method has been parallelized with appropriate modifications to the data structures. The kinetic data structure is implemented with both a priority queue and with a treap. We show that the algorithm scales well when implemented with a priority queue given that appropriate modifications to the grid structure are made.

1.2 Summary of Original Work

Our contribution is a novel algorithm, using a nearest neighbor search technique that works well in the case when particles are in motion and when the definition of a near neighbor requires particles of differing species. Its development is motivated, in part, by the desire to improve the Geant4-DNA toolkit by providing a better method for finding the nearest neighbor. The Geant4-DNA toolkit currently makes use of a kd-tree to perform nearest neighbor searches. A kd-tree provides fast search time but does not offer fast insertion or deletion time, thus it is inefficient in the case when the occupants of the tree (the particles) are in motion. Our method implements the polychromatic nearest neighbor search using a directed acyclic graph of particle groups. The groups are non-empty cells in grids with cell sizes of every reaction radius of interest. The grids offer linear time searches at all relevant reaction radii while at the same time supporting facilities for moving particles between grid cells (and therefore grid cell groupings) with a minimum of overhead.

Time step sizes for these simulations are on the order of fractions of a picosecond. To attain efficient operation, we do not advance the simulation globally at this small time resolution because doing so would result in inefficient operation. Instead, we advance the simulation in a spatially local manner, making use of a kinetic data structure [10] [11] [12] [13] to do so.

A parallel version of the algorithm is also presented. Ideally, this algorithm would run on a radiation oncologist’s desktop workstation, so MPI (message passing interface) technology – best suited for multi-workstation use – was not chosen as the parallelization technology. GPGPU (general purpose graphical processing unit) was also considered, but at this time memory on graphics cards is limited to 4 GB and we wanted this algorithm to be useful in the case when the motions of billions of particles are simulated. OpenMP was therefore selected for the implementation. The main challenge faced was in preserving the physics of the simulation. That is, care needed to be taken to design a system whereby computer processing on no single CPU advanced simulation time past that of any other CPU within a small tolerance. This is challenging since the relative densities of particles in the simulation change over time and the algorithm needed to be insensitive to such differences in order to keep any one CPU from “lagging” behind the others. Parallelization of the kinetic data structure is handled using multiple priority queues, resulting in a time stepping mechanism that operates in a spatially mixed and randomized manner. All operations done at every time step are spatially local (saving run time), and despite the fact that the data structures use shared memory, no mutexing or other expensive inter-process communication mechanism is required. Three versions of the algorithm are presented. Two (the standard priority queue and treap variants) require the use of a ticket acquisition mechanism to ensure that regions to be processed don’t overlap one another. Ticket acquisition is performed serially, and is therefore a potential bottleneck when a large number of cores are used. This bottleneck can be overcome by processing multiple particle movements after acquiring a ticket. To make this possible, the kinetic data structure is implemented with a treap (a binary tree that also has the properties of a heap). The priority key of the treap is used as the local time step time, and the heap key of the treap is used to represent the ticket index for the region in which the particle movement takes place. The third method uses spatial discretization in exchange for the ticket acquisition mechanism as the method of ensuring that different threads operate in different regions of space.

1.4 Organization

Chapter 2 presents the background material necessary to understand the relevant concepts in radiation therapy and how those concepts are incorporated into related computer simulations. Ionizing radiation as well as relevant physical quantities are defined and a description of the relevant physical processes are presented. Chapter 3 provides details on a new algorithmic approach called the proximity grid method that offers a faster approach than the current state of the art. Our unique solution to the nearest neighbor search is presented. Details are also given for parallelization of this algorithm and results are discussed in chapter 4. Chapter 5 offers a discussion.

CHAPTER 2

BACKGROUND

The following sections briefly present the field of radiation therapy and Monte Carlo simulation of therapeutic radiation in living tissue. Section 2.1 gives an overview of the physics involved in radiation therapy as well as some of the terminology used, and it explains some of the factors that make biological dose estimation difficult.

Sections 2.3 and 2.4 describe the three stages of activity that follow therapeutic irradiation within living cells and the Monte Carlo methodology used to simulate them in a computer.

A state-of-the-art Monte Carlo tool called the Geant4-DNA toolkit is discussed next. The goal of this research is to perform a simulation on-par with what is done in Geant4-DNA and to improve on the simulation time.

2.1 Radiation Therapy

Radiation therapy is the medical use of ionizing radiation to destroy cancer cells within living tissue. The radiation is delivered in the form of either high energy photons (X-rays or gamma rays), or as bombardment by high energy, heavy particles (protons, neutrons, or heavier ions). In both cases the goal is to kill the cancerous tissue by ionizing atoms within its cells. Ionization - the acquisition of electrical charge by an atom or molecule due to the gain or loss of electrons - alters the chemical structure of biomolecules, including the DNA molecule. X-rays are photons that contain enough energy to break chemical bonds and ionize atoms, but photons of longer wavelengths (thus lower energies) do not. For example, a whole body exposure to a 70 kg man of only 4 J/kg of X-ray radiation is lethal, but this same amount of energy in the form of heat (i.e., many more, long wavelength, low energy, infra-red photons) raises the same quantity of water by only 0.002℃ - a harmless exposure. Thus, when we discuss ionizing radiation by photons, we mean X-rays or gamma rays. We ignore ultraviolet radiation (which is also ionizing) because it is not energetic enough for therapeutic purposes. Radiation in the form of electrons, protons or heavier ions, if energetic enough, is also ionizing. The greater the mass of the particles used, the greater the depth of penetration of the radiation.

Radiation therapy using heavy charged particles exhibits a clinically advantageous behavior that can be seen in the Bragg peak discussed in section 2.1.3. It is advantageous because the majority of the energy is deposited deep in biological tissue.

The goal of radiation therapy is to apply the dose to the cancerous tissue without affecting the healthy tissue that often surrounds the tumorous area. Since the deposited energy often cannot be injected into the tumor without also being injected into non-tumorous tissue, the goal is to maximize the dose to the tumor while simultaneously minimizing it in the healthy tissue. Furthermore, energy from the initial radiation transfer spreads within the tissue through secondary effects and is strong enough, in and of itself, to cause damage to non-tumorous tissue. Therefore in the field of radiation therapy there are two objectives: to deliver enough directed energy to a tumor site so that the therapy is effective, and to limit the effects of that same therapy on non-tumorous tissue.

The most common types of radiation used in a modern day cancer clinic include high energy X-rays (6MeV and 18 MeV from a clinical linear accelerator), Cobalt-60 gamma rays (1.17 MeV and 1.33 MeV) from a gamma knife, or high energy protons (75-250 MeV produced by a cyclotron or synchrotron) [14] [15]. The particle beams are referred to as ionizing radiation because they are energetic enough to eject one or more orbital electrons and ionize the atoms or molecules it interacts with. The important characteristic of ionizing radiation is the local release of large amounts of energy. When molecules in stable chemical bonds lose one or more of their orbital electrons the bonds become unstable and break. The radiation used in these techniques is strong enough to break strong chemical bonds such as the biologically common C=C bond – a chemical bond held together with 4.9 keV [16] - and can induce biological damage to DNA molecules. The research presented here is motivated by the desire to simulate radiation damage to DNA in biological material. When living organisms are exposed to ionizing radiation (such as X-rays, gamma rays, electrons, protons, and heavy ions), the incident particles can damage the DNA molecule in the cell nucleus causing loss of genetic information that may lead to cell mutation or death [3]. Ionizing radiation may damage DNA either directly or indirectly. In the former situation, the radiation directly interacts with the DNA molecules causing breaks in the helical structure. In the latter case, the radiation interacts with other atoms or molecules surrounding the DNA (e.g., water) to produce free radicals that are able to diffuse far enough to reach and damage the strands of DNA molecules. For example, the following ionization events frequently take place: $H_2O \rightarrow H_2O^+ + e^-$, $H_2O^+ + H_2O \rightarrow H_3O^+ + OH\bullet$. The highly reactive hydroxyl radical OH∙ can then interact with DNA molecules causing strand breaks.

2.1.1 Ionizing Radiation

Radiation is the propagation of energy through vacuum or matter carried along by energetic protons, neutrons or photons (or other heavier particles). The process is energy conserving. In radiation therapy, we are interested in radiation that, upon impact, is energetic enough to eject electrons from stable orbits, creating ionized states of chemical matter. These ionized states lead to chemical changes, and in the case of biological matter such as DNA molecules, these changes can be lethal. Thus we are concerned with quantities such as the instantaneous rate of flow of radiative particles per unit area (flux) and with the total flow of particles crossing a unit area in a given period of time (fluence). These quantities are defined as follows with cross-sectional area, A, through which the particles flow, time, t, and number of radiative particles, N.

$$\phi = \frac{dN}{dA \cdot dt} \qquad (flux)$$

$$\Phi = \frac{dN}{dA} = \int \phi \, dt \qquad (fluence)$$

If each particle has an average energy, E, then the energy flux and energy fluence is defined as

$$\psi = \frac{dE}{dA \cdot dt} \qquad (energy\ flux\ in\ Joules)$$

$$\Psi = \frac{dE}{dA} = \int \psi \, dt \qquad (energy\ fluence\ in\ Watts)$$

In SI units, these quantities are measured in Joules (typically in the MeV range) and in Watts, respectively.

An important quantity measured in radiation therapy is called absorbed dose. It is a measure of energy deposited per unit mass and has units of Grays (J/kg). For a given energy, E, and mass, m, the dose D, is defined as

$$D = \frac{dE}{dm}.$$

Unfortunately, absorbed dose cannot always be directly measured. Alternative measures of dose include effective dose, and equivalent dose that are weighted by either organ volume or known biological effects due to the type of radiation used.

2.1.2 Compton Scattering, the Photoelectric Effect, Pair Production, and Rayleigh Scattering

Electromagnetic radiation such as X-ray radiation is a stream of photons that transfers it's energy to the surrounding tissue producing fast moving charged particles. It is these fast moving charged particles that cause the ionization that damage DNA molecules – not the radiation itself, thus the incident radiation is said to be indirectly ionizing. At energies characteristic of those seen in a Cobalt-60 linear accelerator, the dominant mechanism of energy transfer is called Compton scattering. Compton scattering describes the way energy is transferred to free or loosely bound electrons as the photons collide with them. An incident photon may lose from 0 – 80% of its energy in any given (inelastic) collision, transferring that energy to the electron in the form of kinetic energy. All remaining energy is emitted in the form of a new photon having a longer wavelength that may participate in additional scattering events.

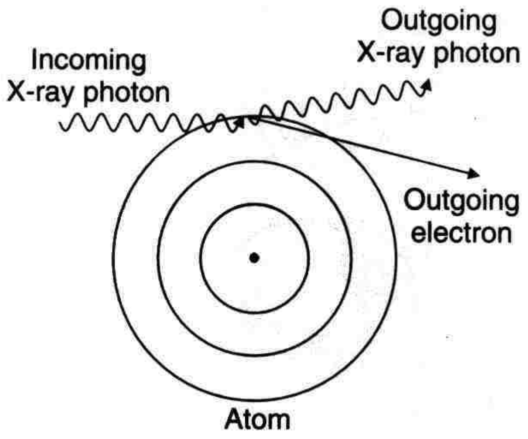


Figure 1. Compton scattering. [67]

The x-ray photon is incident upon a loosely bound or free electron.

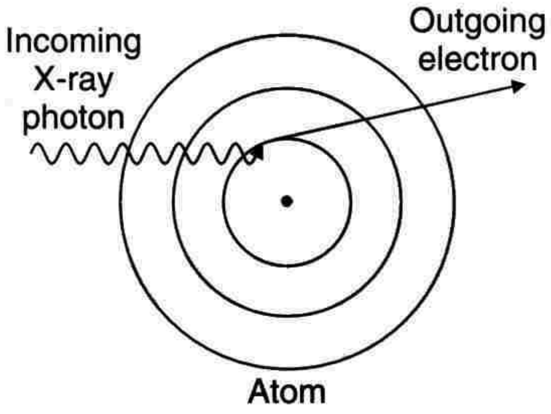


Figure 2. The Photo-electric Effect. [67]

The photon is incident upon a tightly bound electron. In the case shown here, some energy is used to break the orbital bond and the remaining energy is transferred to the electron in the form of kinetic energy.

The photo-electric effect is similar to Compton scattering except that the photon is incident upon a tightly bound electron. In this case, as the electron absorbs energy, it may either change orbital shells or be ejected from the atom completely. If the electron changes shells, a secondary photon is emitted with energy equal to the difference in binding energy between the two orbital shells. If the electron is separated from the atom, some of the energy of the original photon is used to overcome the electron's binding energy. A secondary photon may also be emitted. If one is not emitted, the remaining energy manifests solely as the kinetic energy of the ejected electron. If one is emitted, the energy is split between the velocity of the electron and the wavelength of the new photon.

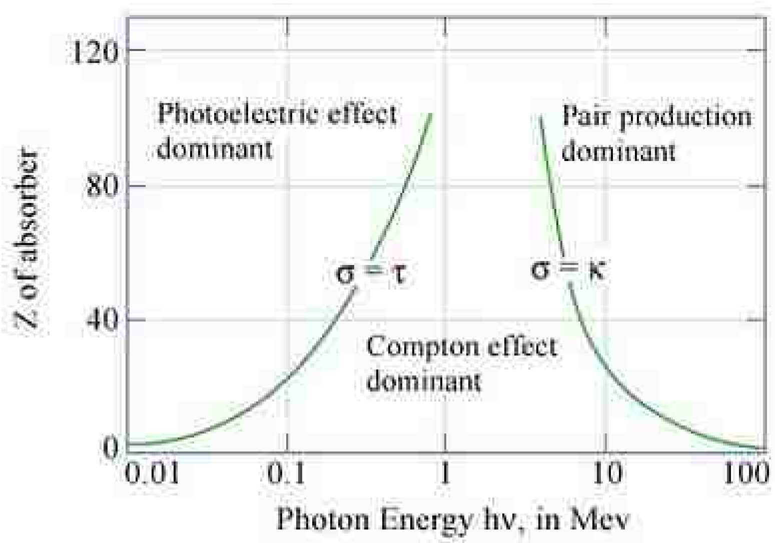


Figure 3. Photon energy vs. target atomic number. [17]

The red line $\sigma = \tau$ indicates the region where the photoelectric effect occurs as often as the Compton effect and the red line $\sigma = \kappa$ indicates the region where pair production occurs as often as the Compton effect.

Both Compton scattering and the photo-electric effect occur during photon-based radiation therapy, but Compton scattering dominates at higher energies, while the photo-electric effect becomes more prominent at lower energies (see Figure 3). The Compton and photo-electric absorption effects in living tissue are not the same. Since Compton scattering is a process dealing with free or loosely bound outer shell electrons while the photo-electric effect occurs with tightly bound inner shell electrons, the absorption of energy by the Compton process is independent of atomic number, but absorption by the photo-electric effect varies roughly as the cube of the atomic number. This means that if a radiative energy that favors energy transfer by the photo-electric effect is used then bone will receive a greater dose than muscle or other soft tissue. On the other hand, if the energy of the beam is increased so that Compton scattering becomes dominant, then the hard and soft tissues receive a more equally distributed dose.

Pair production is the creation of an elementary particle and its anti-particle. Pair production occurs when a photon with energy greater than or equal to the rest mass of the two created particles interacts with the nucleus of an atom. For example, in the creation of an electron and its anti-particle, the positron, the minimum required energy of the incident photon is $1.022\, MeV = 2m_ec^2$, where m_e is the rest mass of the electron and c is the speed of light. The probability of pair production increases with energy and is approximately proportional to the square of the atomic number of the target atom.

Rayleigh scattering also occurs in photon-based radiation therapy. Rayleigh scattering is an interaction between rays of light and polarizable atoms whose size is smaller than the wavelength of the light. This type of scattering is what causes the sky to appear blue in the presence of sunlight. On collision of the photon with the atom, the oscillation of the light ray causes the atom to vibrate at the same frequency as the light. As a result of the oscillation, the atom becomes a radiating dipole, giving off the energy from the incident ray as a new photon. The newly emitted photon has the same wavelength as the original one, but it has a new random direction. This type of scattering is called elastic because no energy is lost in the process. The impact on radiation therapy is that since the ray changes direction, coherency in the overall beam as a whole is lost. In materials with low atomic number such as most soft biological tissues, this type of scattering occurs for photons with energies below 20 keV.

2.1.3 Depth vs. Dose and the Bragg Peak

X-ray or gamma ray radiation are not the only energy delivery mechanisms used in radiation therapy. In 1946, physicist Robert Wilson [18] suggested that heavy charged particles could be used to treat cancer in a way that minimizes damage to healthy tissue. His work was based on the observation that heavy particles such as carbon ions or protons deliver the bulk of their energy deep within the target medium and they release comparatively little energy at the point of entry into the body. This seemingly counterintuitive behavior forms the basis of modern particle therapy. Radiation therapy is called particle therapy when any particle heavier than an electron (i.e., a hadron particle) is used as an energy delivery instrument. This type of therapy is also called hadron therapy. When energetic photons, such as X-rays or gamma rays, are incident upon a material, the majority of the energy is released near the surface. This is because the mechanisms of energy exchange between photons and matter is through direct collision (Compton scattering, the photo-electric effect, pair production and the Rayleigh effect), where much of the energy of the photon is transformed into kinetic energy as it is absorbed into the molecular mass. This is not the case with ionizing energy deposited by neutrons, protons or other ions. This second type of therapy - hadron therapy - releases more energy as the particles slow down and come to rest. The reason for this is because energy is exchanged with the target medium through Coulomb force interactions that cause the formation of ionic bonds. These bonds form more easily as the particles slow down. Charged particles drifting slowly past ions can more easily be trapped in orbit around them, forming a bond, while faster ones can more easily escape the force of electromagnetic attraction between them. By adjusting the input velocity of the hadron particles, the location at which the majority will come to rest can be changed. In this way, the therapeutic dose can be shaped to fit both the size and location of a tumor. Hadron therapy allows for minimal dose applied at the entrance site, no dose applied at the exit site, and maximal dose applied to the tumorous area. The red curve in Figure 4 is an example of this type of dose vs. depth and it is called a Bragg peak. It gets its name from William Bragg who first observed the behavior in 1903. The blue curve is composed of many smaller beams with each beam having a different particle velocity. The sum of the applied doses has a “flat top” near 20 cm showing the shape of the dose as a function of tissue depth. This flat top is called a spread out Bragg peak (SOBP) and forms the cornerstone of radiation particle therapy today. The magenta curve shows an example of a more conventional X-ray or gamma ray dose. Both energy delivery mechanisms, photons and hadrons, are powerful tools at the radiation oncologist’s disposal.

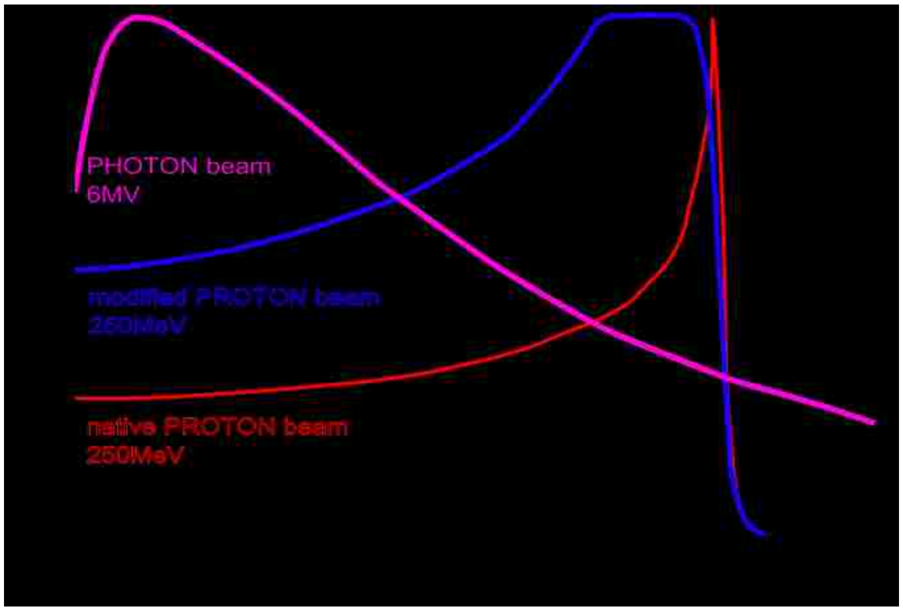


Figure 4. Dose vs. Depth. [19]

The dose vs. tissue depth of a 250 MeV proton beam is shown in red. A beam consisting of the sum of many smaller beams with a total energy of 250 MeV is shown in blue. Note the flat top that is created 20 cm into the body. In magenta, the dose deposition of a 6 MeV photon beam is shown.

One disadvantage of using photons (X-rays or gamma rays) in radiation treatment of cancerous tissue deep within the body that can be seen in Figure 4 is that since most of the energy is released where the beam enters at the surface of the body, many more healthy cells will be killed than cancerous ones. The depth-dose curve, or Bragg curve, (the red curve in Figure 4) is a measure of stopping power described by the Bethe equation

$$-\frac{dE}{dx} = \frac{4\pi}{m_e c^2} \cdot \frac{nz^2}{\beta^2} \cdot \left(\frac{e^2}{4\pi\varepsilon_0}\right)^2 \cdot \left[\ln\left(\frac{2m_e c^2 \beta^2}{I \cdot (1-\beta^2)}\right) - \beta^2\right]$$

where

$$-\frac{dE}{dx} \equiv energy\ lost\ per\ unit\ distance$$

$$m_e \equiv rest\ mass\ of\ the\ electron$$

$$z \equiv particle\ charge$$

$$e \equiv electron\ charge$$

$$n \equiv electron\ number\ density$$

$$I \equiv mean\ excitation\ potential$$

$$\beta \equiv \frac{v}{c}\ for\ particles\ with\ velocity, v$$

$$c \equiv the\ speed\ of\ light$$

$$\varepsilon_0 \equiv vaccuum\ permittivity.$$

The mean excitation potential energy, I, is a function of atomic number and may be obtained by lookup in a table. The electron density, n, is

$$n = \frac{N_A \cdot Z \cdot \rho}{A \cdot M_u}$$

where

$$N_A \equiv Avogadro's\ number$$

$$Z \equiv atomic\ number$$

$$\rho \equiv material\ density$$

$$A \equiv relative\ atomic\ mass$$

$$M_u \equiv molar\ mass\ constant.$$

With stopping power defined as $S(E) = -\,dE/dx$, the mean range of incident particles is obtained by integration and is calculated as

$$\Delta x = \int_0^{E_0} \frac{1}{S(E)}\, dE$$

for particles having initial energy, E_0.

The effect of the Bragg peak used to shape a dose in hadron therapy can be seen in Figure 5 that shows physical dose at the site of a cranial tumor. Note that the beam intensity is significantly greater at the tumor location.

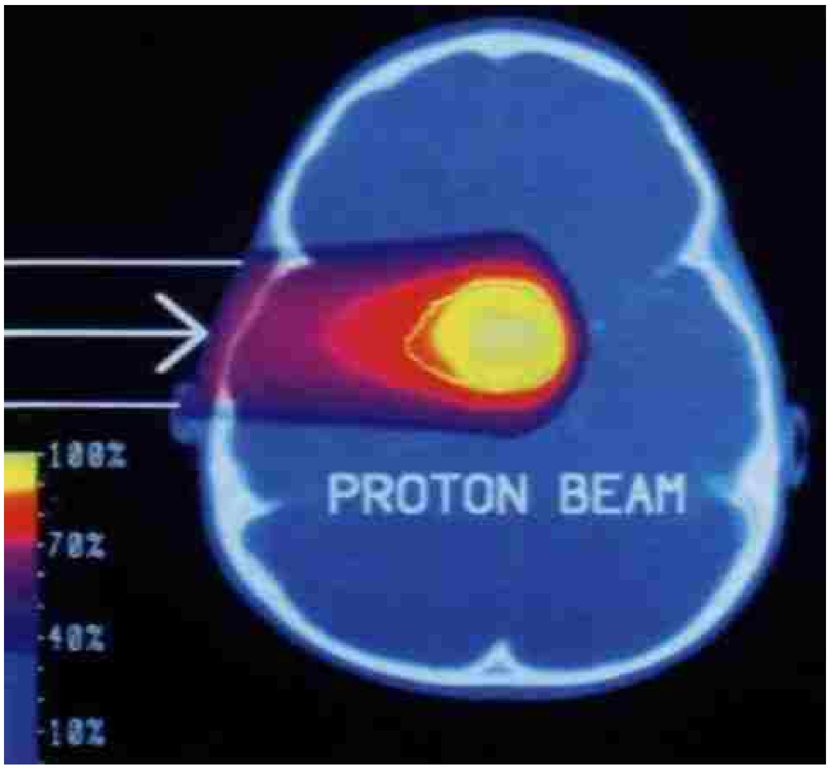


Figure 5. Proton beam dose at a cranial tumor site.

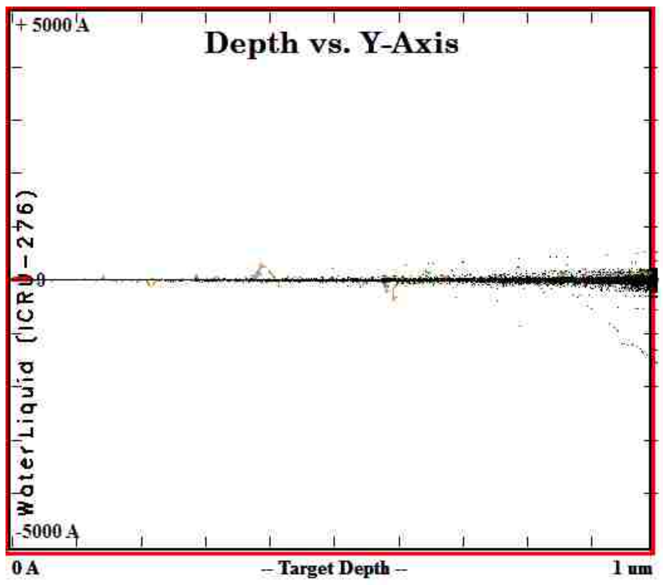


Figure 6. A simulation showing a beam of 7,500 1 MeV Hydrogen atoms shot through water. [20]

Figure 6 shows a beam of hydrogen particles with 1 MeV of energy being shot through $1\mu m$ of water. The results were generated by SRIM [20].

2.1.4 Relative Biological Effectiveness (RBE)

When the effectiveness of a dose of radiation on biological matter is compared to the effectiveness of the same dose of a different type of radiation measured in a laboratory, we use the term relative biological effectiveness (RBE), or equivalently, biological dose. The reference radiation source (often gamma or X-rays) is defined to have an RBE of 1.0. If D_{ref} is the reference dose required to produce some effect, and D_{test} is the dose of another type of radiation required to produce the same effect, then RBE is defined as the ratio of D_{ref} to D_{test} where the two doses may be complex functions of many variables.

$$RBE = \frac{D_{ref}(fluence, flux, tissue\ type, radiation\ type, \ldots)}{D_{test}(fluence, flux, tissue\ type, radiation\ type, \ldots)}$$

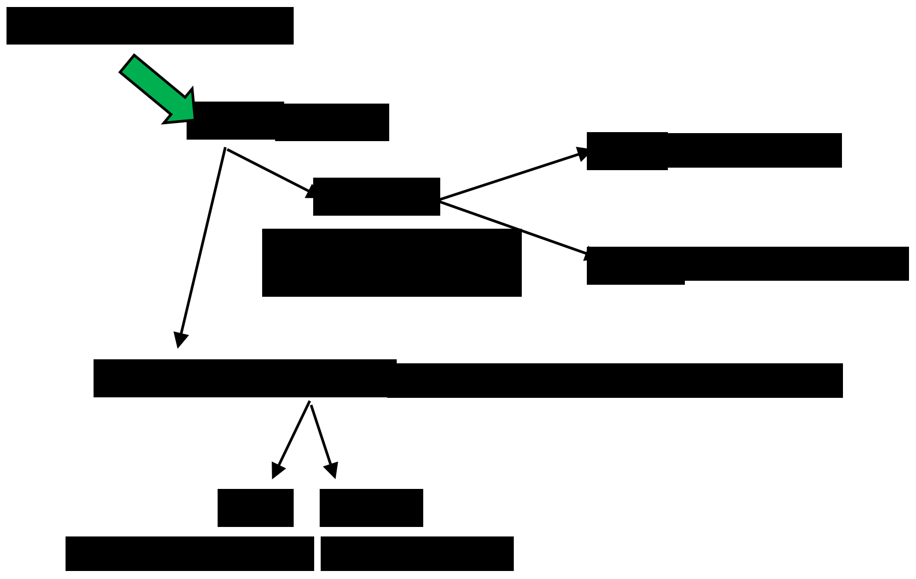
When planning radiation therapy, the type of radiation that will yield the greatest RBE is sought because tumorous cell death can be achieved using the lowest possible dose. Both the reference dose and the test dose are functions of physical dose, dose rate, tissue type, radiation type, and other factors. The RBE provides an estimate of the expected cell failure in a patient given a known cell failure in the laboratory.

2.1.5 Relevant Chemical Reactions and the Creation of Free Radicals

We now discuss the chemical reactions relevant to the simulation being performed here. Let us consider the case when a beam of ionizing radiation is incident on water molecules. Of the many types of molecules in the body, the case of incidence with water is the most important because the body consists mostly of water. For simplicity, all simulations performed here assume the target body consists entirely of water.

When energetic particles interact with water the collision produces a free electron and a positively charged water molecule. Two events follow this. The free electron reacts with another water molecule to produce a negatively charged water molecule, and the positively charged water molecule dissociates into a hydrogen ion and a hydroxyl radical. Finally, the negatively charged water molecule dissociates into a hydrogen radical and a hydroxyl ion. These reactions produce free electrons, H^{-} and OH^{-} ions, and the free radicals, $H\bullet$ and

$OH\bullet$. Free radicals are molecules with an unpaired valence electron. These highly reactive and biologically dangerous molecules will unselectively pair with an electron from another atom or molecule, possibly including those that make up a DNA molecule. Eventually, all of these particles recombine to form water, but some of the free radicals will damage or destroy DNA molecules before they recombine. In total, there are seven particle species excluding water, and ten reactions (see Figure 7, Table 1 and Table 2).



[redacted]

When free radicals encounter DNA molecules they can cause DNA strand breaks. If only one strand is broken, the unbroken strand can often be used as a template to repair the broken side, in which case the breakage is repairable. If both strands are broken, there are three mechanisms of DNA repair that can be employed by the cell called non-homologous end joining, microhomology-mediated end joining, and homologous recombination, but the damage is often unrepairable. In the event that the damage cannot be repaired, the cell will die no later than at the next mitosis cycle when the entire strand of DNA is copied as the one cell attempts to split into two. The goal of radiation therapy is to kill cancerous cells by causing these double strand breaks.

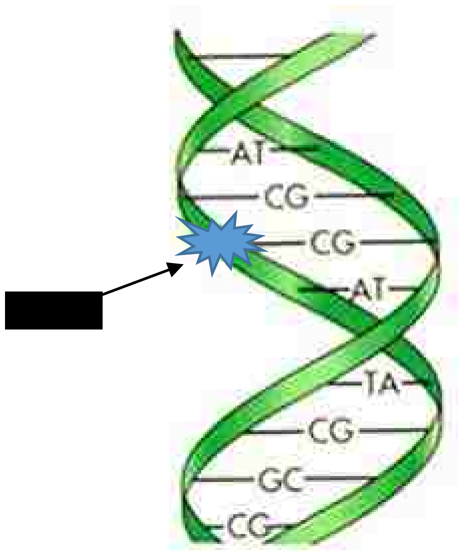


Figure 8. DNA molecule damaged by a free radical.

2.3 Three Stages of Activity Following Irradiation

The majority of the damage caused by radiation is due to chemical reactions with water within cells. The reactions occur rapidly and are non-selective and random. When ionizing radiation enters living organisms, it follows three consecutive stages. The first stage following cellular irradiation is called the *physical stage*. In this stage the molecules in the cell undergo elastic and inelastic physical interactions. The physical stage is characterized by energy deposition and is extremely fast, lasting only about 1 femtosecond (10^{-15} second).

The second stage is called the *physico-chemical stage* and lasts roughly from 1 femtosecond to 1 picosecond (10^{-12} second). This stage is characterized by thermalization and solvation of subexitation electrons, electronic hole migration, and electronic recombination. Thus, particles intermix, seeking thermal equilibrium, while excited electrons seek to give off their excess energy as photons or vibrational energy (heat) and drop back down to their valence shells. The result of the physico-chemical stage is an array of radiolytic species that are thermalized and subject to the diffusion equation.

The third stage is called the *chemical stage*, and lasts from approximately 1 picosecond to 1 microsecond (10^{-6} second). It involves radiolysis of the cellular fluids and the creation of radioactive molecular and atomic particles. In this stage, Brownian motion governs particle movements, where the thermalized radiolytic species diffuse in the medium reacting with each other and with biomolecules such as DNA. The excitations and ionizations created by the energy transfer and ensuing chemical reactions lead to the creation of free radicals - molecules or atoms that have at least one unpaired electron in an orbital shell. The Debye-Smoluchowski (DS) equation describes the movements of the free radicals as

$$\frac{\partial p(r,t|r_0)}{\partial t} = D\nabla^2 p(r,t|r_0) + D\beta\nabla \cdot p(r,t|r_0)F(r)$$

where $F(r)$ is an external force field, $\beta = \frac{1}{k_B T}$, k_B is the Boltzmann constant, T is the temperature and $p(r,t|r_0)$ is the probability density of a particle being at location r at time t given the initial location r_0. This is an application of the diffusion equation and in the absence of an externally applied force the solution to it is a Gaussian distribution with

$$p(r,t|r_0) = \frac{1}{(4\pi Dt)^{1/2}} e^{-\frac{(r-r_0)^2}{4Dt}} \qquad 1.$$

Where D is the diffusion coefficient, t is time and p is the probability that a particle initially located at r_0 will be located a distance $r - r_0$ away after time t [21]. Table 1 shows the experimentally derived diffusion coefficients and reaction radii of some important free radicals.

Table 2 shows some of the more important chemical reactions that can take place between free radicals.

SPECIES	**DIFFUSION COEFFICIENT, D ($10^{-9}M^2S^{-1}$)**
e_{aq}^-	4.9
$\bullet OH$	2.8
$H\bullet$	7.0
H_30^+	9.0
H_2	4.8
OH^-	5.0
H_20_2	2.3

Table 1. Diffusion coefficients of free radicals [22]

REACTION	REACTION RATE (10^{10} $M^{-1}S^{-1}$)
$H\bullet + e^-_{aq} + H_2O \rightarrow OH^- + H_2$	2.65
$H\bullet + \bullet OH \rightarrow H_2 0$	1.44
$H\bullet + H\bullet \rightarrow H_2$	1.20
$H_2 + \bullet OH \rightarrow H\bullet + H_2 0$	$4.17\text{x}10^{-3}$
$H_2 0_2 + e^-_{aq} \rightarrow OH^- + \bullet OH$	1.41
$H_3 0^+ + e^-_{aq} \rightarrow H\bullet + H_2 0$	2.11
$H_3 0^+ + OH^- \rightarrow 2H_2 0$	14.3
$\bullet OH + e^-_{aq} \rightarrow OH^-$	2.95
$\bullet OH + \bullet OH \rightarrow H_2 0_2$	0.44
$e^-_{aq} + e^-_{aq} + 2H_2 0 \rightarrow 2OH^- + H_2$	0.50

Table 2. Diffusion controlled reaction rates between free radicals [22]

2.4 Monte Carlo Simulations

The term, “Monte Carlo method”, is a term that refers to the use of repeated random sampling and probability statistics to solve problems that either don’t have closed form solutions or for other reasons are difficult to solve. The method has a wide range of applications and is used to solve problems in fields ranging from economics to nuclear physics. One of the earliest and most famous implementations of the method was presented by Nicholas Metropolis [23] in his 1953 study of the ideal gas law, $PV = nRT$, where the Boltzman distribution, $e^{-\mathcal{E}/kT}$, was repeatedly used to move an initial distribution of 224 particles (a simulated gas) into a new distribution. After many particle movements, the system, as a whole, seeks a state of low energy, in agreement with theoretical predictions.

One of the simplest illustrations of the method can be seen by observing how it is used to compute a numerical solution to an integral between two limits of integration. The technique consists of the repeated random sampling of points in the domain and range of the function. The area under the curve is the solution to the problem and that area is estimated to be the ratio of sampled points under the curve to the total number of sampled points multiplied by the area of the enclosing box (see Figure 9).

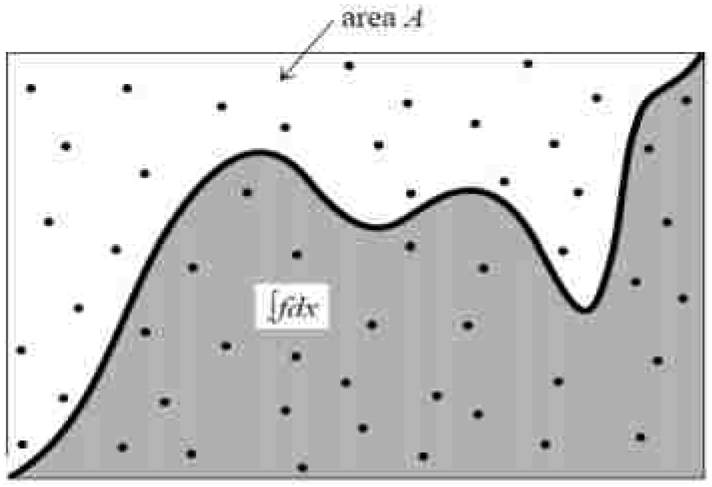


Figure 9. Using the Monte Carlo method to solve an integral.

The accuracy of the method improves as more and more random samples are generated and there is no restriction regarding what type of integral can be solved. Thus, this is a simple and powerful method, but obtaining a high degree of accuracy with it can come at the cost of significant computer time and resources.

2.4.1 Kinetic Monte Carlo Simulations and Diffusion

Monte Carlo simulations calibrated with experimental measurements provide the most accurate means of simulating the physical and physico-chemical stages [24] of the problem of interest here. We are interested in the simulation of the computationally challenging third stage, i.e., the chemical stage where indirect damage to DNA molecules delivered by free radicals forms the basis of study. Because the simulations evolve through time, our Monte Carlo simulations are referred to as kinetic Monte Carlo simulations (also known as a Gillespie algorithm [25]). There are tools in use today that have kinetic Monte Carlo dose calculation engines and perform this kind of computation such as the Geant4-DNA Toolkit [4] [5] [6] [7], PARTRAC [26] [27], and RADACK [28] and all of them require significant computer time and resources to solve reasonably sized problems of this type.

Accurate modeling of the chemical stage requires tracking the motions of millions or even billions of particles over a million picoseconds (the chemical stage lasts approximately 1 μs), but simulations of this size are prohibitively expensive to run even on large computer clusters. To put the scale of these simulations in perspective, with a mere quarter million incident electrons at 1MeV, there are more than 12 million •OH free radicals generated within the first picosecond [4] [29] [30] and more than a billion primary ions. The time scale under consideration is 1 ps to 1 μs where a time step on the order of 1 ps is normally used [31]. Thus the scale of a full fidelity simulation is enormous, and can easily overwhelm a state-of-the-art computing cluster. Clearly, any algorithmic improvements that can be made to this process will be greatly beneficial.

For these simulations, the diffusion time is proportional to the square of the diffusion distance and since no other forces are present, the positions of the molecules are governed solely by Brownian motion (i.e. a random walk). The following equation relates the mean squared displacement as a function of diffusivity, D, and tine, t. It is the second moment (i.e., variance) of equation 1 and it relates the average path length of a diffusing particle to the time it is in motion.

$$\overline{x^2} = 2Dt$$

These chemical reactions are *diffusion limited* because the speed of the reaction is fast compared to the movements of the reacting molecules. Reaction times are on the order of fractions of a picosecond. In this scenario a kinetic Monte Carlo simulation is a direct implementation of a random walk of a large number of simulated particles over time in software. We are interested in large scale Monte Carlo simulations where ionized molecules can be numbered in the hundreds of thousands or millions. A brute force approach to solving this problem compares all pairs of particles at each time step requiring $O(m \cdot n^2)$ run time, for n particles and m discrete time steps.

The recent and open source Geant4-DNA Toolkit [4] [5] [6] [7] uses a hierarchical data structure called a kd- tree [32] to provide fast nearest neighbor searches. In this approach time is discretized, and all reactions that occur take place at the beginning of the time interval. Within each time interval, the particles diffuse – that is, they make random movements. To determine which pair of particles react, a kd-tree is built based on the positions of the particles at the beginning of each time step. A closest pair is found whose reaction time is used to determine the next time interval. The main drawback to this algorithm is that the tree has to be rebuilt for each iteration because all particles are in new locations. With m time steps, this is an $O\big(mn\, log(n)\big)$ algorithm, where $O(n\log n)$ is the time to build the tree and the expected time to find the closest pair in each iteration. If the theoretically best time bounds of searches in the kd-tree are to be realized, then the tree must be balanced after the particles are inserted. This expensive step diminishes the benefit of the fast search. Karamitros [5] developed an improvement to this method that makes use of dynamically sized time steps defined to be the length of time required for the current nearest neighbors to react with some probability, but the use of the kd-tree itself remains a bottleneck.

2.5 Geant4-DNA

Geant4-DNA is an open source, extensible toolkit that simulates the passage of particles through matter. Its use spans multiple application domains such as high energy physics, astrophysics, space science, medical physics, and radiation protection. It is a Monte Carlo system that combines the general purpose Geant4 architectural design with functionality specific to needs in the field of radiobiology. This tool was developed to meet the same goals outlined in this dissertation. It can be used to estimate the rate of cell survival following irradiation by a high energy source. A plot of dose vs. survival rate as output from Geant4-DNA is shown in Figure 10. Geant4-DNA is a state-of-the-art tool used for calculating radiometric dose.

The Monte Carlo engine in the Geant4-DNA toolkit simulates the movements and interactions of the incident high energy particles used when delivering therapeutic doses into cancerous tissue. A kd-tree is used to provide fast nearest neighbor searches in order to determine when the next reaction occurs. Some inefficiencies associated with this approach will be discussed in section 2.4.1. It is hoped that the proximity grid method presented in chapter 3 can eventually be integrated into Geant4-DNA.

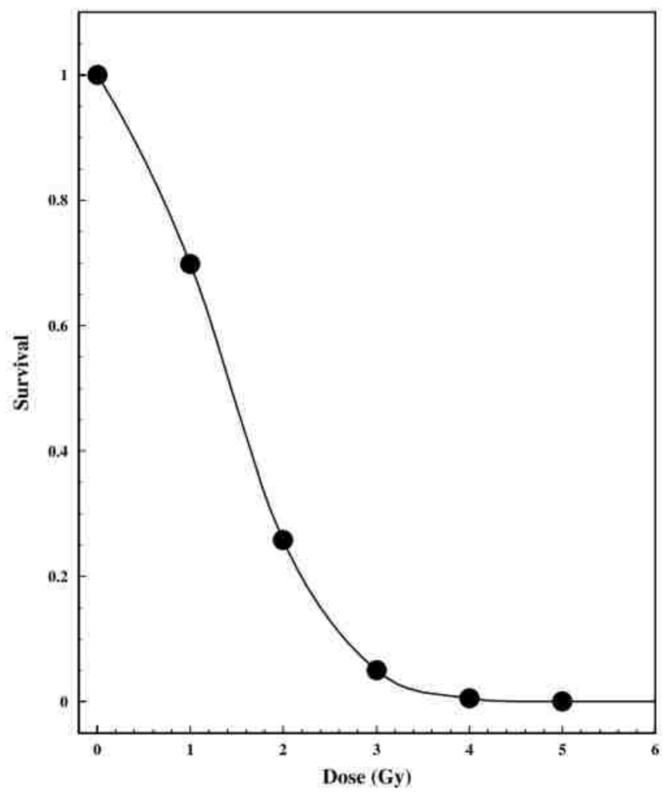


Figure 10. Geant4-DNA fractional survival.

Fractional survival rate of a population of hamster cells irradiated with a 3.66 MeV proton beam vs. dose absorbed [33] as reported by Geant4-DNA.

2.6 Fick's Second Law of Diffusion

Although we directly simulate the motions of molecules, it can be useful to derive an equation describing the macroscopic behavior of a diffusing system. One attempt at parallelization of the proximity grid method made use of the equations presented below as a way to transfer moving particles between processors in charge of different regions of space, but the method was unsuccessful. The derivation is included here for reference. Adolf Fick's second law of diffusion [34] [35] describes the change in particle concentration in one dimension over time where particles are under the influence of diffusion and there are no externally applied forces. The equation is

$$\frac{\partial c(x,t)}{\partial t} = D\frac{\partial^2 c(x,t)}{\partial x^2}$$

where D is the diffusion coefficient (unit area per second), $c(x,t)$ is the concentration (number of particles per unit volume) at location x and time t, and x points in the direction of the change in particle concentration. It predicts how the particle concentration changes with respect to time due to diffusion. The general solution [35] can be obtained by applying the similarity method [35] [36]. After deriving the general solution we will apply boundary conditions corresponding to the diffusion we would expect to see when a drop of dopant is dropped into a bath of liquid. In this way we will arrive at a useful equation for our purposes. The method starts by guessing at a prototype solution with the aid of dimensional analysis. The expected functional form is

$$c(x,t) = t^{-\beta}F(\eta)$$

$$\eta = x^2/4Dt.$$

This form has a "size" term, $t^{-\beta}$, and a "stretch" term, $F(\eta)$, where F is an unknown function, β is an unknown constant, and t is time. The exponent 2 in the term x^2 is an educated guess that arises because x appears in Fick's Law as a second order derivative. The Dt in the denominator serves to make the term, η, unitless, allowing the function F to behave as a universal stretch factor that is insensitive to any physical quantity (as it should be). The factor of four appears for mathematical convenience. The $t^{-\beta}$ term represents the temporal decay of the maximum concentration, with the unitless constant β expected to be positive. The first and second derivatives of $c(x,t)$ with respect to t and x are

$$\frac{\partial c(x,t)}{\partial t} = -\beta t^{-\beta-1}\,F(\eta) + t^{-\beta}\frac{F}{d\eta}\frac{d\eta}{dt} = -\beta t^{-\beta-1}\,F(\eta) - \eta t^{-\beta-1}\frac{dF}{d\eta}$$

$$\frac{\partial c(x,t)}{\partial x} = t^{-\beta}\frac{dF}{d\eta}\frac{d\eta}{dx} = \frac{\beta x t^{-\beta-1}}{2D}\frac{dF}{d\eta}$$

$$\frac{\partial^2 c(x,t)}{\partial x^2} = \frac{t^{-\beta-1}}{2D}\frac{dF}{d\eta} + \frac{x t^{-\beta-1}}{2D}\frac{dF^2}{d\eta^2}\frac{d\eta}{dx} = \frac{t^{-\beta-1}}{2D}\frac{dF}{d\eta} + \frac{t^{-\beta-1}}{D}\eta\frac{dF^2}{d\eta^2}\,.$$

Substitution into Fick's Law yields

$$-\beta t^{-\beta-1}\,F(\eta) - \eta t^{-\beta-1}\frac{dF}{d\eta} = \frac{t^{-\beta-1}}{2}\frac{dF}{d\eta} + \eta t^{-\beta-1}\frac{dF^2}{d\eta^2}.$$

Note how both t and D cancel out due to the careful way that η is defined. Continuing to simplify:

$$-\beta\,F(\eta) - \eta\frac{dF}{d\eta} = \frac{1}{2}\frac{dF}{d\eta} + \eta\frac{dF^2}{d\eta^2}$$

$$\beta\,F(\eta) + \eta\frac{dF}{d\eta} + \frac{1}{2}\frac{dF}{d\eta} + \eta\frac{dF^2}{d\eta^2} = 0$$

$$\eta\frac{d}{d\eta}\left(\frac{dF}{d\eta} + F\right) + \frac{1}{2}\left(\frac{dF}{d\eta} + 2\beta\,F\right) = 0.$$

Since β is a free parameter, we choose it to be $\frac{1}{2}$ so that the terms in parentheses are identical. Then

$$\eta\frac{d}{d\eta}\left(\frac{dF}{d\eta} + F\right) + \frac{1}{2}\left(\frac{dF}{d\eta} + F\right) = 0.$$

A solution to this equation satisfies the condition

$$\frac{dF}{d\eta} + F(\eta) = 0.$$

That is, our solution is conditioned on the fact that the derivative of a function plus the function evaluated at any point is always zero. This is a condition that can only be satisfied with an exponential function, thus

$$F(\eta) = \alpha e^{-\eta}$$

and the general form for the concentration is

$$c(x,t) = \frac{\alpha}{\sqrt{t}} e^{-x^2/4Dt}$$

where α is an unknown constant. Applying the boundary condition that the original number of particles is constant (i.e., there is no particle source or sink), then the integration of the concentration over space at any given time must sum to the number of particles.

$$\int_{-\infty}^{\infty} c(x,t)dx = N$$

$$\int_{-\infty}^{\infty} \frac{\alpha}{\sqrt{t}} e^{-x^2/4Dt} dx = N$$

Defining $y \equiv \frac{x}{2\sqrt{Dt}}$ allows us to rewrite this as

$$2\alpha\sqrt{D} \int_{-\infty}^{\infty} e^{-y^2} dy = N.$$

Due to symmetry this is

$$4\alpha\sqrt{D} \int_{0}^{\infty} e^{-y^2} dy = N.$$

Since $\int_0^\infty e^{-y^2} dy = \frac{\sqrt{\pi}}{2}$ this reduces to

$$2\alpha\sqrt{\pi D} = N$$

$$\alpha = \frac{N}{2\sqrt{\pi D}}$$

and the concentration as a function of time and location in one dimension becomes

$$c(x,t) = \frac{N}{2\sqrt{\pi D t}} e^{-x^2/4Dt}.$$

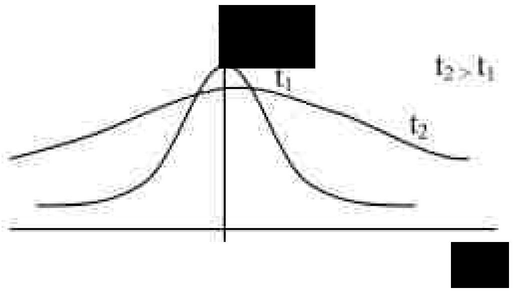

[redacted]
[redacted]

Figure 11. The concentration profile at two times, t_1 and t_2 with $t_2 >, t_1$.

CHAPTER 3

SEQUENTIAL ALGORITHM

The problem we wish to solve is this. Given n radiolytic particles in a variety of molecular species in a bath of H_2O molecules, arranged as a particle beam (e.g., from a gamma knife) and subject to diffusion over a time period of $1\,\mu s$, observe and process all chemical reactions, reporting the radiochemical yield over time. We observe that the reactions in

Table 2 contain only two species on the left hand side, excluding water which is treated as being infinitely abundant and existing everywhere, and that the closest pair of particles in any of the left hand sides of the equations will be the next reaction to process at any instant. Thus, this is an instance of the kinetic polychromatic closest pair problem in computational geometry. Stated formally, the polychromatic closest pair problem is: given n points in a metric space, where each point is one of m species (or colors), find all pairs of closest particles such that the resulting set of pairs is the set of ordered pairs of the elements of m. We use the term "kinetic" because the particles are in motion and we want to ask the closest pair question at each time step during the simulation.

The proximity grid method presented here solves the closest pair problem when particles are in motion, and is based on the following five observations. 1) A grid provides a notion of locality so that entire groupings of particles can be pruned from consideration at once. 2) A grid is efficient in the kinetic case. 3) A kinetic data structure provides a time stepping mechanism that is efficient, requiring the execution of only spatially local operations at each time step. 4) A kinetic data structure does not skip important events the way a fixed time stepping mechanism can. 5) A hash table provides constant time lookup and naturally lends itself to sparse data storage of the contents of the grid.

3.1 Overview

A serial version of the algorithm discussed in the sections that follow is presented here. An OpenMP parallelized version is presented in chapter 4. The serial version achieves $O\big((n+k)\log n\big)$ run time where there are n input particles and the priority queue is serviced k times. The serial version of the code was published in [22], and the parallel version is being prepared for publication at this time.

Section 3.2 discusses the closest pair problem and how it is used to implement a nearest neighbor search in order to identify chemical reactions. Section 3.3 gives the details of the implementation of the time stepping mechanism known as a kinetic data structure. The kinetic data structure is used to efficiently handle the large number of time steps required in the simulation. Finally, section 3.4 outlines the overall program flow.

3.2 Nearest Neighbor Search

3.2.1 The Closest Pair Problem

The closest pair problem is a problem in computational geometry and can be stated as: given n points in a metric space, find the pair of points with the smallest distance between them. The bi-chromatic version of the problem assigns one of two colors to each point and requires that the solution contain points of a different color. The brute force solution to both forms of the problem, obtained by exhaustively comparing the distances between each pair of points, runs in $O(n^2)$ time. This is an important problem to consider for these simulations because knowing where the closest pair of particles is means knowing the participants of the next radiochemical reaction.

The problem of finding the closest pair of points in an array of randomly scattered two dimensional points has been solved in $O(n)$ time by Khuller [37] using what is referred to as a randomized sieve. In this case, the points are motionless. The algorithm proceeds by choosing a point at random and calculating its distance to all other points. Once a lower bound distance, δ, is found, a two dimensional grid is overlaid onto the domain of particles where the cell size is $\delta/3$. All particles that are alone in their grid squares with no particles in an adjacent square (i.e., alone in their neighborhood) are removed from the problem (they fall through the "sieve"). These steps are repeated with the smaller set of particles until it is determined that the next iteration would remove all particles. Finally, at this last iteration, labelled iteration i, a grid of size δ_i is overlaid on the final set of particles and distances are computed between all pairs of particles that are within the same neighborhood (i.e., in the same grid square or an adjacent one). The pair with the minimum distance is the closest pair. Khuller's solution is the best known solution when the points are not in motion and his solution is adaptable to three dimensions, but we wish to consider the problem of finding closest pairs of various particle species that are in motion. If Khuller's static problem is changed so that the points undergo simulated Brownian motion (diffusion) and if the closest pair question is asked at every time step of the simulation, then this solution becomes inefficient because all points removed from the problem at one time step must be re-inserted at the next one.

Nonetheless, it is instructive to notice that the notion of locality provided by the grid is useful in reducing the search space of the problem. An efficient method for solving this problem in the kinetic case is developed here.

This algorithm solves the dynamic bi-chromatic closest pair problem using a layered directed acyclic graph, where each layer is a hash table to store the particles (at the lowest layer) and cells of particles (at all other layers). The use of hash tables allows the identification of the closest pair in linear expected time while also saving space since empty cells of particles need not be stored. Time is not discretized globally throughout the simulation. Rather, a priority queue is used to keep track of time in spatially local segments and its use requires only spatially local work to maintain. With these techniques, the simulation time is reduced to $O((n+k)\log n)$, where $O(n\log n)$ is the time to initialize the data structure, k is the total number of events of interest (i.e., Brownian movements), and $O(\mathit{log}\,\mathrm{n})$ is the time to update the priority queue in each iteration. Implementation and experiments have shown that with the new algorithm the actual run time is nearly linear in the number of input particles and is considerably faster than the current hierarchical approach. More will be said about this algorithm later.

3.2.2 Proximity Grid

A hierarchical grid of grids (the proximity grid) provides a way to solve the closest pair problem where distinct definitions of closeness apply per {reaction, species} pair (each reaction has a different reaction radius) while at the same time satisfying the condition that each particle is represented exactly once in the simulation. This condition is necessary because to relax it would mean that insertions, deletions, or movements of particles would need to be done more than once and would constitute an undesirable performance penalty. The proximity grid may be thought of as a layered directed, acyclic graph, where each layer is a three dimensional grid implemented as a hashed map. Any reaction serves as an entry point (a point at which one may begin iterating over particles), or root node, of the graph. Reaction nodes have one outgoing edge per reactant that points to the grid with an edge length equal to the reaction radius of the associated reaction for that reactant (see Figure 12). There is only one copy of any given particle in the entire simulation, yet that particle is represented at each given reaction radius – a key feature of the method (see Figure 13). It is important to represent the particles at all reaction radii of the reactions they can participate in for two reasons. First, if the cells in the grid have a size equal to a reaction radius, then particles more than two cells apart cannot participate in a reaction and may easily, and inexpensively, be pruned from consideration when checking for reactions. This means that it is only necessary to compute a distance between pairs of particles in adjacent cells. Second, it is important to keep the list of candidate reactants as small as possible so that the number of potential reactions to consider is minimized. The reaction radii considered here vary by as many as three orders of magnitude, so failing to meet this criteria would have a significant impact on run time.

The representation of particles at multiple radii is accomplished by building a tree of cells (see Figure 14). Particles are directly inserted only in the grid with the smallest reaction radius for any given particle species (the leaf nodes of the tree). All other grids containing that particle contain particle cells rather than directly containing particles (the branch nodes of the tree). Iteration through the particles for any cell proceeds by recursively iterating through child cells in depth-first search fashion, and each grid is sparse. Sparsity means that if any gridded region contains no particles, then it also contains no cells (that is, the depth-first search for particles always visits particles, never an empty cell). With this hierarchy of cells, iteration through the particles in any given cell is linear in time regardless of which grid iteration starts in.

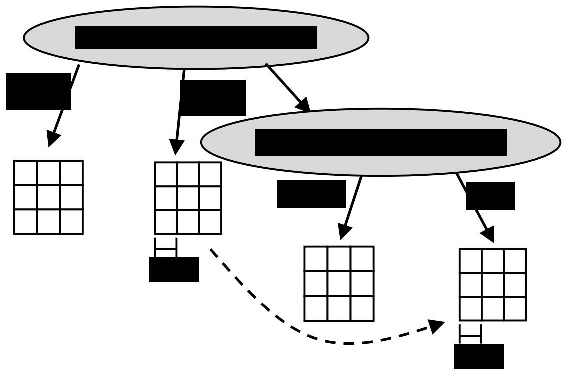


Figure 12. Reaction/reactant graph.

The reaction graph and child reactant grids with the dotted arrow showing the tree of grids for particle "e". r_{e1} is the reaction radius of the $H\bullet + e + H_2O \rightarrow OH + H_2$ reaction, and r_{e2} is the reaction radius of the $H_2O_2 + e \rightarrow OH + \bullet OH$ reaction.

The nearest reactable neighbor for any particle is found by searching through the cell that the particle belongs in and in all twenty-six surrounding cells in 3D space. Iteration begins in the grid corresponding to the reaction radius of interest.

For any particle of interest, any other particle in the enclosing 3×3×3 grid cube region is said to be in proximity to the particle. All particles in proximity to a particle of interest are checked for a reaction.

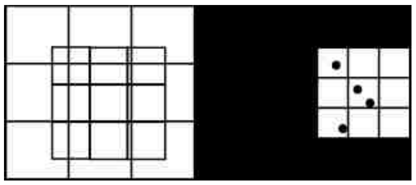


Figure 13. Grid of cells.

Grid of cells for particle, "e", in the first and second reactions from Figure 12. The cell size is the reaction radius for each reaction. The smallest "e" cell contains particles of species "e". All other "e" cells contain "e" grid cells of a smaller reaction radius.

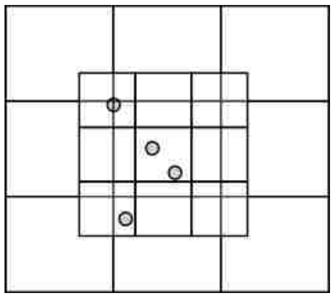


Figure 14. Hierarchy of cells.

There is one such hierarchy per species of particle. There is one level of hierarchy per reaction radius that the particle is associated with.

3.3 Simulation of Time

3.3.1 Kinetic Data Structures

Kinetic data structures [10] [11] [12] [13] are designed to keep track of certain discrete events of a system of continuously moving objects. For example, in our simulation we are interested in tracking reactions (i.e., discrete events) for a set of moving particles. It is customary to simulate such events within the context of a discrete time stepping environment where all particles move in lock-step with one another at every time step, however this type of simulation suffers from the problem that excessively small time steps capture all important events but can take too much computer time to simulate. On the other hand, excessively large steps can miss important events. It is often difficult to find a good compromise between these two extremes especially in light of the fact that the most ideal time step size may not stay constant throughout the simulation. Furthermore, the chemical reactions of interest to us occur on the scale of fractions of a picosecond and one million picoseconds of time (one microsecond) must be simulated. Due to the large number of time steps, it is important to not waste cpu time simulating unnecessary events. A kinetic data structure provides an alternative to a constant time stepping mechanism in a spatially global environment, and when well designed, it can simulate exactly the right number of time steps to capture all important events without wasting simulation time or requiring the determination of an appropriate step size. In addition, each step taken requires only spatially local operations.

The key idea is to identify a small set of discrete attributes that uniquely characterizes a useful event within the simulation. The system then evolves with time, while maintaining the set of discrete attributes which are referred to as a certificate. The times at which the certificates become invalid, arranged in sorted order, form the basis of the time stepping mechanism in a priority queue. Thus, time steps in the simulation do not proceed in fixed increments. They jump forward to the next time at which a certificate loses validity. When a certificate becomes invalid, a small amount of spatially local work is done to re-establish validity and the simulation continues. The work is spatially local because certificates are usually defined in terms of spatially local configurations of geometric objects. A certificate might assert that a given set of four points form a convex hull, for example. When one point moves such that it crosses the hull boundary, the certificate fails. The work required to reestablish validity is simply to redefine the hull in terms of a new set of points - all spatially local operations.

Since the inception of the idea, many classic algorithms have been kinetized, including the closest pair problem in the plane [12]. This is an approach that makes use of three $60°$ right-facing wedges for every point in the domain whose apexes are located at each point (see Figure 15). Each of the three wedges has a left and a right side (both $\pm 30°$ slices) that function as a binary tree of neighboring points such that neighbors on the left $30°$ side are closer to the point at the apex than neighbors on the right $30°$ side. There are three binary trees connected to each point (three wedges) covering a total arc length of $180°$. This coverage allows all points on the right to be considered as "children" of points on the left in a system of binary trees. Consider the placement of point c in the 3-wedge structure anchored at point a in Figure 15. If it were not for the presence of point b, point c would have been placed in the left side of the center wedge (center binary tree) of point a. However, point b is closer to a than c is to a. Therefore point c is placed in point b's lower wedge (lower binary tree) instead. In this way, chains of near neighbors are maintained and the search for a nearest neighbor can be performed in $O(n \log n)$ time.

The flight paths of each point are known and therefore the times at which the arrangement of the points in the tree structure become invalid can be computed. These times are called certificate failures and are inserted into a priority queue of failure times. Time stepping proceeds by processing each failure time, in the order defined by the queue, repairing the trees and inserting new certificates into the queue. These repairs are spatially local operations, requiring only a few tree insertions, deletions, and rotations, per time step which makes the algorithm both efficient and amenable to situations where objects are in motion.

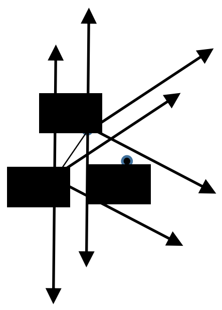


Figure 15. The binary tree structure of near neighbors.

Each node contains three binary sub-trees consisting of three 60° *wedges. The top of the tree is the left-most node. The leaves are on the right.*

3.3.2 Algorithmic Use of the Kinetic Data Structure

For the simulations performed in this dissertation, the motions of particles are random walks governed by a Gaussian distribution from the solution to the diffusion equation. The idea of using a time based priority queue [38] to avoid global time discretization has inspired the development of the following randomized kinetic closest pair algorithm from which we make the following observations.

Observation 1: The randomized closest pair algorithm as discussed earlier in the closest pair section and in [37] relies on the validity of the underlying grid. As long as the moving particles do not move to a different cell, the grid is in a “valid” configuration. This suggests that the certificates should be defined simply as the movement of a particle from one cell to another. Since the particle is subject to Brownian motion, we can use random sampling to calculate the time a particle changes its grid cell. This is a certificate failure and the time of failure can be put in the priority queue.

Observation 2: We are interested in reaction events. Therefore, the reaction radius of a given reaction will determine the size of the associated grid. For any given particle, only the particles in the enclosing 3x3x3 set of cells can react with it. We can therefore locate its nearest neighbor in constant time, and use random sampling to determine the reaction time.

Since the number of events in our priority queue is n, it takes O(logn) time to insert or remove an event from the queue. Our kinetic randomized closest pair algorithm is responsive, local, compact, and efficient as defined by Basch in [12]. These terms mean that the data structure is no more than polylogarithmic in the number of input particles, both in required memory and in run time. For this priority queue, a certificate expires when a particle moves from one cell (of the smallest reaction radius for the associated particle type) to another.

The expected simulation time of our kinetic randomized closest pair is O((n+k)logn), where O(nlogn) is the time to initialize the data structure, k is the total number of events of interest, and O(logn) is the time required to update the priority queue in each iteration.

3.4 Brownian Bridge

In time-based simulations, it is most common to discretize time so that the state of the simulation moves forward with discrete jumps in time. Thus one knows the locations of particles only at those times that are simulated. For a diffusive process, the underlying physics are those of Brownian motion, thus, the unknown inter-particle distances can be smaller between time steps than they are at the time step boundaries where the distances are known. The discrete nature of the simulation effectively imposes a limit on our knowledge of the locations of the particles. This fact is important because radiochemical reactions occur when species are close enough to react. The possibility that two particles are close enough to react between time steps, but not close enough at the time step boundaries can be handled probabilistically.

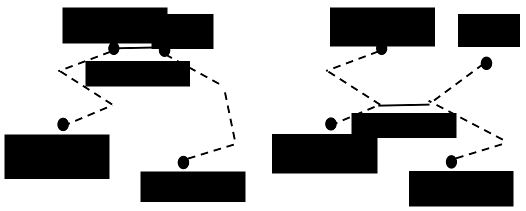


Figure 16. Two particles in Brownian motion.

The particles travel between two known points in a specified time with reaction radius, r. Particles p_0 and $p_1$react because $d_1 \leq r$ (left). Particles p_0 and $p_1$react because $d \leq r$ during flight (right).

Let the locations of particles at time steps t and $t + \Delta t$ be known, but the locations at times $t + \alpha\Delta t$ where $0 < \alpha < 1$ be unknown. The question of whether or not these particles are close enough to react during that time can be answered probabilistically using a construction called a Brownian bridge [39] [5] [40] [41]. A Brownian bridge is a stochastic process that is continuous in time between two endpoints, t and $t + \Delta t$. It is a sequence of random steps that are jointly normally distributed such that the endpoints of the random walk are known but the steps in between are not. A random walker has a known position at t and $t + \Delta t$ and the positions between are not known, but each step the walker takes is the result of a normalized Gaussian sample in the domain. That is, the probability density of any given step is the normal distribution:

$$p(\vec{x}) = \mathcal{N}(\vec{\mu}, \sigma^2) = \mathcal{N}(\vec{\mu}, Dt)$$

$$\mu = \frac{\overrightarrow{x_0} + \overrightarrow{x_1}}{2}$$

where we note that the variance may be thought of as a function of time, t, and a diffusion constant, D [22] [4] [42]. In the Brownian bridge, the uncertainty in $\vec{x}$ is low or zero at t and $t + \Delta t$, and maximal at $t + \frac{\Delta t}{2}$ [40]. From this it is clear that the mean of the distribution, $\vec{\mu}$, is the average of the two endpoints of the bridge, $\overrightarrow{x_0}$ and $\overrightarrow{x_1}$. In our case both $\overrightarrow{x_0}$ and $\overrightarrow{x_1}$ are known with certainty. We have two particles in motion that we wish to relate to a Brownian bridge. To set up this relationship, we will think of the coordinate system defined by the line between the two particles (see $\overrightarrow{\delta(t)}$ in Figure 17). Each step taken during a random walk is a step that lengthens or shortens $\overrightarrow{\delta(t)}$.

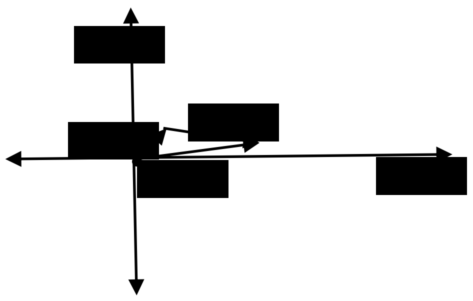


Figure 17. The distance vector between two particles in motion.

Two particles in motion where $\left|\overrightarrow{\delta(t)}\right|$ is the distance between them as a function of time t and $\overrightarrow{x_{0,t}}$ and $\overrightarrow{x_{1,t}}$ locate the particles during flight.

This construction has two notable features. First, the problem expressed this way is one dimensional despite the fact that the particles are moving in three dimensions, and second, the probability of every randomly taken step is a conditional one. That is, we now refer to the conditional probability, $p(y, t|\overrightarrow{x_0})$, when taking each step, where the variable, y, is used to describe the position along the axis defined by $\overrightarrow{\delta(t)}$ in Figure 17 and the origin is defined to be located at $\overrightarrow{x_0}$. For convenience we use the subscripts zero and one to refer to the known location at each of the two time steps. Also for convenience, we let $t_0 = 0$ and $t_1 = \Delta t$. A Brownian bridge is depicted graphically in Figure 18 which shows the position along the one dimensional axis of motion as a function of time. Given the initial distance between the particles, d_0, we ask with what probability y_1 reaches or falls below height r on the bridge within time Δt. We will make use of two boundary conditions – namely the known locations of the particles at t_0 and t_1.

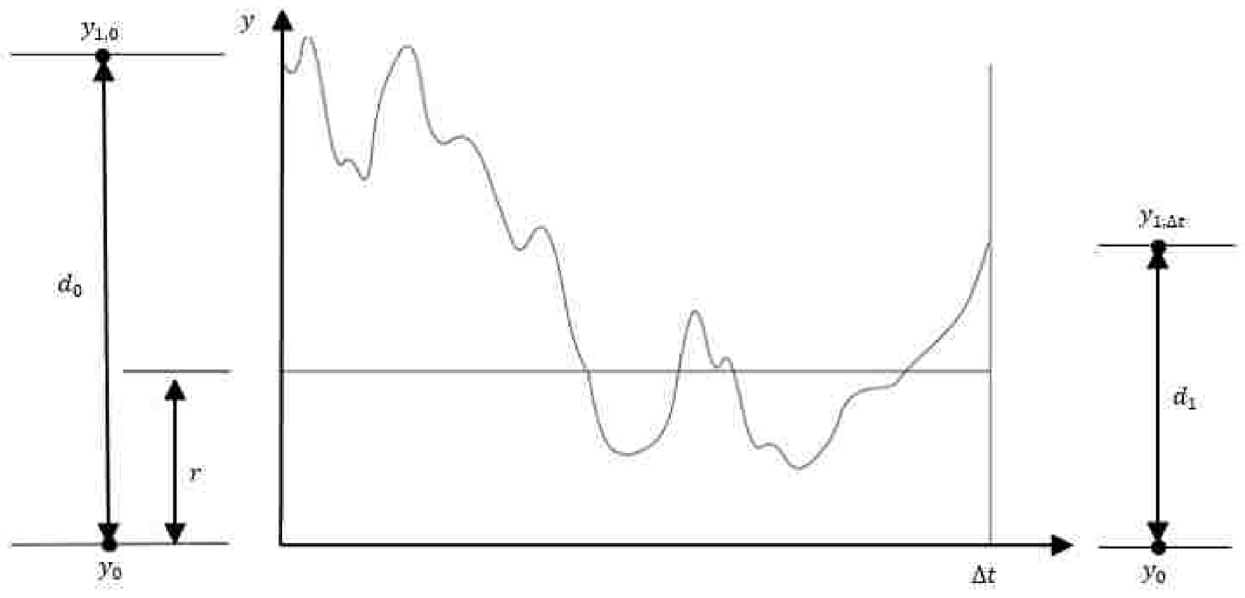


Figure 18. Brownian bridge.

Two particles are initially located at y_0 and $y_1$separated by distance d_0. After time Δt they are separated by d_1. The particles will react if they get closer than r, so we seek the probability that y_1 decreases to height r within time Δt on the bridge.

Let y' be the current height of the bridge while $0 \leq t \leq \Delta t$, then we can write the following cumulative distribution:

$$\begin{aligned} P[y' < r \; and \; t < \Delta t] &= \int_{-\infty}^{r-y'} p(y, t|y_{1,0})dy \\ &= \int_{-\infty}^{y'} p(r-y, t|y_{1,0})dy \end{aligned}$$

The probability density, $p(r-y, t|y_{1,0})$ (see equation 1, sect. 2.3), expresses the probability that a random walker located at y reaches location r (see Figure 18) and it may therefore be written as

$$p\left(r - y, t|y_{1,0}\right) = \frac{1}{(4\pi Dt)^{1/2}} e^{-\frac{\left(r-y-y_{1,0}\right)^2}{4Dt}}.$$

Note that the probability density makes use of the first boundary condition - the location $y_{1,0}$ - and we simply write it in. The quadratic term in the exponent can be factored into the sum of two terms:

$$\left(r - y - y_{1,0}\right)^2$$

$$\left(-y - \left(y_{1,0} - r\right)\right)^2$$

$$\left(r + y - y_{1,0}\right)^2 + 4y\left(y_{1,0} - r\right)$$

leading to a probability density where the conditional and unconditional portions of the probability are isolated from one another.

$$p\left(r - y, t|y_{1,0}\right) = \frac{1}{(4\pi Dt)^{1/2}} e^{-\frac{\left(r+y-y_{1,0}\right)^2}{4Dt}} e^{-\frac{y\left(y_{1,0}-r\right)}{Dt}}$$

The unconditional probability density, $p\left(y_{1,0}, t\right)$, is the Gaussian portion of the equation above. Graphically, it is the location of the tail of the vector $\overrightarrow{\delta(t)}$ (i.e. the location of one random walker) in Figure 17.

$$\frac{1}{(4\pi Dt)^{1/2}} e^{-\frac{\left(r+y-y_{1,0}\right)^2}{4Dt}}$$ (Unconditional Gaussian portion).

$$e^{-\frac{y\left(y_{1,0}-r\right)}{Dt}}$$ (Conditional portion).

The remaining term is the conditional portion which we label p_{bridge}. The second boundary condition may be applied by observing what happens as the threshold distance,

r, approaches the distance between the particles at the end of the bridge, d_1. In this situation, the probability of crossing the threshold distance, r, is 1 and therefore $y = y_{1,\Delta t} - r$. The bridge probability density can then be written

$$p_{bridge} = e^{-\frac{(y_{1,\Delta t}-r)(y_{1,0}-r)}{Dt}}.$$

Note that the equation is symmetric with respect to the beginning and ending inter-particle distances, as it must be since the probability density must be the same no matter which direction we choose to cross the bridge. This probability density can be used, with a roll of the dice, to determine if a reaction occurs despite the fact that the inter-particles distances at two known time steps, t and $t + \Delta t$, are both greater than the reaction radius.

3.5 Serial Implementation

The serial version of this algorithm [22] has been implemented in the C++ programming language. The simulation was set up by creating particles that were randomly generated and assigned locations that were normally distributed and symmetric about the X-axis. This simulates the distribution of particles shortly after the initial burst of ionizing radiation in water. The size of the problem domain is $300\ nm$.

Cells in the proximity grids are objects that are stored in hash tables, and we take care to remove empty cells as the simulation progresses. This mitigates the effect of hash table collisions and keeps the run time cost of iteration through the particles linear in time. The pseudo code for this algorithm is as follows:

```
1. Insert all particles into the priority queue and proximity grid, processing any reactions found.
2. While certificate expiration times are available, and total time $< 1\mu s$ do:
    (a) Pop the next certificate and move the associated particle to its new location.
    (b) Check all neighbors in proximity to it for a reaction. If a reaction occurs, replace reactants with products.
    (c) Update the priority queue with new certificates.
3. Report all particle distributions.
```

Figure 19. Pseudo-code

The reaction radius for each reaction is computed from the diffusion constants for the species involved in the reaction and the reaction rate as described by [30] [42], where R is the reaction radius, k_d is the reaction rate, and D is the sum of the diffusion constants for the reacting species.

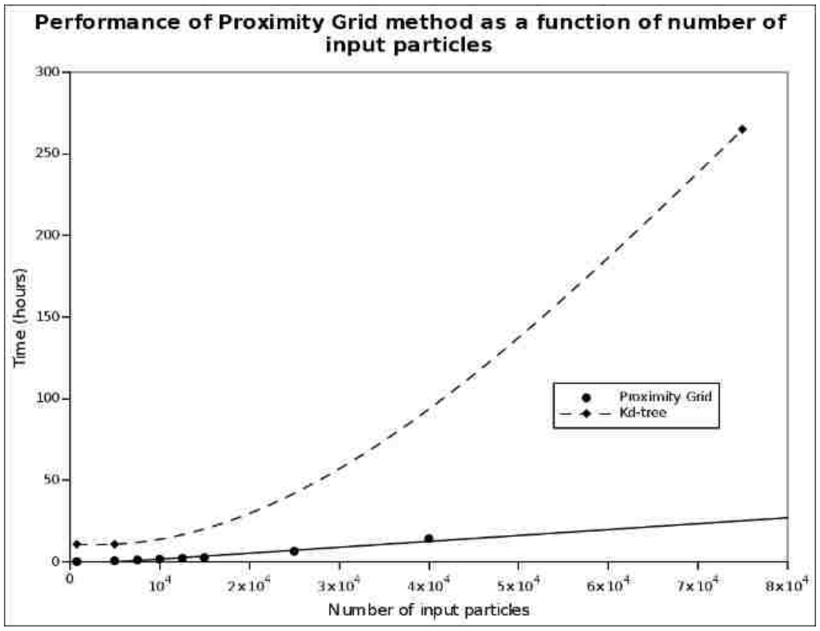
$$R = \frac{k_d}{4\pi D}$$

To handle the situation of how two particles are determined to be close enough for a reaction to occur given that the passage of time is constrained to occur in discrete jumps, we use a Brownian bridge as described earlier.

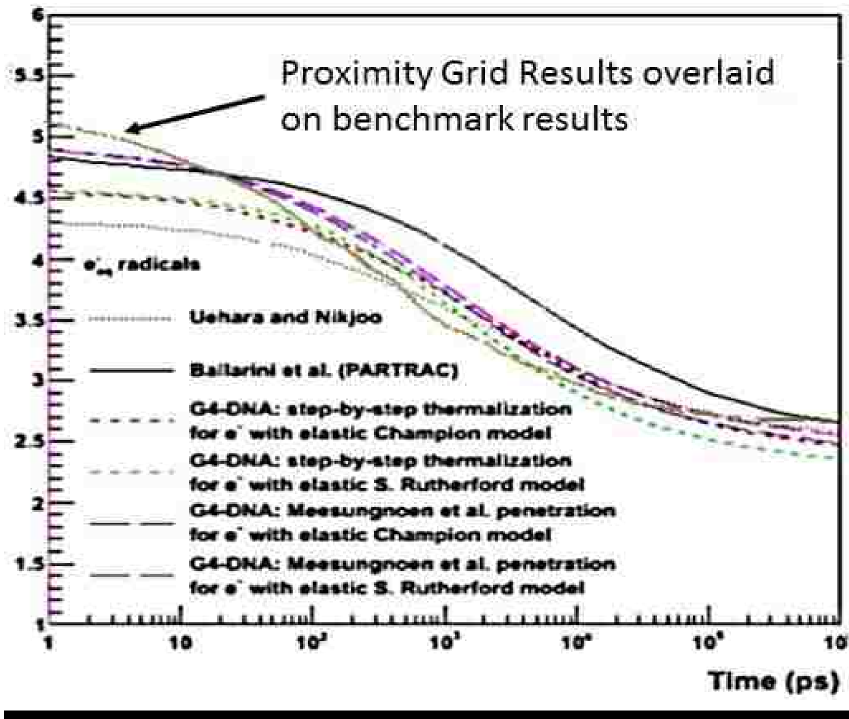
3.6 Serial Version Results

The serial version of the algorithm was tested on a desktop computer running Debian Linux with 16 GB RAM and 8 Intel core i7 CPUs (2.8 GHz) (only one core was used). The running time was calculated from the number of CPU cycles for each algorithm. For comparison, a baseline kd-tree method was also run. Details about the kd-tree method can be found in [22]. Both codes used 75,000 particles as input, corresponding to an input energy of approximately 50 keV. The kd-tree method required 265 hours to complete while the proximity grid method performs this same run in 25 hours – a factor of 10 better. Figure 20 compares the run times between the two methods.

The concentration of radiolytic species has been compared with previous results [4] obtained using the Geant4-DNA toolkit and found the results to be consistent with one



[redacted]



[redacted]

another. Figure 21 shows the results for e_{aq}^{-} species from a total of 7,500 particles for all species (corresponding to approximately 50 keV input energy).

CHAPTER 4

PARALLEL IMPLEMENTATIONS

We present three parallel versions of the proximity grid algorithm based on OpenMP technology and we show a way to achieve a constant 80% scalability with respect to the number of processors used. All three methods parallelize the priority queue and grid data structures, and they do it in a way that avoids the use of expensive mutexes. They are insensitive to changes in particle density as the simulation proceeds. The first implementation is a ticket-based method. It divides space into regions and assigns ticket indexes to them. Each processor must obtain the ticket corresponding to the region for which the next particle needs to be processed. The second method uses a treap data structure to implement the priority queue. The treap's priority queue key is used as the priority queue key, and the heap key is used to organize subtrees in the treap by ticket index. This data structure makes it possible for each processor to process more than one particle for each ticket that it has obtained. The third implementation, dubbed the zone method, is a more standard spatial discretization method. In this method, the priority queue is organized not only by priority queue key, but also by x-coordinate. The three methods are compared for scalability and overall effectiveness.

4.1 Challenges With Respect to Parallelization

Let us now discuss some of the challenges that must be overcome with respect to parallelization of this code. We will discuss the sources of resource contention, the requirement to preserve the accuracy of the physics of the simulation, the restrictions associated with parallelizing the proximity grid data structures, and the choice of OpenMP as the parallelization technology.

4.1.1 Sources of Resource Contention

In order for the various tasks performed by the code to be executed in parallel, it is necessary to identify those computations that do not depend on the completion of other computations – i.e., independent computations. Those tasks meeting this criteria are candidates for parallelization, but those tasks should be made as large as possible because there is an overhead associated with spawning parallel threads of execution. In the proximity grid algorithm, the task being repeated over and over again is the movement of a particle followed by the check for a reaction. This is a situation where the largest unit of parallelizable work is small. It is so small, in fact, that the overhead associated with keeping threads operating in independent regions of space (called ticket acquisition in section 4.2), or the repeated calling of expensive functions such as malloc(), can be non-negligible with respect to the length of time it takes to perform the task.

We therefore explore two possible sources of resource contention in the proximity grid algorithm in order to determine if they cause a loss of scalability. The first is that, since the ticket acquisition process (described in section 4.2) is necessarily a serial operation, there may be some point at which, given enough processors, threads finish their work and begin to wait in line to obtain a ticket – in other words, there may be a point at which the time it takes to obtain a ticket is non-negligible as compared to the time it takes to move a particle and check for a reaction. To determine if this is an issue, we devise a way to give each thread more work to do for each ticket that has been obtained. The result is that threads need to obtain new tickets less frequently, avoiding the bottleneck (if the bottleneck exists). This variant of the algorithm makes use of a treap data structure (described in section 4.3).

The second possible source of resource contention is that insertions and deletions in the proximity grid data structure are frequently made as particles are moved from cell to cell. The particles themselves are retrieved from a memory pool and require no dynamic memory allocation. However, during the hash map insert operation, the C++ operator new() is called in order to allocate the memory used to maintain the data structure (i.e., in the hash map's buckets). Since many threads may be executing code that resize their buckets at the same time, then the call to operator new() may be a bottleneck. Glibc's implementation of malloc, attempts to avoid locking the memory heap as much as possible, however heap locking does occur. We modify the zone method (see section 4.4) to use pre-allocated memory in the hash map data structure with the result being that operator new() is called only during the setup phase and it is not called at all when the Monte Carlo portion of the simulation starts. In addition, we show the result of substituting glibc's malloc() with tcmalloc() – a thread caching version of malloc().

4.1.2 The Accuracy of the Physics With Respect to Simulation Time

Parallel computations must not only be independent in terms of numerical computation and memory access, they must also be independent in terms of any real-world physics being simulated by the overall algorithm. It is not sufficient to isolate data and computation within each thread if the simulated physics within each thread depend upon one another. For example, large scale Monte Carlo simulations of physical environments such as a box full of moving molecules (i.e., a gas, or a liquid) may be parallelized such that one subset of molecules are managed by one thread and a second subset is managed by a second thread. Even if data and computations are kept separate, the physical quantity of time must remain synchronized between the threads in order for the physical processes (i.e., chemical reactions) to proceed as they should. One method of accomplishing this is to break the problem down into spatial domains and process each domain independently for short periods of time, thus keeping the physical quantity of time nearly synchronized.

Spatial domains may be processed in either a deterministic or a random fashion. Random processing of sub-regions in the problem domain can offer some advantages over deterministic approaches. For example, if the problem domain exhibits changes in density over time, the spatial region centered at a randomly chosen molecule can be assigned to the next available processor. In this way, regions of space get a share of processor time that is proportional to their particle density. An example of a fluid flow algorithm of this type has been developed by O'Keeffe [43].

Procassini [44] solves a similar kind of problem in a different way. His method dynamically assigns processors to the domains requiring the greatest amount of work. In this method, the problem geometry is divided spatially and each region is processed in a deterministic order. Each domain is replicated a variable number of times – one per assigned processor – with each processor responsible for handling an equal number of particles in the domain. Periodic testing is done to estimate particle density within each replicated domain to determine if load rebalancing would speed up the calculation. Since the rebalancing operation is expensive, it is only done if it is determined to be worthwhile.

In the proximity grid algorithm, this issue is handled elegantly by virtue of the use of multiple priority queues, one per processor, and each filled randomly with certificates, as described in section 4.2. Because the queues have a random distribution of certificates with respect to particle location, varying particle density does not cause the flow of simulation time to diverge among the processors.

4.1.3 Parallel Forms of the Proximity Grid Data Structures

The research presented here uses hashed containers and a kinetic priority queue to solve the closest pair problem. Parallel forms of these containers are required to parallelize the proximity grid algorithm. Parallel hashing has been studied extensively in the literature [45] [46] [47] [48] [49] [50]. These solutions find ways to avoid or minimize collision resolution in the hashed container. Collision resolution, a necessary component of containers using imperfect hashing, involves the building and accessing of linked lists which does not fit naturally in a parallel environment because it can be difficult to keep all cores busy while list traversal takes place (the lists have an unknown size and therefore an unknown amount of work is required to traverse them). Because of this, perfect hashing is sometimes used, but perfect hashing has the disadvantage of being space inefficient. The approach by Alcantara [50], uses a perfect hash and achieves space efficiency by rebuilding the hash table during every insertion and deletion. The domain is broken down into non-interacting regions, thus this can be done in a massively parallel way. Therefore they remain efficient operations, given enough processors. Breaking the problem down into smaller, non-interacting sub-problems is a key to achieving this efficiency. The serial version of the proximity grid algorithm achieves efficiency, in part, by keeping the hierarchy of cells and particles sparsely organized, and this is accomplished by removing cells from parent cells as they become empty and inserting new cells as needed. Cells in the proximity grid therefore interact with other cells in some spatially local vicinity.

Methods [51] [52] have been developed that interlace regions of space into “active” regions and “inactive” ones. Processors periodically visit and operate on regions marked as active, advancing simulation time in the region for some small length of time (see Figure 22). The active regions are larger than the largest reaction radius or particle movement size so that any reaction or movement that occurs can be processed independently of others.

There are potential drawbacks to spatial decomposition, however. As already noted, approaches such as this can suffer from the fact that since the particles undergo diffusion, their spatial distribution changes over time making their density a time-varying quantity. This means that there can be periods of relative inactivity of some threads with respect to the others as the simulation proceeds. Since we strive to keep all threads busy at all times, this is not an ideal situation. Furthermore, as can be seen in Figure 23, as time marches on and diffusion takes place, the distribution of particles spreads out. This means that the processors responsible for handling the regions at the ends of the sub-domains will be asked to do more work than the ones in the interior regions as time moves forward. This is not only a load imbalance, but it creates a physically unrealistic situation since it causes simulated time to proceed more slowly in the two end regions than it does in the interior ones. The parallel version of the proximity grid method makes use of spatial decomposition, but it does so in a dynamic way that avoids these problems.

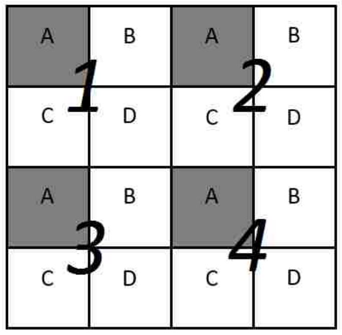


Figure 22. Domain decomposition.

An example of a region of space in the domain of a Monte Carlo simulation divided into sixteen regions and operated on by four processors. The shaded region (region A) is currently active. Processors operate in sequence on regions A, B, C, and D. All operations are isolated from one another.

There are also algorithms for implementing parallel priority queues [53] [54] [55] [56] [57] [58]. Brodal [53] shows how to implement a parallel priority queue using a binomial heap. Constant time insertion and deletion is achieved by pipelining a "merge key" operation to process n queue operations at once given n processors. The method makes use of one local queue per processor. Operations to each local queue are performed in isolation from the other queues, thus no mutexes are required. For any operation on the global queue, Q, Brodal defines a "merge key" function to assist in performing the operation. For example, let $\overline{\forall}$ denote a parallelized loop over all i, then:

$$Minimum(\ Q\ ) \equiv \overline{\forall}\ i\colon MergeKey(\ Min, Q_i\ )$$

is Brodal's definition of the $Minimum$ operation where merging keys means selecting the minimum value of all local queues in this case.

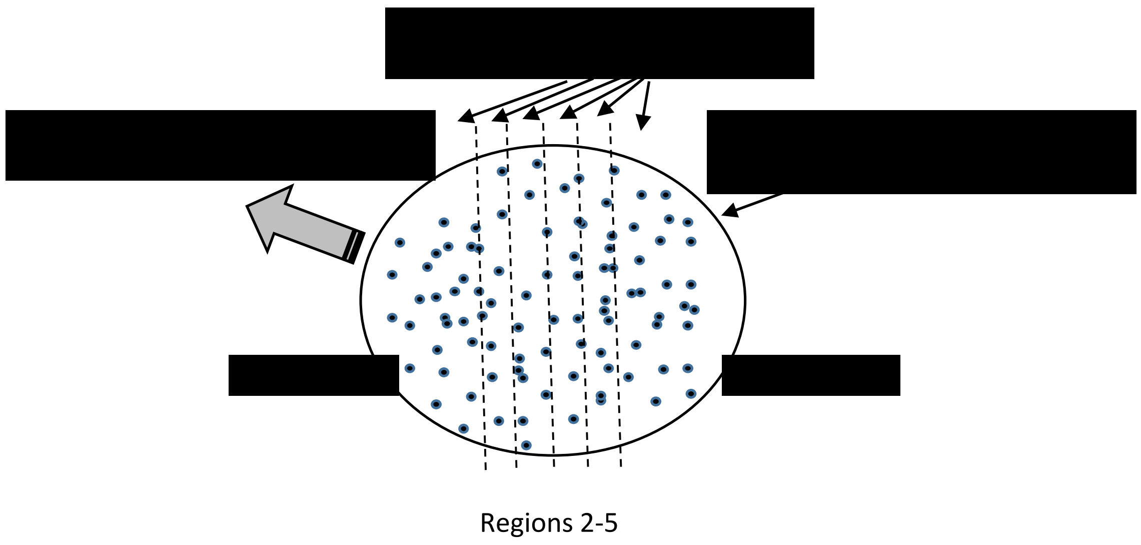
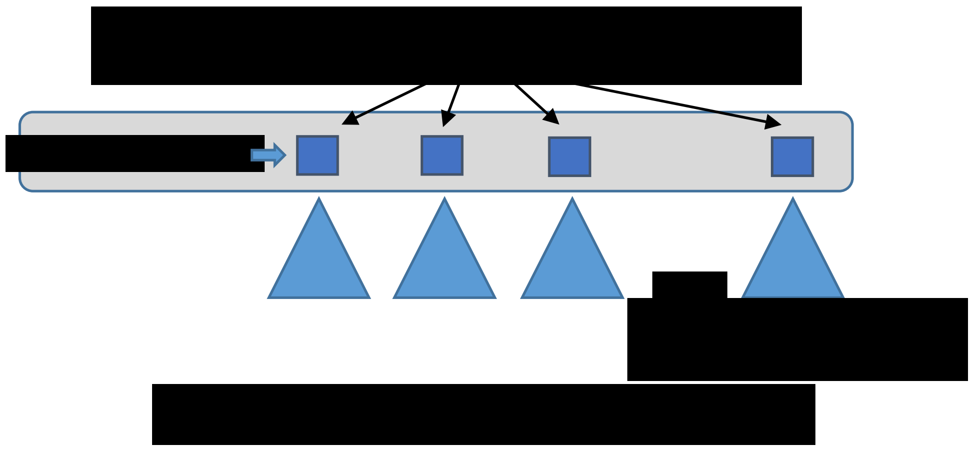


Figure 23. Spatial decomposition & workload analysis.

As diffusion takes place, there are more and more particles in regions 1 and 6. This can cause a workload imbalance among processors.

Given set $E = \{e_1, e_2, e_3, \ldots, e_p\}$, for some maximum number of elements p, the insert function is defined similarly.

$$Insert(\,Q, E\,) \equiv \overline{\forall}\; i : MergeKey(\,Min, Q_i, e_i\,)$$



In Brodal’s implementation, the case $e_i = e_j$ where $i \neq j$ is not disallowed. In this case the same element exists in multiple local queues – a waste of memory – but otherwise harmless. For the parallel algorithm presented in the next section, a parallelization of the priority queue similar to the technique by Brodal is used.

4.1.4 The Choice of OpenMP as the Parallelization Technology

Finally, for the work done here, only OpenMP parallelization is considered because it is desirable to run this algorithm on a desktop workstation that might be found in a radiation oncologist’s office. Although such a workstation could reasonably be expected to have a GPGPU (e.g., a CUDA-capable nVidia graphics card), today’s best graphics hardware has an upper bound of approximately 4GB of on-board memory – a size that could be too small for large problems, whereas main memory can be configured with 64 GB memory or more. Thus, the OpenMP approach seemed the most appropriate to follow.

4.2 The Parallel Priority Queue Variant

The parallelized version of the proximity grid method makes use of a dynamic form of spatial decomposition that avoids the pitfalls of a dynamically changing spatial distribution of particles mentioned in section 4.1. This, combined with a parallel priority queue inspired from Brodal’s queue, are the basis of the parallel implementation of the proximity grid method. In the discussion that follows, when Brodal’s priority queue is discussed, we use the term, key, to mean the unique and sortable id that refers to an item in the queue. When, by analogy, discussion turns to the kinetic data structure in the proximity grid method, we instead use the term, certificate, which has an expiration time that acts as a key.

What follows next is a discussion of the way the two data structures – the proximity grid, and the kinetic data structure – are parallelized to support the parallel implementation of this algorithm.

4.2.1 Parallelized Grid

The proximity grid is a network of cells within cells. Cells are located using hashed containers with each key mapped to a particular region of space. Insertion and removal of cells in the hashed maps cause rehashing of the container as the data structure changes size. All read/write access to the maps would therefore need to be guarded against concurrent use by multiple threads. In the parallel version of the proximity grid algorithm, the arrangement of hashed containers having an (X,Y,Z) key has been modified to be a vector of hashed maps. The vector, containing the X-coordinate key, is pre-allocated to the largest size needed by the simulation and insertions and removals in the vector are not done. The hashed maps have a (Y,Z) key. As will be shown in the next section, this means that, the proximity grid stores shared data (the particles in the simulation), yet does not require any form of mutually exclusive access to the data.

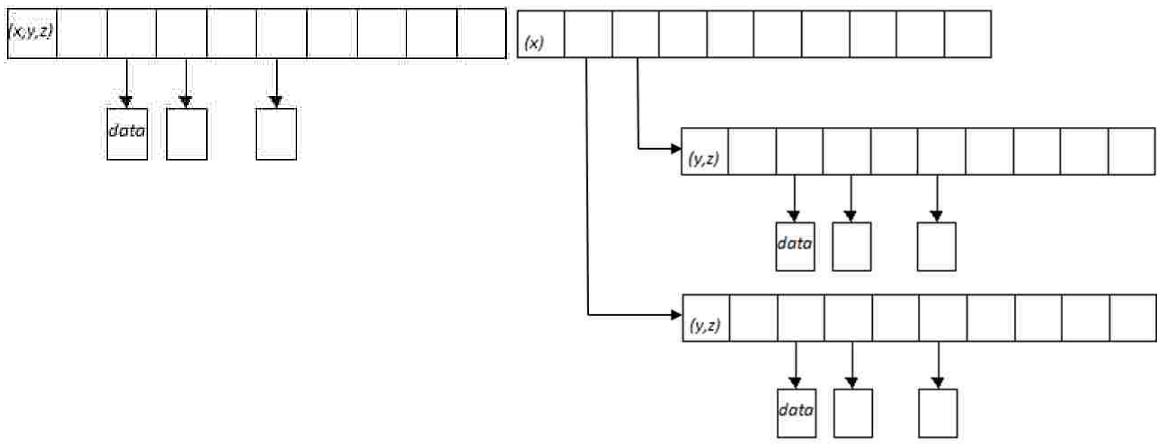


Figure 25. Serial vs. Parallelized Hash Map.

Hash map storing a cell index key appropriate for a given radius of a chemical reaction in Cartesian coordinates. In the serial version of the algorithm, a single map is used per radiochemical reaction (left side).

A vector of hash maps is used in the parallel version of the algorithm to avoid concurrency issues. The vector of hash maps is indexed by the x cell coordinate and the hash maps use a (y,z) coordinate as the key (right).

4.2.2 Parallelized Kinetic Data Structure

The parallelized priority queue is partitioned into p queues where there are p processors. Each queue is filled with an equal number of certificates selected at random, thus the queues are not subdivided spatially. Rather, they are thoroughly spatially mixed. The top of each queue contains a certificate failure time that is either close to, or exactly at, the next time step. In Brodal's implementation [53] of a parallel queue, the top of the parallelized queue is the item that has the smallest key among the items at the tops of the individual queues. Where Brodal computes an absolute minimum among them, we process each certificate in parallel. This is the definition of one parallelized time step.

The following operation is defined (shown in Figure 28), returning a list of minimum (more specifically, near minimum) expiring certificates, given queues, Q_i, number of processors, p, the standard Min priority queue operation, and the standard list $Append$ operation.

$$MinimumList(\, Q, p\,) \equiv \forall\, i: Append(\, Min, Q_i\,)$$

The time steps in $MinimumList(\, Q, p\,)$ can be processed in parallel if the associated particles are far away from one another. More precisely, the time steps may be processed in parallel without the use of mutexes if the distance between any pair of particles is greater than six times the largest reaction radius (see Figure 26).

To achieve this separation, a ticket mechanism is set up using the X-axis. One ticket corresponds to a region of the X-axis that is the size of the largest reaction radius. Before any processing is allowed on a given certificate, the code attempts to aquire up to six tickets. Three tickets for the source cell and the two cells surrounding it, and three tickets for the destination cell and the two cells surrounding it. If one of the six tickets have already been taken, then processing is not allowed. The regions assigned to a given processor are therefore dynamic and non-overlapping. Ticket acquisition is serialized by an OpenMP critical section. The serial nature of the ticket acquisition mechanism creates the possibility of the performance bottleneck described in section 4.1.1. In order to investigate this possibility, a treap version of this algorithm was developed (see section 4.3).

A reaction may occur as each certificate is processed. In this case, other certificates will be invalidated in addition to the one at the top of the queue. Those certificates being invalidated may reside in any of the queues, however there is no need to protect any memory from mutual access from multiple threads. The reason is that each thread is working in its own region of space along the X-axis. When reactions occur, reacting particles may safely be marked as destroyed regardless of which queue the certificate for that particle belongs to. The particle is destroyed later by the thread that owns the certificate at the appropriate time.

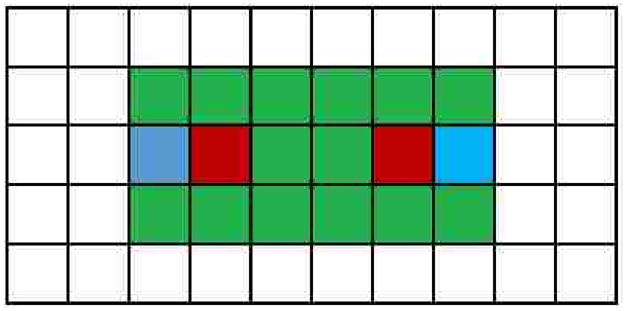
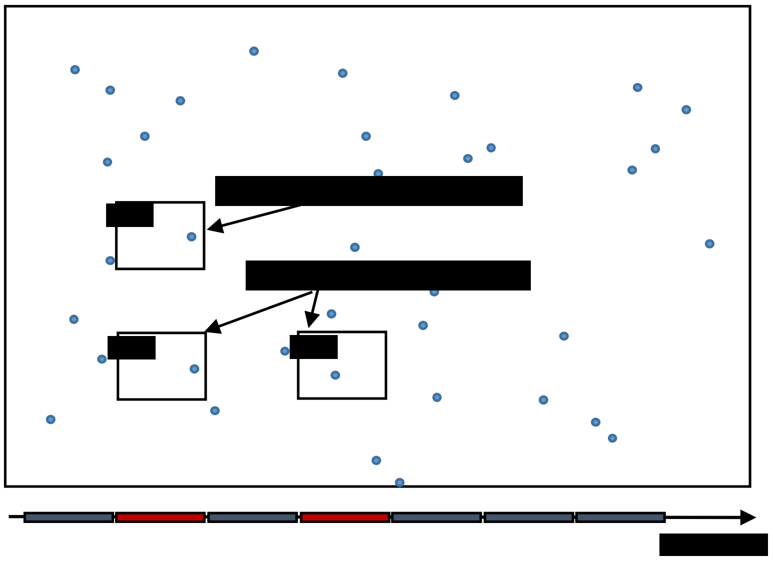


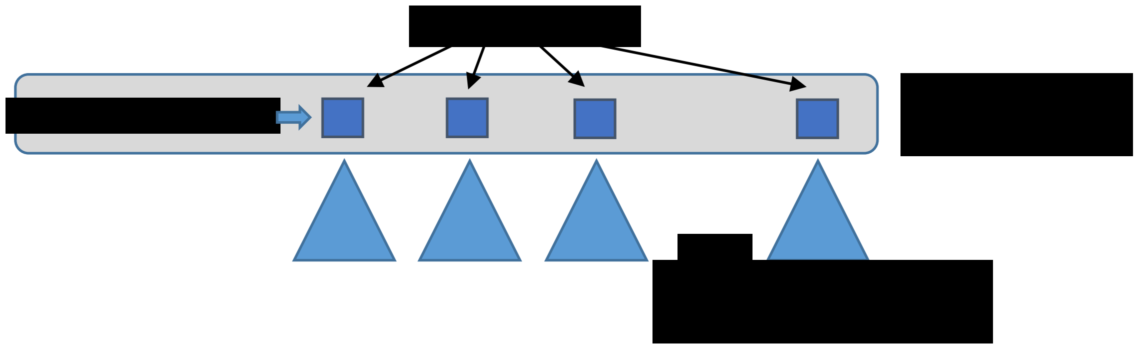
Figure 26. Minimum separation distance required for parallel processing.

The squares represent the cells with the largest reaction radius. Blue squares are source cells and red squares are destination cells for two particle movements. Green squares represent cells that may contain particles reacting with the two particles in motion. As long as the particles are at least six cell radii apart, the movements and any associated reactions can be processed in parallel.

The expected worst case run time of the serial version of the algorithm is $O\big((n+k)\log n\big)$ given n particles and k required insertion/removal operation pairs in the priority queue. For the parallel version this becomes $O\left(\left(\frac{n+k}{p}\right)\log\frac{n}{p}\right)$ where there are p processors. A flowchart of the process is shown below.



[redacted]



[redacted]

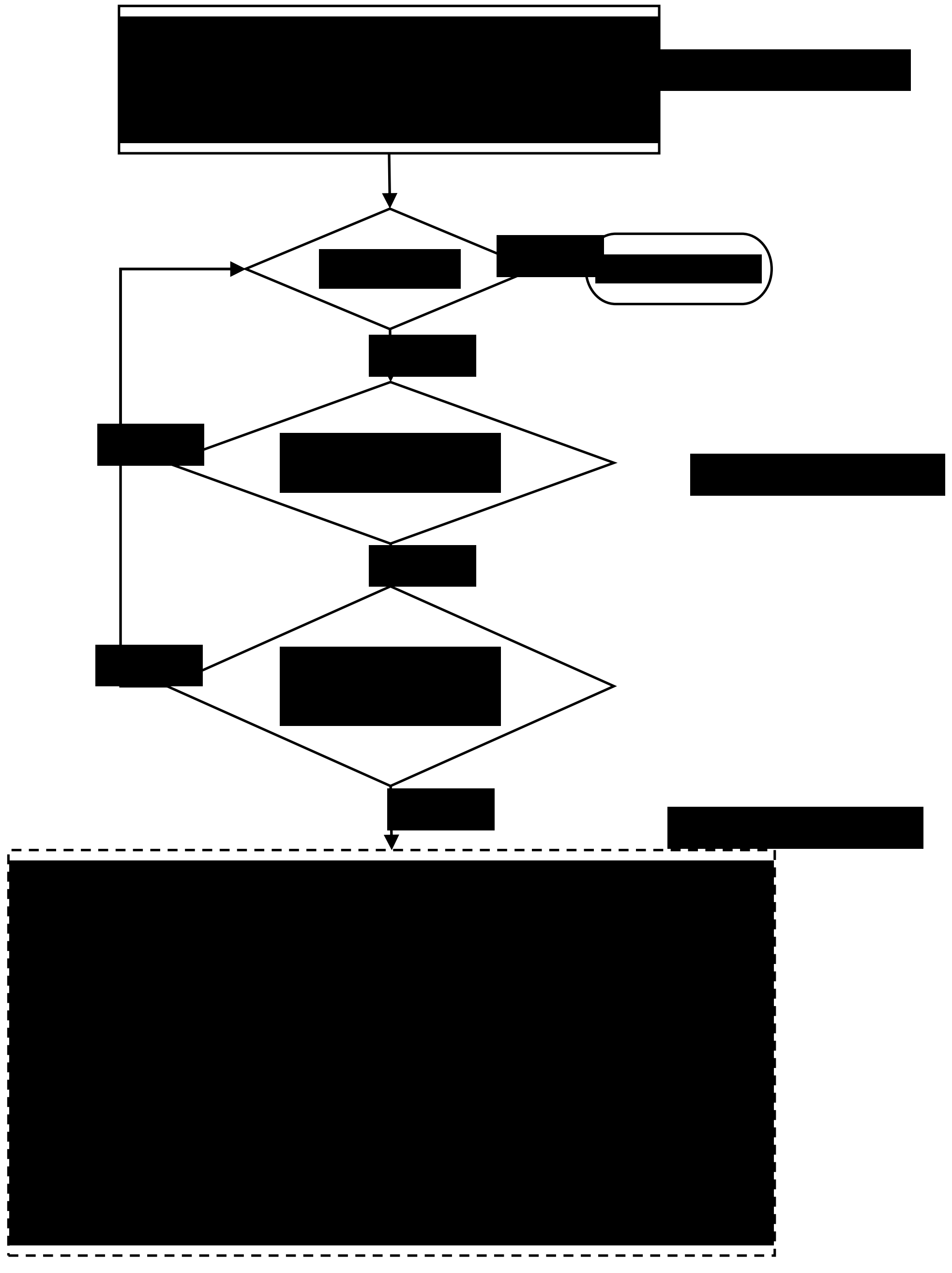


Figure 29. Flowchart of the parallel priority queue method.

4.3. Parallel Priority Queue Variant Using a Treap

A portion of the parallel version of the algorithm described in section 4.2, namely the ticket acquisition mechanism, operates serially. The question arises whether or not this serial section of code might become a bottleneck as the number of parallel threads increase. That is, as the number of threads increase, is there a point at which the work done by one thread to process a ticket is roughly equal to the work done by a large number of threads attempting to obtain a ticket? If such a point exists, threads will begin to wait in line to acquire tickets, and thus become a bottleneck. This issue can be resolved if each thread can be given more work to do while it has a ticket. In the context of the priority queue algorithm, doing more work means processing more certificates near the top of the queue in the same space for which it has already obtained a ticket. This may be accomplished elegantly by implementing the kinetic data structure with a data structure called a treap [59] instead of with a priority queue. A treap data structure combines the features of a binary tree and a heap. It offers two keys. The first key is a priority key. All child nodes of a given node in the treap have a lower priority than the node itself. Thus a treap's priority ordering makes it a suitable replacement for a priority queue. In addition, the nodes of a treap are organized in heap order. Therefore the second key – the heap index – provides a way to easily identify those nodes near the top of the treap (and thus close to the next certificate expiration time) having a given ticket index. In this version of the proximity grid algorithm, we replace the priority queue with a treap. The priority ordering of the queue is the certificate expiration time, just as it is in the priority queue version. The heap index is the ticket index of the ticket with the lowest X-coordinate among those tickets affected by a particle movement. The next figure shows a possible arrangement of the certificates in the treap version of the kinetic data structure.

Although the treap has the features of a binary tree, it is only well balanced if the priority key is randomized with respect to the heap key. This is not strictly the case here, however the certificate expiration times are reasonably well randomized with respect to the X-coordinate. Thus, when the number of particles is large, the tree can be said to be approximately well balanced.

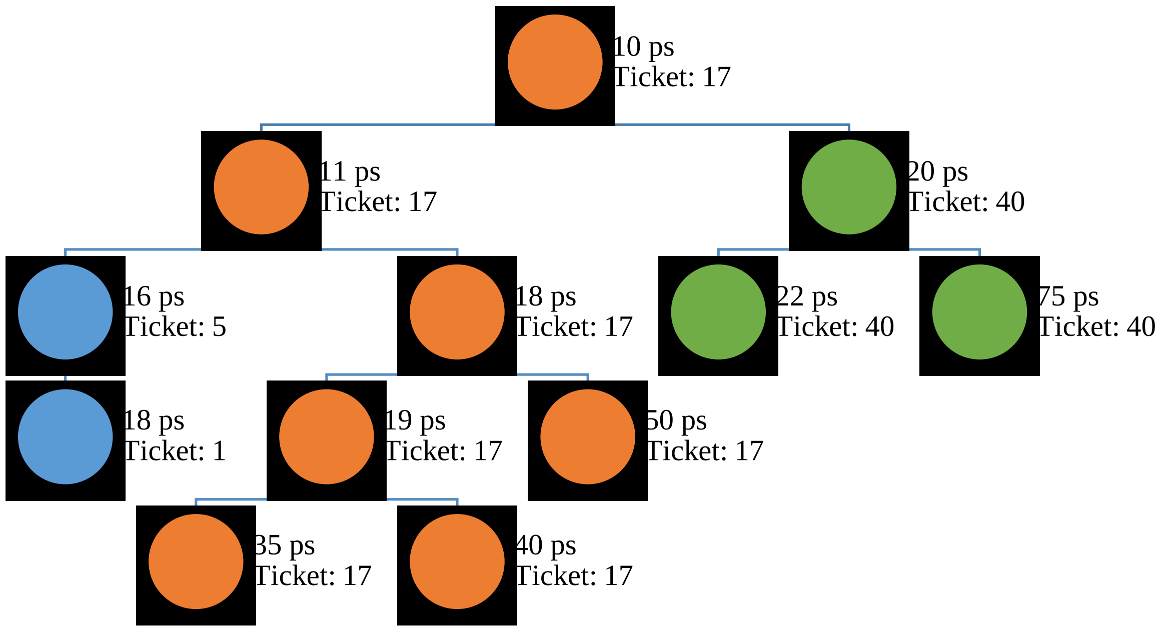


Figure 30. A possible organization of the treap.

The top node (orange) has the highest priority and will be processed next. It also has ticket index 17. Other certificates near the top of the treap with ticket index 17 are easily found by traversing the children of the top node. The blue nodes have ticket indices less than index 17 (following heap order) and priorities later than the top priority. Green nodes have a ticket index greater than 17 and priorities later than the top priority.

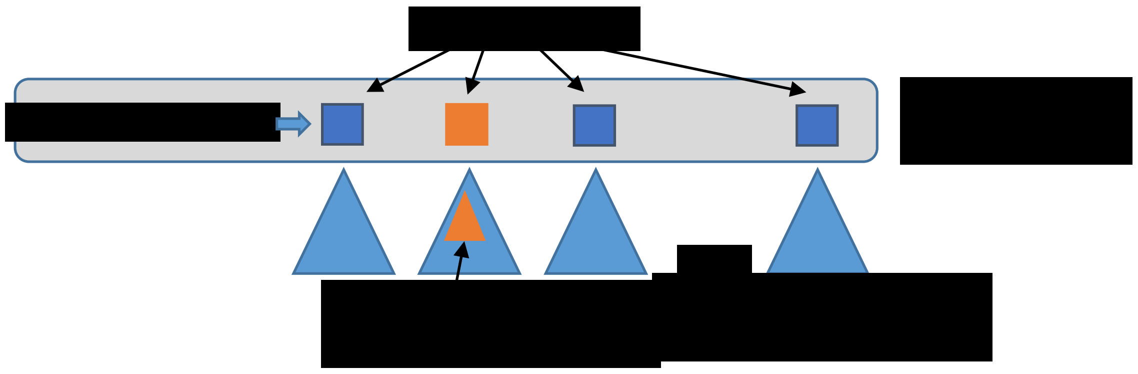


Figure 31. Parallelized kinetic data structure (treap variant).

The parallelized kinetic data structure consists of one treap per processor. The certificate at the top of each queue is processed in parallel, representing one time step, just as in the priority queue variant. In addition, all nodes having the same ticket index as the top node are also processed (nodes shown in orange).

The algorithm is modified in the following two ways. First, during ticket acquisition, if a ticket is not available for the top certificate in the queue, an attempt is made to obtain a ticket for the child node in the treap that does not have the top node’s ticket index. If a ticket still cannot be obtained, one more attempt is made with the next child node. If these two attempts fail, the current thread is masked out, just as with the priority queue version. The second modification has to do with how certificates are processed. Observe that the child node of the current certificate in the treap that has the same ticket index as that of the node itself is a priority queue of certificates with the given ticket index. In addition to processing the top node of the treap, we also process the nodes of this inner priority queue in priority order until a small simulation time expires. The result of these two changes is that fewer threads sit idle due to ticket conflicts and more work is done in the region in which each thread has obtained a ticket so that the need to obtain a new ticket occurs less frequently.

4.4. Parallel Spatially Discretized Variant

The previous sections explored the possibility of using a ticket acquisition mechanism to give threads permission to work in particular regions of space. Now we alter the algorithm so that threads work in pre-arranged spatially distinct regions that we will call zones. Each zone is a section of the X-axis such that threads execute six zones apart. Execution occurs in round-robin fashion with a stride of six as shown in Figure 32. The width of a zone is the maximum reaction radius among all reactions considered in the simulation. This distance allows for particles to move from one zone to any adjacent zone and react with particles in any resulting adjacent zone and still be operating in independence (See Figure 26). There is one priority queue and one proximity grid per zone. The movement of particles across zones is handled without the use of mutexes because it has been pre-arranged that all adjacent zones are unused by any other processor. This version of the algorithm, in contrast to the first two versions, is subject to the performance effects of changing density discussed in section 4.1.2. However, so long as the density gradient is small with respect to the width of a zone, the effect may be negligible. We will see in section 4.5 that this is the case. Execution occurs in six rounds. During round one, all processors work in the zones assigned to round one (see Figure 32). During round two, all processors work in the zones assigned to round two, and so on, until all six rounds have been completed. After the sixth round is complete, the process repeats until the simulation is done.

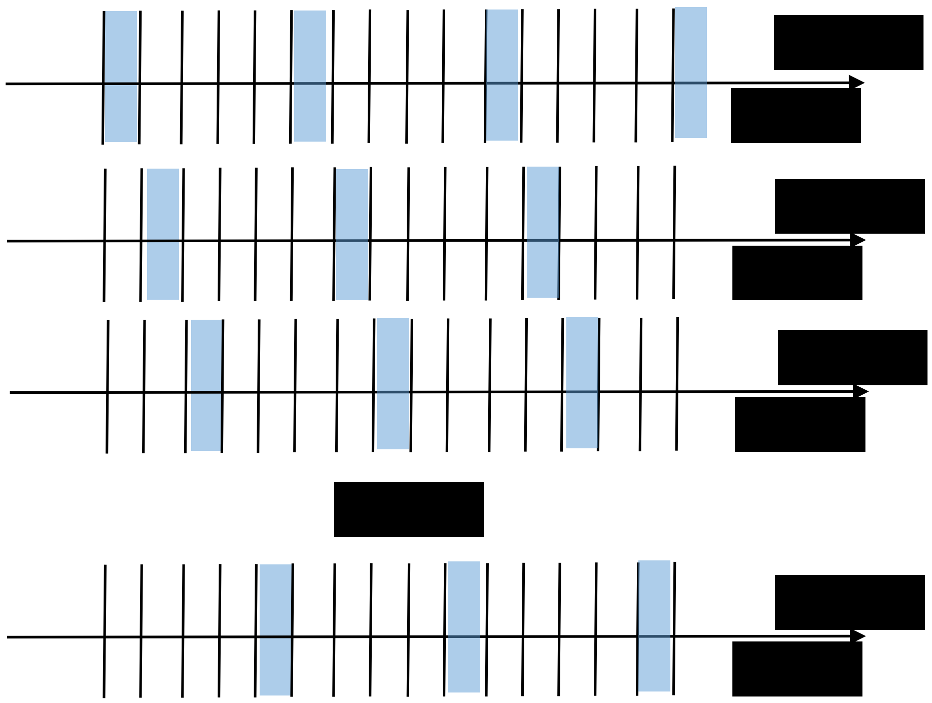


Figure 32. Round-robin execution of the Zone method.

Execution occurs in the shaded region (zone). Processors work in all shaded regions of the X-axis during round one first, then the shaded regions of round 2, and so on, until all six rounds are complete. Then the process repeats.

4.5. Results for Parallel Versions

The OpenMP version of the code based on the priority queue method was compared with the serial version on a desktop computer running Debian Linux with 16 GB RAM and 8 Intel core i7 CPUs (2.8 GHz). The run time was calculated based on "wall clock time" using OpenMP's omp_get_wtime() function.

The serial version was run using 70,000 particles requiring 9 hours to complete. The OpenMP version, using 8 cores, ran the same problem in 0.36 hours – a factor of 25 better. Figure 33 and Figure 34 compare the run times between the two methods.

The concentration of radiolytic species has also been compared with previous results [4]. Figure 35 below shows the results for e_{aq}^- species from a total of 7,500 particles for all species (corresponding to approximately 50 keV input energy) for both the serial and parallel versions.

Further testing was done using on an AMD Opteron 6174 processor containing 24 cores (48 hyper-threaded cores). To better gauge the scalability of the results, only the first 24 physical cores were used. The tests that follow all simulate 100,000 input particles.

Figure 36 shows that the priority queue and zone variants perform about as well as one another. The treap variant does not beat them. Therefore it is apparently not advantageous – at least out to 24 cores - to process more certificates with a given ticket index for every ticket that has been acquired.

The scalability of the three methods is also shown in Figure 37. All three methods show comparable scalability. The code runs approximately 11 times faster using 24 cores than with a single core for all three code variants. When the zone method is modified so that the hash table uses pre-allocated, large, fixed size buckets, the scalability is significantly improved. Twenty four processors are able to improve the run time by a factor of seventeen, but the single processor run time performance is degraded. It is noteworthy that the efficiency quickly levels off and stays approximately constant at 80%.

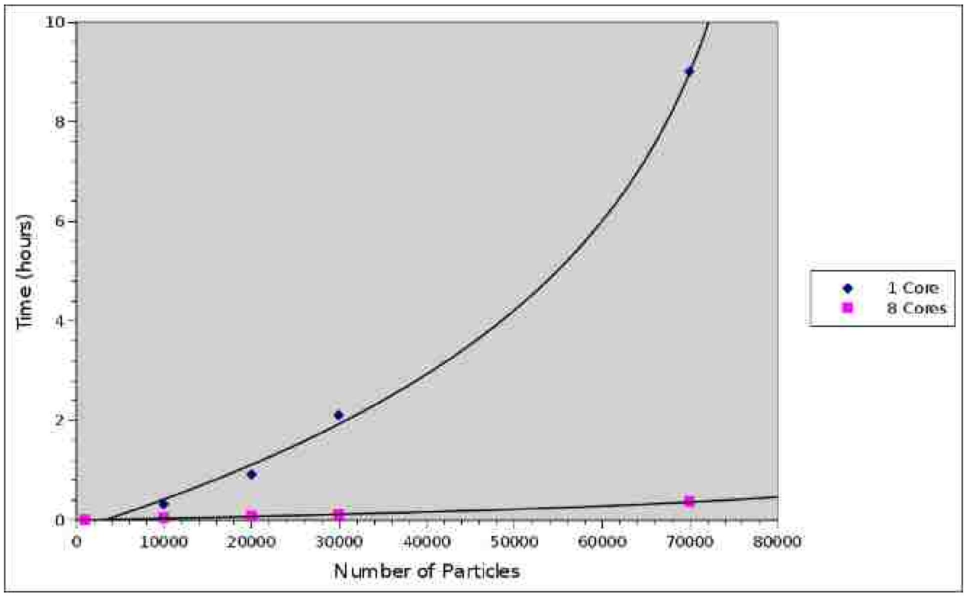


Figure 33. Comparison of the serial vs. OpenMP parallelized priority queue version of the code.

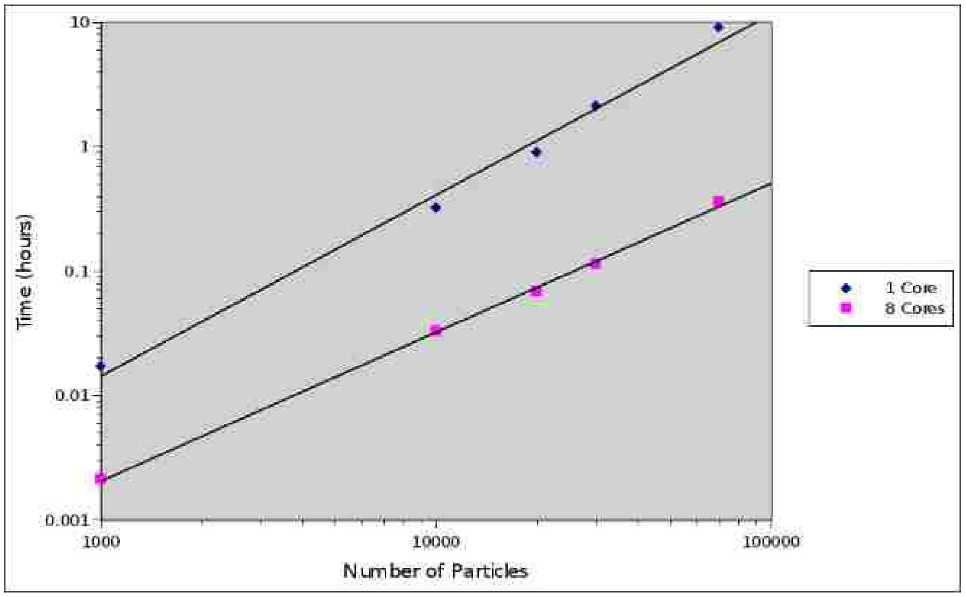


Figure 34. Log-log plot of the run times in Figure 33.

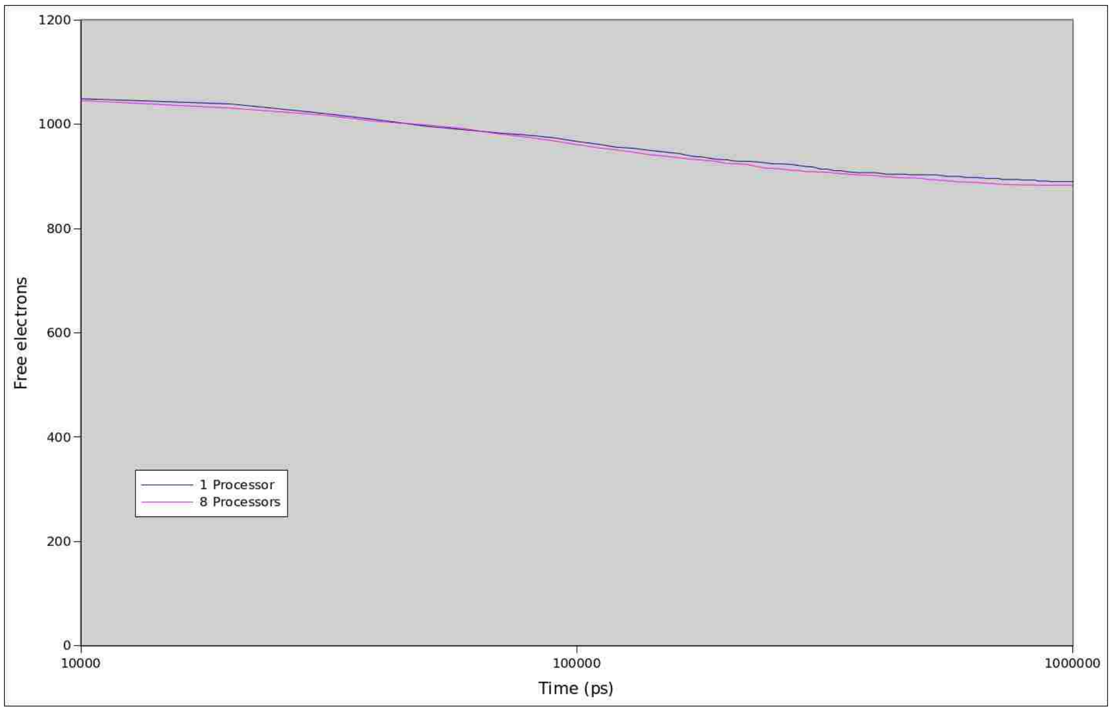


Figure 35. Comparison of the serial and parallel results showing agreement in the number of free electrons over time.

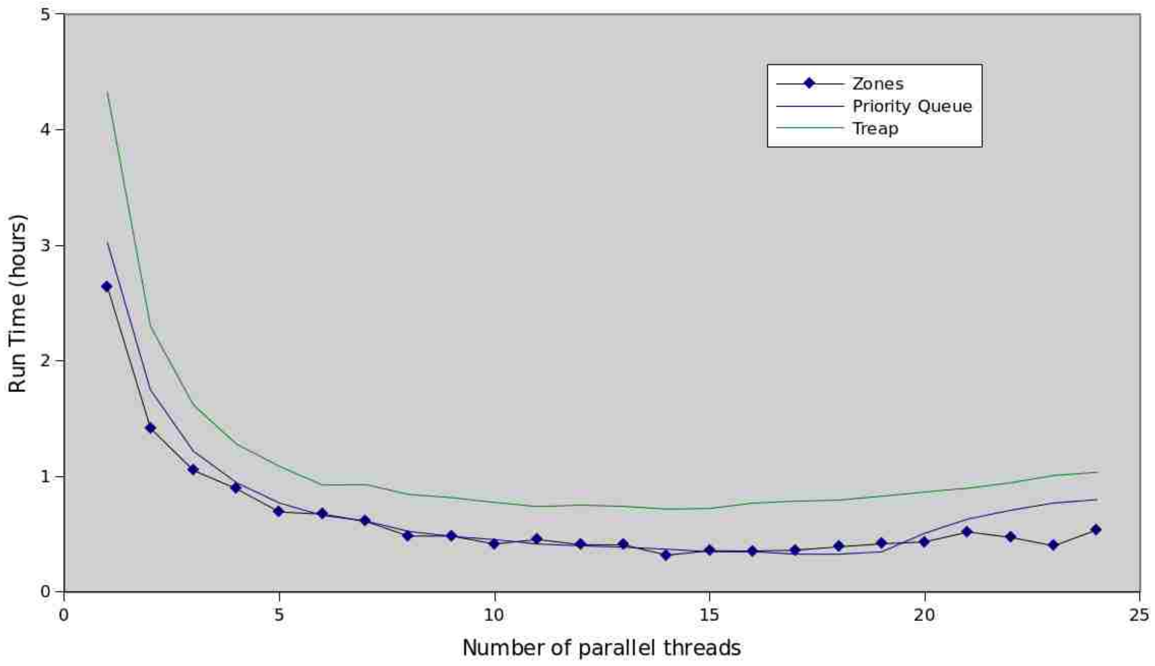


Figure 36. Comparison of Zone, Priority Queue, and Treap Methods.

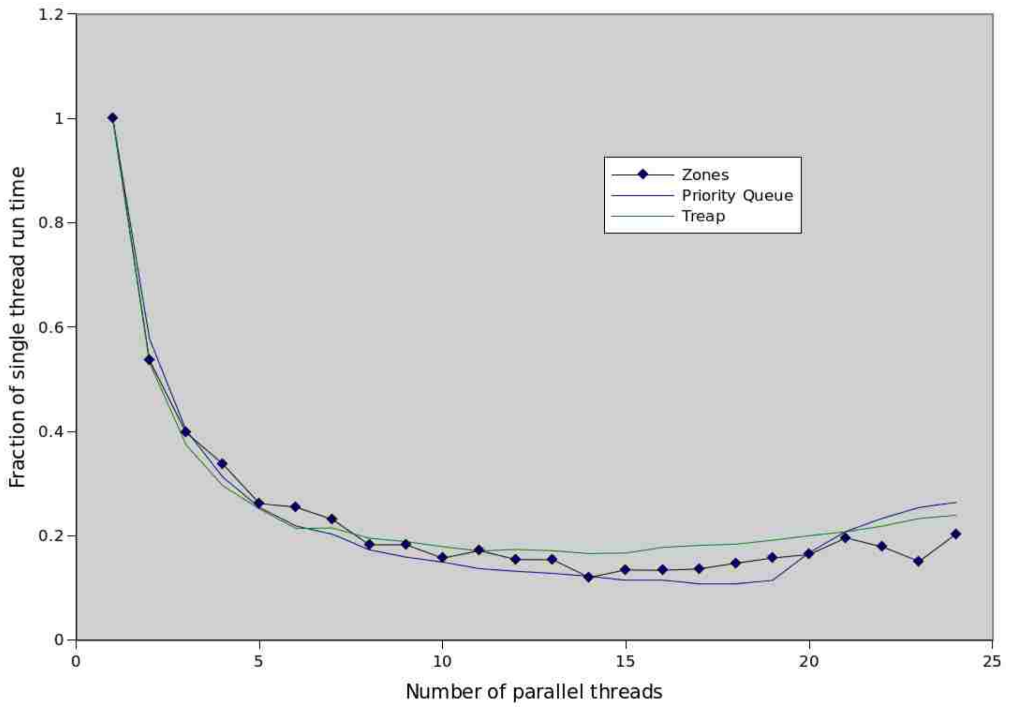


Figure 37. Scalability of the Priority Queue, Treap, and Zone Methods.

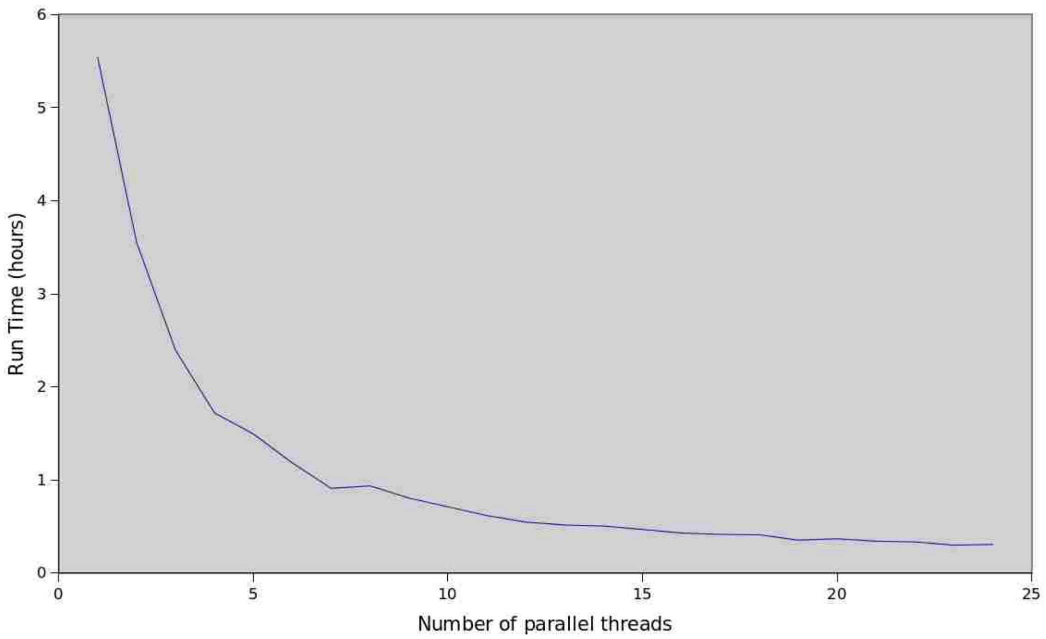


Figure 38. Zone Method With Pre-Allocated Memory.

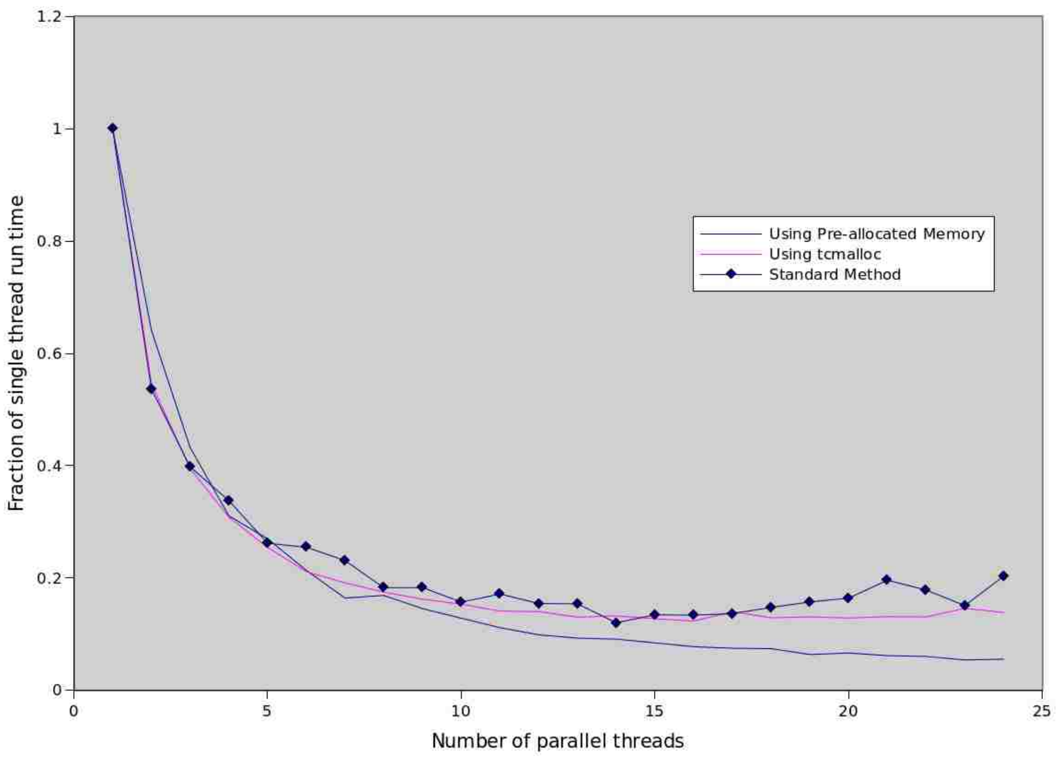


Figure 39. Scalability of the zone method.

The methods shown use pre-allocated memory and memory allocation with the tcmalloc [60] (thread caching malloc) library. The original zone method is shown for comparison.

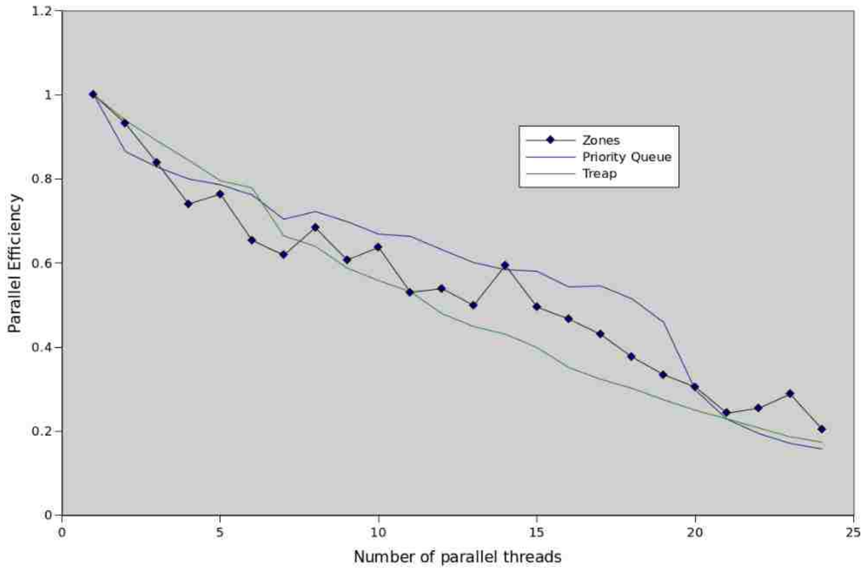


Figure 40. Efficiency of Parallel Methods.

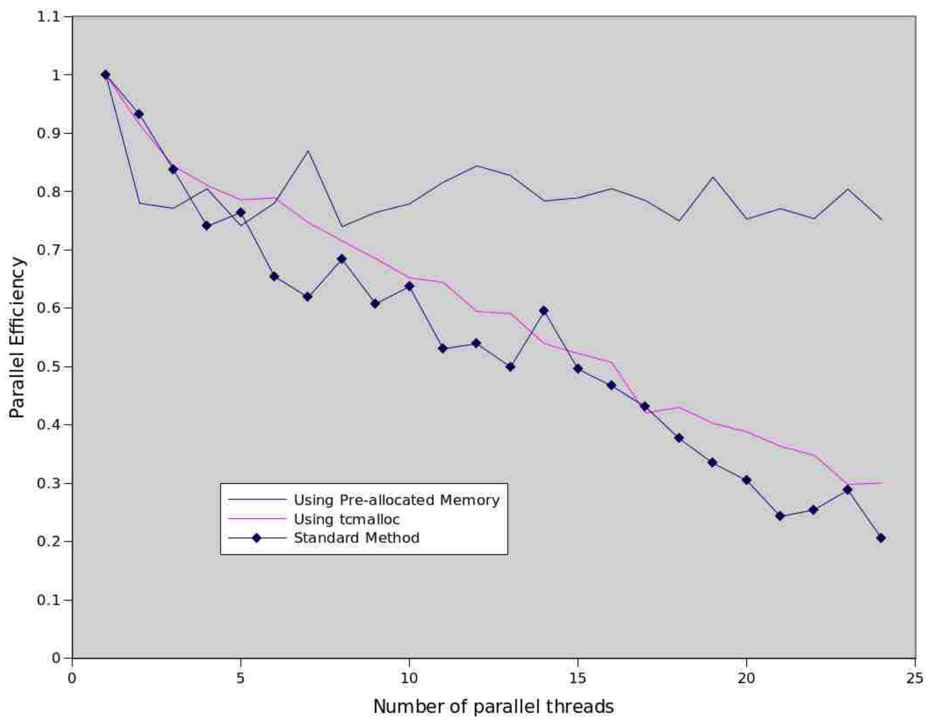


Figure 41. Efficiency of zone method and variants.

In one variant, the proximity grid is implemented to use a large, fixed size, pre-allocated pool of memory and no memory is allocated during the simulation. In the second variant, the c++ library's malloc is replaced with tcmalloc [60] (thread caching malloc).

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion and Discussion

We have shown an efficient method for simulating the radiolytic reactions that cause radiation damage in irradiated living tissue. We have shown how a worst case run time of $O\big((n+k)\log n\big)$ can be achieved in the serial implementation– a significant improvement over the $O(tn\log n)$ run time of the kd-tree method – without sacrificing accuracy – by designing a data structure (the proximity grid) that exploits locality in the search for near neighbors using many simultaneous definitions of nearness of potentially reacting particles. A simulation of 75,000 input particles taking 265 hours (11 days) by a benchmark kd-tree version of the code was completed in only 25 hours using the proximity grid method.

The physics of simulations of this type are often simplified by assuming that the body is composed entirely of water. The proximity grid data structure supports any number of reacting particle species, making it generally useful in situations where this simplification is not imposed. Therefore this data structure permits realism with respect to the number of particle species and equations that may be simulated.

We have provided an elegant parallel implementation that avoids the use of mutexes and we have shown parallel scalability out to at least 24 processors. Our solution exhibits a constant parallel efficiency of 80%, achieved by using a hash map data structure with a fixed number and size of buckets. It is elegant because its efficiency is insensitive to changing particle density as the simulation proceeds. No one processor advances past another in simulated time during the simulation and no one processor has less work to do than another.

A version of the code was developed using a treap data structure as the priority queue. A treap (tree and heap) data structure is a tree whose nodes are additionally organized in heap order. A treap offers a priority key and a heap order search key. By using the additional heap key to store certificates by ticket, it becomes possible for each thread to process more than one certificate for each ticket it had acquired. The treap version of the code made it possible to determine whether or not ticket acquisition – the only serial portion of the code – was a performance bottleneck. That is, we wanted to know if threads were waiting in line to obtain a ticket. The results of the treap version vs. the priority queue version (Figure 36) reveal that the ticket acquisition mechanism does not constitute a bottleneck – at least out to 24 parallel threads. In the event that adding additional processors reveals that ticket acquisition does become a bottleneck (this can be identified by generating a plot like the one in Figure 37 and observing that the treap and priority queue lines cross), the treap version of the code may be used instead of the standard priority queue as an elegant way to remove it.

A version of the code is provided that decomposes the problem into spatial domains. This is a more standard approach to solving the problem that serves as a baseline for comparison. It is subject to load imbalances due to changes in particle density and as a result, care needed to be taken to compensate for this problem. We handled the issue by periodically pausing the operation of threads until the others caught up. We find that the proximity grid version compares well with it and is a more elegant approach since it requires no special handling due to load imbalances caused by changing particle density.

For those versions that did not use pre-allocated memory in the hash map data structure, useful scalability was observed out to approximately 13 parallel threads. It was found that adding additional processors beyond this number does not help performance and, in fact, performance decreases slightly. Internal memory allocation by the hash mapped data structure in the proximity grid is a bottleneck that inhibits efficient parallelization (see Figure 38 and Figure 41). As the simulation runs, allocation with malloc() occurs only when the buckets in the hash map itself are resized. This bottleneck was explored further by substituting the system's memory allocation function, malloc(), with tcmalloc() [60] (a thread caching version of malloc()). The idea is that if memory, once allocated, was cached on a per thread basis for use in future requests for memory, then the bottleneck could be removed. It was found that thread caching does not remove the bottleneck (see Figure 41). However, writing the hash map data structure in such a way that both the number of buckets and the number of items in a bucket are fixed then the bottleneck is eliminated. Note that the hash map implemented this way is cache-friendly, whereas a standard hash map, making use of pointers to allocated memory, is not. This fact likely plays a role in the performance improvement.

5.1 Future Work

This work was done with the hope that it could eventually be integrated into the GEANT4-DNA toolkit. The proximity grid algorithm is in a sufficiently mature state that an effort like this could be undertaken. Integration should proceed with the ticket-based parallel version because it has been shown to be as efficient as the spatially discretized version and by design it doesn’t suffer from the negative performance effect of varying particle density.

With regard to memory pre-allocation in the hash map data structure, it solves the efficiency bottleneck with respect to parallelism and an implementation in Geant4 can certainly proceed with it. However, it is desirable to provide an algorithm that is not bound by this code restriction. To improve on this, areas of future research should include writing custom versions of malloc() and free(), investigating the possibility that the code suffers from false sharing, and running the code on other hardware architectures.

The standard versions of malloc() and free() are general purpose memory allocation and disposal functions. Custom versions of them could be written to address the specific needs of this code. For example, they could cache all disposed memory on a per thread basis for later re-use, giving none of it back regardless of size the way tcmalloc does, and they could provide memory in chunks that are guaranteed to be far enough away in physical memory so as not to cause false sharing. If false sharing is the cause, then there will be data blocks in the code that are within 64 bytes of each other (the width of a cache line) that are being written to by more than one processor at a time (invalidating the entire cache line). One place that false sharing might occur is in the case when threads are operating in adjacent (adjacent in physical memory) or nearly adjacent hash maps. A reasonable test to see if this is at least a possibility is to vary the number of input particles to see if there is any dependency on parallel performance. One might expect the effect of false sharing to diminish with an increasing number of input particles if false sharing between particles is the cause, but this is not the case. As particle density increases, the number of radiochemical reactions also increase. The result of this is that the excited radiochemical particles return to water faster, dropping out of the simulation and thereby also boosting performance and tampering with the results of the test. Instead, perhaps one might devise several input geometries with a constant particle density. As the number of input particles increase, so does the volume of the input so that the particle density remains constant. For a given number of processors, the time to solution should be linear with respect to the number of input particles. If false sharing has an effect on performance, then the effect should diminish with increasing number of particles.

REFERENCES

[1] A. C. Society, "American Cancer Society. Cancer Facts & Figures 2014," Atlanta, 2014.

[2] J. Wilson, "CNN Health," CNN, 11 March 2014. [Online]. Available: http://www.cnn.com/2014/03/11/health/cancer-care-asco-report/. [Accessed 16 March 2014].

[3] H. Y., Microdosimetric Response of Physical and Biological Systems to Low and High LET Radiations, Theory and Applications to Dosimetry, Elsevier, 2006.

[4] M. Karamitros, "Modeling Radiation Chemistry in the Geant4 Toolkit," *Progress in Nuclear Science and Technology,* pp. 503-508, 2011.

[5] Karamitros M., Mantero A., Incerti S., Luan S., Tran H. N., Champion C., Allison J., Baldacchino G., Davidkova M., Friedland W., Ivantchenko V., Ivantchenko A., Nieminem P., Stepan V., "Diffusion-controlled reactions modeling in Geant4-DNA," *Journal of Computational Physics,* vol. 274, pp. 841-882, 2014.

[6] S. Chauvie, Z. Francis, S. Guatelli, S. Incerti, B. Mascialino, P. Moretto, P. Niemainem, M. G. Pia, "Geant4 physics processes for microdosimetry simulation: design foundadtion and implementation of the first set of models," *IEEE Trans. Nucl. Sci.,* vol. 54, no. 6-2, pp. 2619-2628, 2007.

[7] S. Incerti, G. Baldacchino, M. Bernal, R. Capra, C. Champion, Z. Francis, S. Guatelli, P. Guèye, A. Mantero, B. Mascialino, P. Moretto, P. Nieminen, A. Rosenfeld, C. Villagrasa and C. Zacharatou, "The Geant4-DNA Project," *Int. J. Model. Simul. Sci. Comput.,* vol. 1, pp. 157-178, 2010.

[8] "FLUKA; A fully integrated Monte Carlo simulation package. (http://www.fluka.org)".

[9] "MCNP; A General Monte Carlo N-particle transport code (http://mcnp-green.lanl.gov)".

[10] Agarwal, Pankaj; Kaplan, Haim; Sharir, Micha, "Kinetic and Dynamic Data Structures for Closest Pair and All Nearest Neighbors," *ACM Transactions on Algorithms,* vol. 5, no. 1, 2008.

[11] Basch J., Guibas L. J.; C., Silverstein; L., Zhang, "A practical evaluation of kinetic data structures," in *13th Symposium of Computational Geometry*, 1997.

[12] J., Basch; Guibas, L. J.; J., Hershberger, "Data Structures for Mobile Data," in *8th Symposium on Discrete Algorithms,* 1997.

[13] Dinesh Mehta, Sartaj Sahni, "Ch. 23 Kinetic Data Structures," in *Handbook of Dtaa Structures and Applications,* Chapman & Hall, 2004.

[14] Hendee W. R., Ibbott G. S., Hendee E. G., Radiation Therapy Physics 3rd ed., Wiley. John & Sons Inc., 2004.

[15] K. F. M., The Physics of Radiation Therapy 4th ed., Lippincott Williams & Wilkins, 2009.

[16] H. E. J., Radiobiology for the Radiologist 5th ed., Lippincott Williams & Wilkins, 2000.

[17] F. H. Attix, Introduction to Radiological Physics and Radiation Dosimetry, Wiley-VCH, 1986.

[18] R. Wilson, "Radiological use of fast protons," *Radiology*, vol. 47, p. 487, 1946.

[19] D. E. Bezak, "South Australian Medical Heritage Society Inc," [Online]. Available: http://samhs.org.au/Virtual%20Museum/xrays/Braggs-peak-rxth/braggpeakrxth.htm. [Accessed 16 8 2015].

[20] Ziegler J. F., Ziegler M. D., Biersack J. P., "SRIM Stopping Range of Ions in Matter," SRIM.com, 1984.

[21] R. S.A., Diffusion Limited Reactions (Vol. 25 in Chemical Kinetics edited by Bamford CH, Tipper CFH and Compton RG), Elsevier, 1985.

[22] Bloom S., Luan S., Karamitros M., Incerti S., "Geometric Algorithms and Data Structures for Simulating Diffusion Limited Reactions," in *Symposium on the Theory of Modeling and Simulation*, Tampa, 2014.

[23] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E., "Equation of State Calculations by Fast Computing Machines," *Journal of Chemical Physics,* 1953.

[24] J. Lucido, "Incorporating Microdosimetry into Radiation Therapy Treatment Planning with Multi-Scale Monte Carlo Simulations," 2013.

[25] D. T. Gillespie, "Exact Stochastic Simulationof Coupled Chemical Reactions," *Journ. of Phys. Chem.,* vol. 81, no. 25, pp. 2340-2361, 1977.

[26] e. a. Friedland W., "Track structures, DNA structures and radiation effects in the biophysical Monte Carlo simulation code PARTRAC," *Mutat. Res. : Fundam. Mol. Mech. Mutagen.,* 03 01 2011.

[27] e. a. Ballarini F., "Stochastic aspects and uncertainties in the prechemical and chemical stages of electron tracks in liquid water: a quantitative analysis based on Monte Carlo simulations," *Radiat. Environ. Biophys.,* vol. 39, no. 3, pp. 179-188, 2009.

[28] M. V., B. M. and B. E. A., "Computer aided stochastic modeling of the radiolysis of liquid water," *Radiat. Res.,* vol. 149, no. 3, pp. 224-236, 1998.

[29] Kreipl M. S., Friedland W., Paretzke H. G., "Time- and space-resolved Monte Carlo study of water radiolysis for photon, electron and ion irradiation," *Radiat. Environ. Biophys.,* vol. 48, no. 1, pp. 11-20, 2009.

[30] S. Uehara and H. Nikjoo, "Monte carlo simulation of water radiolysis for low-energy charged particles," *J. of Radiat. Res.,* vol. 47, no. 1, pp. 69-81, 2006.

[31] Friedland W., Jacob P., Paretzke H.G., Merzagora M., Ottolenghi A., "Simulation of DNA fragment distributions after irradiation with photons.," *Radiat. Environ. Biophys.,* vol. 38, pp. 39-47, 1999.

[32] J. L. B. R. A. F. J. H. Freidman, "An Algorithm for Finding Best Matches in Logarithmic Expected Time," *ACM Transactions on Mathematical Software,* vol. 3, pp. 209-226, 1977.

[33] Chauvie S., Francis Z., GuatelliS., Incerti S., Mascialino B., Montarou P., Moretto P., Nieman P., Pia M., "Models of biological effects of radiation in the Geant4 Toolkit," *IEEE Nuclear Science Symposium Conference Record,* pp. 803-805, 2006.

[34] Nellis G., Klein S., "Mass Transfer," in *Heat Transfer,* Cambridge, Cambridge University Press, 2012, pp. 974-978.

[35] Socolofsky S., Jirka G., "Concepts, Definitions, and the Diffusion Equation," in *Special Topics in Mixing and Transport Processes in the Environment,* College Station, Texas A&M University, 2005, pp. 1-19.

[36] Fischer, H. B., List, E. G., Koh, R. C. Y., Imberger, J., Brooks, N. H., Mixing in Inland and Coastal Waters, New York: Academic Press, 1979.

[37] S. Khuller and M. Tossi, "A Simple Randomized Seive Algorithm for the Closest Pair Problem," *Information and Computation,* no. 118, pp. 34-37, 1995.

[38] K. H. a. S. M. Agarwal PK, " Kinetic and Dynamic Data Structures for Closest Pairs and All Nearest Neighbors.," *ACM Transactions on Algorithms.,* 2007.

[39] C. J., Stochastic Processes, 2007.

[40] Jon S. Horne, Edward O. Garton, Stephen M. Krone, Jesse S. Lewis, "Analyzing Animal Movements Using Brownian Bridges," *Ecology,* vol. 88, no. 9, pp. 2354-2363, 2007.

[41] N. Privault, "Brownian Motion and Stochastic Calculus," in *Stochastic Finance: An Introduction with Market Examples,* Chapman and Hall/CRC, 2013.

[42] R. N. Hamm, J. E. Turner and M. G. Stabin, "Monte Carlo simulation of diffusion and reaction in water radiolysis - a study of reactant 'jump through' and jump distances," *Radiant Environ. Biophys.,* vol. 36, pp. 229-234, 1998.

[43] C. J. O'Keeffe, Ruichao Ren, G. Orkoulas, "Spatial updating grand canonical Monte Carlo alg. for fluid sim.: generalization to contin. pot. and par. impl.," *Journ. Chem. Phys.,* vol. 127, 2007.

[44] Procassini R. J., O'Brien, M. J., Taylor J M, "Load Balancing of Parallel Monte Carlo Transport Calculations," in *International Topical Meeting on Mathematics and Computations,* Avignon, 2005.

[45] D. M., "An optimal parallel dictionary.," *SPAA '89 Proceedings of the first annual ACM symposium on Parallel algorithms and architectures,* pp. 360-368, 1989.

[46] Karlin A., Upfal E., "Parallel Hashing, an Efficient Implementation of Shared Memory," *J. ACM*, vol. 4, pp. 876-892, 1988.

[47] Mattias Y., Viskin U., "On parallel hashing and integer sorting," *J. Algorithms,* vol. 12, no. 4, pp. 573-606, 1991.

[48] M. F., "Hashing strategies for simulating shared memory on distributed memory machines," in *Parallel Architectures and Their Efficient Use, Lecture notes in Computer Science*, 1993, pp. 20-29.

[49] A. DAF, Efficient Hash Tables on the GPU, Ph. D. Dissertation, UC Davis, 2011.

[50] Alcantara D., Sharif A., Abbasinejad F., Sengupta S., Mitzenmacher M., "Real-time parallel hashing on the GPU," *ACM Transactions on Graphics* , vol. 28, no. 5, 2009.

[51] Ruichao Ren, G. Orkoulas, "Parallel Markov chain Monte Carlo simulations," *Journ. Chem. Phys.*, no. 126, 2007.

[52] G. T. Barkema, T. MacFarland , "Parallel simulation of the Ising model," *Phys. Rev. E*, vol. 50, p. 1623, 1994.

[53] Brodal GS, Traff JL, Zaroliagis CD, "A Parallel Priority Queue with Constant Time Operations," *Journal of Parallel and Distributed Computing,* vol. 49, no. 1, pp. 4-21, 1998.

[54] Chen DZ, Hu XS, "Fast and Efficient Operations on Parallel Priority Queues," *Algorithms and Computation, Lecture Notes in Computer Science,* vol. 834, pp. 279-287, 1994.

[55] Das HK, Horng WB, "Managing a parallel heap efficiently," *Parallel Architectures and Languages Europe Lecture Notes in Computer Science,* vol. 505, pp. 270-287, 1991.

[56] Deo N., Prasad S., "Parallel heap, an optimal parallel priority queue," *J. Supercomputing,* vol. 6, no. 1, pp. 87-98, 1992.

[57] P. S., "Parallel heap. A practical priority queue for fine to medium grained applications on small multiprocessor," in *IEEE Symposium on Parallel and Distributed Computing,* 1995.

[58] S. P., "Randomized Priority Queue for Fast Parallel Access," *Journal of Parallel and Distributed Computing,* vol. 49, no. 1, pp. 86-97, 1998.

[59] Aragon, C., Seidel, R., "Randomized Search Trees," IEEE, Berkley, 1989.

[60] Google, "http://goog-perftools.sourceforge.net/doc/tcmalloc.html," Google, [Online]. Available: http://goog-perftools.sourceforge.net/doc/tcmalloc.html. [Accessed 20 April 2015].

[61] J. J. Wilkens, U. Oelfke, "Analytical linear energy transfer calculations for proton therapy," *Medical Physics,* vol. 30, no. 5, p. 806, 2003.

[62] Bloom S., Luan S., "Geometric Algorithms and Data Structures for Parallelizing Simulations of Diffusion Limited Reactions," *Not yet submitted for publication..*

[63] "Cancer Statistics," *American Cancer Society,* 2004.

[64] Frongillo Y., Goulet T., Fraser M. J., Cobut V., Patau J. P., Jay-Gerin J. P., "Monte Carlo simulation of fast electron and proton tracks in liquid water-II. Nonhomogeneous chemistry," *Radiation Physics and Chemistry,* vol. 51, no. 3, pp. 245-254, 1998.

[65] Preparata F. P., Shamos M. I., Computational Geometry, an Introduction, 1985.

[66] Nellis G., Klein S., "Ch 9. Mass Transfer," in *Heat Transfer*, Cambridge , Cambridge University Press, 2012, pp. 974-978.

[67] U. o. Cincinati. [Online]. Available: cmap.ucfilespace.uc.edu. [Accessed 16 8 2015].

[68] Agarwal PK, Edelsbrunner H, Schwarzkopf O, and Welzl E., "Euclidean minimum spanning trees and bichromatic closest pairs.," *Proceedings of the sixth annual symposium on Computational geometry,* pp. 203-210, 1990.

[69] Abello J, Pardalos PM, and Resende MG., Handbook of Massive Data Sets., Norwell Mass: Kluwer Academic Publishers, 2002.

[70] Agostinelli. S, "Geant4 - a simulation toolkit.," *Nuclear Instruments and Methods in Physics Research,* vol. A, no. 506, pp. 250-303, 2003.