Investigation of the Cell Labeling Procedure and the Appearance of Monozygotic Twins

by Carol M. Jim

B.A. in Mathematics, May 2006, Hood College
M.S. in Computer Science, May 2010, Hood College

A Dissertation submitted to

The Faculty of
School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

August 31, 2015

Dissertation directed by

Simon Y. Berkovich
Professor of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Carol Maggie Jim has passed the Final Examination for the degree of Doctor of Philosophy as of July 28, 2015. This is the final and approved form of the dissertation.

Investigation of the Cell Labeling Procedure and the Appearance of Monozygotic Twins

Carol M. Jim

Dissertation Research Committee:

Simon Y. Berkovich, Professor of Engineering and Applied Science, Dissertation Director

Abdou S. Youssef, Professor of Engineering and Applied Science, Committee Member

Nan Zhang, Associate Professor of Computer Science, Committee Member

Alexander Spector, Research Professor, Johns Hopkins University, Committee Member

© Copyright 2015 by Carol M. Jim
All rights reserved.

Acknowledgments

I would like to express my deepest gratitude and thanks to my dissertation director, Dr. Simon Berkovich, for all of his guidance, help, advice, and patience throughout my research. I sincerely appreciate his continued support and assistance even after his retirement. I would also like to thank the members of my committee, Dr. Abdou Youssef, Dr. Nan Zhang, and Dr. Alexander Spector, for their time spent serving as my committee members and for their constructive feedback and comments. Thanks also go out to Dr. Xiuzhen Cheng for her time and feedback in reviewing my dissertation proposal.

A very special thanks goes out to Dr. Ahmed Salem, my former thesis advisor when I was pursuing my Master's degree, and now, my colleague and my brother. He has always been there for me through both the good and the tough times and constantly provides me with unwavering and never-ending support and encouragement. It was him who motivated and encouraged me to pursue a Ph.D. degree in the first place. For everything, I will be eternally grateful.

Finally, I would like to thank my family for all of their love, encouragement, and support they have given me throughout my entire life. Words cannot express how grateful I am to my mom, dad, and brother. Without them, I would not be where I am today, and this Ph.D. would not have been possible.

Abstract of Dissertation

Investigation of the Cell Labeling Procedure and the Appearance of Monozygotic Twins

The origins of pairs of monozygotic twins and higher order multiples, i.e. triplets, quadruplets, etc., have been extensively studied but still little is understood. To gain insight into this event, certain possible cell labeling schemes that model an organism's development are analyzed. The phenomenon of quadruplet twins is exposed during the process. We predict that monozygotic quadruplets are not really quadruplets but instead are two pairs of monozygotic twins where the pairs slightly differ. From the considered models, the probability of monozygotic twins is found to be $(1/2)^K$, and we discover from our analysis that the probability of monozygotic quadruplets, or triplets in the case of the death of an embryo, is $(1/8)^K$, where K is a species-specific integer representing the number of pairs of homologous chromosomes. This investigation into twinning provides a foundation for understanding the process of cell development through which the cell development mechanism is established. The failure of the internal cellular clock from this mechanism may play an important role in cancerogenesis. The parameter K may determine cancerization with a probability threshold that is approximately inversely proportional to the Hayflick limit, so exposure to small levels of ionizing radiation and chemical pollution may not produce cancer.

Table of Contents

Acknowledgments ..... **iv**

Abstract of Dissertation ..... **v**

List of Figures ..... **viii**

Chapter 1: Introduction **1**

Chapter 2: Literature Review **5**

2.1 Mechanisms and Etiology of Monozygotic Twinning 5

2.2 Estimating Frequency of Monozygotic Twinning 8

2.2.1 Hellin's Law 9

2.2.2 Weinberg's Differential Rule 10

2.2.3 Bonnier's Equation 12

2.2.4 Berkovich's Hypothesis 12

Chapter 3: Model Design **15**

3.1 Possible Cell Labeling Models 15

3.1.1 Binary Cell Labeling Scheme 15

3.1.2 Chromosomes Labeling Scheme 17

3.1.2.1 Model Analysis 17

3.1.2.2 Generating the Model Using Python 20

3.1.3 Asymmetric Labeling Scheme 22

3.1.3.1 Model Analysis 23

3.1.3.2 Generating the Model Using Python 24

3.2 Cell Labeling Combinations 26

3.2.1 Likewise and Crosswise Pairings....26
3.2.2 Generating the Combinations Using Python....30
Chapter 4: Results and Discussion....34
4.1 Appearance of Quadruplet Twins as Identical Pairs....35
4.2 Examination of Further Cell Divisions....39
4.2.1 Further Divisions of the Chromosomes Labeling Scheme....40
4.2.2 Further Divisions of the Asymmetric Labeling Scheme....41
4.3 Implications of Telomeres on Aging....44
4.4 Implications of Considered Mechanism of Twinning on Cancerogenesis....47
Chapter 5: Conclusion....51
References....53
Appendix A: Program to Generate Model and Labeling Combinations....64
Appendix B: Raw Sample Results....76
Appendix C: Program to Generate Asymmetric Labeling Model....77

List of Figures

Figure 1-1: Chromosome with telomeres 2

Figure 1-2: Comparison of the formation of monozygotic twins and dizygotic twins 3

Figure 2-1: A human blastocyst 6

Figure 2-2: A human ovum 7

Figure 2-3: Summary of parameter *K* values from *P(MZT)* in various species 13

Figure 3-1: Binary cell labeling scheme 16

Figure 3-2: Chromosomes labeling scheme of the cellular clock model 19

Figure 3-3: Pseudocode to generate the chromosomes labeling scheme 21

Figure 3-4: Asymmetric labeling scheme with restorative mechanisms 24

Figure 3-5: Pseudocode to generate the asymmetric labeling scheme 26

Figure 3-6: Two possible cell labeling combinations of (a) likewise combination and (b) crosswise combination 28

Figure 3-7: Cell labeling of possible combinations of homologous chromosomes at the initial zygote division for the chromosomes labeling scheme 29

Figure 3-8: Initial zygote division of the asymmetric labeling scheme 29

Figure 3-9: Pseudocode to generate all possible cell labeling combinations for a specified cell division 32

Figure 4-1: Quadruplet twins appearance in the second cell division of the chromosomes labeling scheme 35

Figure 4-2: Quadruplet twins in the second cell division of the asymmetric labeling scheme 36

Figure 4-3: Octuplet twins appearance in the third cell division of the chromosomes labeling scheme 40

Figure 4-4: Octuplet twins in the third cell division of the asymmetric labeling scheme 42

Chapter 1: Introduction

A multi-cellular organism begins its development from a single cell, which is the fertilized egg or zygote. Numerous binary cell divisions and cell deaths occur forming the organism. The issues of cell development of an evolving organism can be viewed in terms of developmental trees from an abstract mathematical perspective. These development models are useful for understanding the complex process of how a single cell develops into a multi-cellular organism through cell divisions and differentiation. Throughout a multi-cellular organism's development, cell differentiation occurs where sets of cells specialize to have completely different functions, thus forming various tissues and cell types.

In the developmental tree, the zygote can be thought of as the root of the binary tree since it is the earliest stage of a multi-cellular organism's development whereas the cells of an organism are nodes in the tree. Pairs of branches along with corresponding internal nodes represent a cell division of the preceding cell. Labeling the developmental tree is the process of differentiating between cells. Tracing a path from the root to a specific cell in the tree reveals the history of its divisions.

Throughout a cell's lifetime, Leonard Hayflick [1] demonstrated that normal human fetal cells will divide approximately 40 to 60 times before cell division halts, a phenomenon known as the Hayflick limit. This is because Alexey Olovnikov [2] suggested that telomeres, the ends of eukaryotic chromosomes as illustrated in Figure 1-1, slightly shorten with each cell division until a critical length is reached. In humans, telomere shortening eventually causes cell division to cease which is a key indicator of aging. Aside from aging, telomeres have also been implicated in cancer research. Thus,

Hayflick's discovery promoted the idea of the internal cellular clock, a counting mechanism responsible for cell death and the control of cell differentiation. If the internal cellular clock fails, the cell may become cancerous. We may learn more about the structure of the internal counting mechanism by studying the probability of monozygotic twinning (MZT).

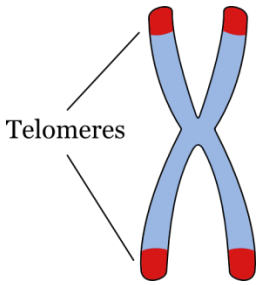

Figure 1-1: Chromosome with telomeres.

Monozygotic twins derive from a single zygote that divides into two separate cell masses early in the developmental stage, each giving rise to an independent individual. This is different from dizygotic twins where two zygotes develop at the same time into two individuals. Figure 1-2 illustrates the difference between the derivation of monozygotic twins and dizygotic twins. Dizygotic twins can either be of the same sex or different sexes since they share about 50% of their genetic information, which is similar to singleton siblings. Since monozygotic twins carry the same genetic information, they are usually the same sex. There are, however, a few extremely rare cases [3, 4, 5] where monozygotic twin siblings are of different sexes. This may occur in a set of identical twin boys where one of the twins loses a Y chromosome during a cell division resulting in a girl with an XO chromosome having Turner's syndrome. A case is also reported in

monozygotic triplets [6] where one of the three identical triplets is a female because of the lost Y chromosome.

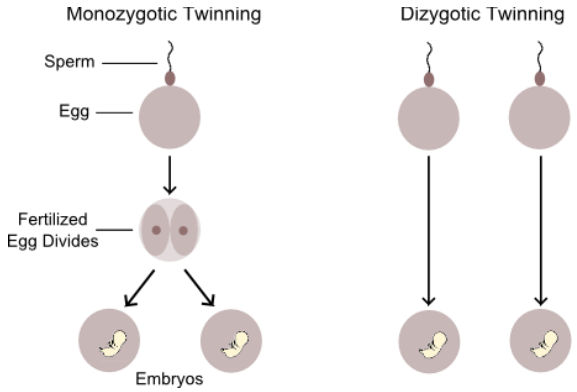

Figure 1-2: Comparison of the formation of monozygotic twins and dizygotic twins.

It is unclear as to why some zygotes split whereas others do not. However, observations suggest that MZT in humans is more heavily dependent on genetic mechanisms whereas environmental factors influence dizygotic twinning (DZT) [7]. MZT is a seemingly random event and occurs at a constant frequency in all populations throughout the world. Of all naturally conceived twinning events, MZT occurs less often than DZT with MZT only occurring approximately 30% of the time while DZT occurs approximately 70% of the time.

Little is known about the origins of pairs of monozygotic twins and higher order multiples such as monozygotic triplets, quadruplets, etc. in both humans and animals. In this dissertation, we provide some insight into the mechanisms involved in twinning by way of cell division labeling. In addition to twinning, these same mechanisms may provide some implications into cancerogenesis. The significance of telomeres on the human aging process is also considered. Both the aging process and cancerogenesis have been noteworthy topics under extensive study currently. The reasons why humans age and the cause of aging have been questioned but no concrete answers exist. Likewise, the

process of certain cells becoming cancerous and methods to halt this process are still being investigated. From our considered models, the mechanisms involved in the two opposite circumstances of twinning and cancerogenesis may provide a foundation for the understanding of the origins of these two disparate processes.

The remainder of the dissertation is organized as follows. Chapter 2 provides a review of the mechanisms involved in twinning along with researched statistical frequencies of MZT in various animals and in plants. Chapter 3 provides an analysis of possible cell division labeling models and our implementations of two current models in the Python programming language to investigate this problem. In addition, the possible cell labeling combinations that result from our models are also examined. Chapter 4 includes a discussion of primary results that can be deduced from our models. Particularly, the appearance of quadruplet twins in the second cell division along with an examination of results from further divisions is investigated. Suggestions for experimental verification are also examined. Furthermore, we discuss the implications that telomeres as modeled in the considered cell labeling schemes have on aging and the implications of the proposed twinning mechanism on cancerogenesis. Finally, a summary of our findings can be found in Chapter 5.

Chapter 2: Literature Review

2.1 Mechanisms and Etiology of Monozygotic Twinning

The mechanisms that affect the early stages of a zygote's development leading to the splitting of the zygote in MZT are still unknown. On the contrary, the causes of spontaneous DZT are better understood. The mechanisms leading to spontaneous DZT function on multiple ovulation and an increased level of follicle-stimulating hormone [8, 9]. Maternal age, the previous number of pregnancies, ethnicity, genetic inheritance, and body composition are all factors affecting DZT rates [7, 9, 10]. An increase in maternal age peaking at around age 37, an increase in the number of previous pregnancies, and tall women all increase the chance of DZT. It has also been observed that women of African descent have the highest DZT rates with a rate of more than 18 twins per 1,000 births in Central Africa whereas the lowest rates are seen in Asia and Latin America with a rate that is less than 8 twins per 1,000 births [10]. The variation in twinning rates among ethnicities is prevalent mostly due to DZT occurrences since MZT rates are relatively constant worldwide.

The randomness and rarity of MZT events makes it difficult to determine its etiology but some proposed mechanisms exist in an attempt to explain this phenomenon. Blickstein [11] reviews a few theories about the mechanism of zygotic splitting. We provide a comprehensive review of the proposed mechanisms found in the literature hereafter. With the exception of Theory #4, the presented theories may apply to spontaneous, i.e. without the use of assisted reproductive technologies, MZT. However, we still present the theory which proposes a mechanism for twinning when in-vitro fertilization (IVF) is used for the sake of completeness. It should be noted that

monozygotic twins resulting from assisted reproductive technologies may be influenced by different mechanisms than spontaneously occurring monozygotic twins. The debate regarding the mechanisms and timing of MZT is still alive and well today as seen in the recent publications [12, 13, 14, 15].

Theory #1: Every embryo develops along an axis giving it a bilateral symmetry. In the case of the development of more than one axis, the dominant axis causes other axes to degenerate. If two axes co-dominate, then MZT results [16].

Theory #2: In the early stages of a zygote's development, cells within the blastocyst, i.e. a structure formed in early development that contains an inner cell mass which forms the embryo as depicted in Figure 2-1 (adapted from [17]), exhibit genetic differences and become discordant such that they separate into two different cell masses [8, 18, 19]. The two cell masses develop into two distinct embryos. The genetic differences may occur due to the development of a chromosome abnormality, a gene mutation, or some change in the gene control processes. An example of the latter is DNA methylation which is involved in the control of gene expression during cell differentiation.

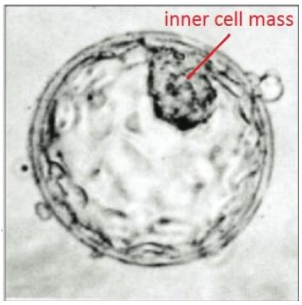

Figure 2-1: A human blastocyst.

Theory #3: Skewed X-chromosome inactivation may play a role in the twinning process for female monozygotic twins [8]. In the process of X-chromosome inactivation, one of the two female's X chromosomes becomes inactivated. Usually, it is a random choice on which X chromosome gets inactivated, but sometimes, the choice is non-random or skewed for certain tissues or organs. This may lead to various X-linked recessive diseases in the female twin.

Theory #4: During micromanipulation in the IVF process, the zona pellucida, i.e. a membrane that surrounds the developing egg as shown in Figure 2-2 (adapted from [20]), may be breached or punctured leading to a herniation of the blastocyst. The inner cell mass may then separate into two masses, each developing into an embryo. The study in [21] shows that each reported case of monozygotic twin pregnancies resulted from a breach in the zona pellucida or a naturally thin zona pellucida.

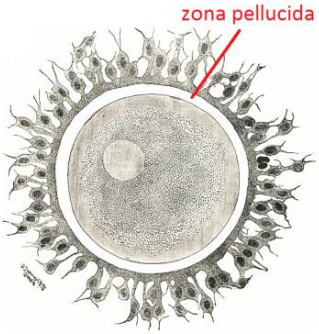

Figure 2-2: A human ovum.

Theory #5: The blastocyst collapses splitting the progenitor cells, i.e. cells that contain fundamental genetic material and have yet to differentiate into their target cells, in half into two distinct inner cell masses on opposite sides of the embryo forming monozygotic twins [22].

Even with all of these theories, the true mechanism and etiology behind MZT is still unknown. Furthermore, there have been some criticisms with some of the mentioned theories. Skewed X-chromosome inactivation (Theory #3) seems to be more responsible for explaining the predominance of female monozygotic twins rather than providing logic behind the mechanism of MZT. The study in [23] observed no significant difference in the degree of skewing between female monozygotic twin pairs and female singletons suggesting that skewed X-chromosome inactivation does not directly influence the mechanism of twin formation. As for the breach of the zona pellucida following micromanipulation in the IVF process (Theory #4), several published studies [24, 25, 26] involving pregnancies resulting from zona pellucida micromanipulation disproved this theory. All of the theories propose to explain the mechanism of MZT, however, they are unsuitable to explain the mechanism behind the formation of monozygotic triplets, quadruplets, and other higher order multiples. The lack of animal models is a major limiting factor in uncovering the mysteries behind the mechanism of monozygotic splitting.

2.2 Estimating Frequency of Monozygotic Twinning

A fundamental problem in the research on twinning is estimating the frequency of twinning occurrences. Because of the rarity and randomness of the occurrences, especially for higher order multiples, it is difficult to formulate a generic estimate for twinning rate. However, some laws and methods exist that provide a reasonably accurate estimate for twinning in general and for both MZT and DZT.

2.2.1 Hellin's Law

One of the main contributors of a formally defined statistical approximation for multiple maternities is from Hellin [27] resulting in Hellin's law. Although it was Strassmann [28] who was the first to present Hellin's law, Hellin generalized Strassmann's findings. Hellin's law states that

$$P(x\text{-tuplets}) \approx \frac{1}{89^{x-1}} \qquad (2.1)$$

Therefore, for humans, the probability of having twins is about 1 in 89 pregnancies, triplets is about 1 in 89^2 pregnancies, quadruplets is about 1 in 89^3 pregnancies, and etc. Hellin's law is only valid when fertilization occurs naturally and not when assisted reproductive technology procedures such as IVF are used to conceive. There is a difference in how Hellin and Strassmann presented the law. Hellin relates the frequencies of multiple births with respect to singleton births whereas Strassmann relates the frequencies of multiple births with respect to all births, which is commonly what we follow today [29]. However, the same probabilistic approximation is used in both cases.

Hellin's law is appealing because of its mathematical simplicity. Even though the mathematical relation is approximately correct especially in large data sets where averages in multiple maternity rates can be stable, there are discrepancies that are hard to explain. Fellman and Eriksson [30] attempt to identify and explain these discrepancies with respect to Hellin's law. Sometimes the rates of higher order multiples are either too high or too low when Hellin's law is used as an approximation. In addition, many statistical studies are based on empirical rates rather than theoretical rates which are represented by the law. Thus, these studies cannot prove the correctness of the law but

can only serve to recognize errors that cannot possibly be considered as random. Hellin's law should serve only as an approximation of the probability of multiple natural maternities in humans since it cannot hold as a general rule as seen in the mathematical proof in [31]. It should also be noted that Hellin's law applies for all natural births that include both monozygotic and dizygotic multiples.

2.2.2 Weinberg's Differential Rule

In 1901, the German physician-geneticist Wilhelm Weinberg [32] proposed another method to estimate the frequency of monozygotic versus dizygotic twin pairs to the total twinning rate. His method is based on two assumptions: (1) there is an equal probability of having a male and a female, and (2) the sexes in a dizygotic twin pair are independently determined, i.e. dizygotic twins are half the opposite sex (50% male-female/female-male) and half the same sex (25% male-male and 25% female-female). Hence, Weinberg's differential rule states that if we assume that n_1 = number of same sex twins and n_2 = number of opposite sex twins, then the number of monozygotic (MZ) and dizygotic (DZ) twins can be estimated as $MZ = n_1 - n_2$ and $DZ = 2n_2$, respectively.

Assuming that all monozygotic twins must be of the same sex and twins that are the opposite sex must be dizygotic, the excess of same sex twin pairs from both monozygotic and dizygotic twins can be used to estimate the frequency of MZT. Therefore, Weinberg's differential rule can also be written as

$$MZ = n - 2n_2 \qquad (2.2)$$

where n = total number of twin pairs and n_2 is defined as above. Unlike Hellin's law where the approximation of twinning rate is only for human populations, Weinberg's differential rule has also been widely used in approximating MZT rates in dairy cattle. When used for this purpose, the MZT frequency is often expressed as

$$MZ = \frac{n - 2n_2}{n_1} \qquad (2.3)$$

where n, n_1, and n_2 are all defined above.

There are some flaws in the assumptions presumed in Weinberg's differential rule that leads us to question its validity. The first assumption on having an equal probability of having a male and a female is not entirely true. The second assumption on dizygotic twins being half the opposite sex and half the same sex is not very accurate since it has been observed that the frequency of same sex pairs is higher than opposite sex pairs in samples of dizygotic twins [33, 34]. This would suggest that Weinberg's differential rule underestimates the total number of dizygotic twins and overestimates the total number of monozygotic twins. In addition, although extremely rare, a few cases have been reported where monozygotic twins are of the opposite sex [3, 4, 5]. Despite the presented flaws, there have been arguments supporting the validity and reliability of Weinberg's differential rule [35, 36]. The accuracy of the rule may be dependent on the country and time period from where the samples of twinning data are used.

2.2.3 Bonnier's Equation

Rather than following the first assumption of an equal sex ratio of 50:50 of male to female, Gert Bonnier [37] modified Weinberg's differential rule to use the observed sex ratios in the twinning samples. Thus, Bonnier estimates the MZT frequency to be

$$MZ = \frac{2pqn - n_2}{2pq(n - n_2)} \tag{2.4}$$

where p = observed frequency of males in the twin sample, q = observed frequency of females in the twin sample, and n and n_2 are defined above. The gain in accuracy by using the observed sex ratios, however, is minimal unless the sex ratio deviates far from 50:50 [38].

2.2.4 Berkovich's Hypothesis

By modeling the cell development of an organism, Simon Berkovich [39] formulated a hypothesis for the probability of MZT in humans and animals, and consequently, may also be applied to plants. Berkovich's hypothesis states that the probability of MZT, *P(MZT)*, is

$$P(MZT) = \left(\frac{1}{2}\right)^K \tag{2.5}$$

where K is a species-specific integer representing the number of pairs of homologous chromosomes.

Much of the data on MZT frequency in various species support Berkovich's hypothesis. The frequency of MZT in humans is consistent at about 1 in 250 to 300

births [40, 41], which is close to $(1/2)^8$. MZT data in animals is available, though scant. Berkovich and Bloom [42] and Berkovich [43] provide a comprehensive review of data on MZT frequency. Figure 2-3 summarizes the parameter *K* values found for various species mentioned both in the review and henceforth. In mice, it is found that MZT occurs at a frequency with 95% confidence limits of 0.2% and 2.6% [44], which gives a midrange value of 1.4%. This midrange value is close to $(1/2)^6$, which is approximately 1.56%. A study on twinning and mutation rate in pigs [45] shows that of the sampled population, MZT in pigs occurs at a rate of 1 in 525 offspring, which is close to $(1/2)^9$. In a study on dairy cattle [46], data from two farms report a MZT frequency of 5.5% among all twin births and 0.33% per birth. This observed MZT frequency of 0.33% by calving event is relatively close to the frequency observed for humans at about 0.4%, which means that the probability of twinning in cattle is close to $(1/2)^8$.

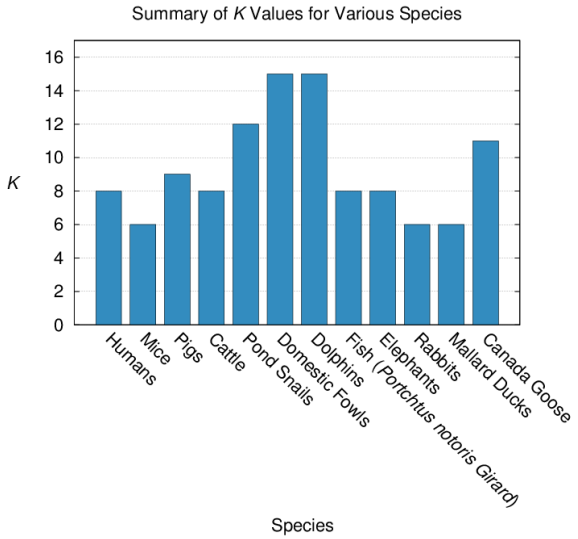

Figure 2-3: Summary of parameter *K* values from *P(MZT)* in various species.

MZT in plants is more complex than in animals. Plants may either undergo asexual reproduction or sexual reproduction as in the case of some flowering and seed plants.

Like sperms and eggs in animals, a plant's male and female sex cells consist of pollen and ovules. Polyembryony, i.e. the phenomenon of multiple embryos developing from a single seed or fertilized egg, is a common occurrence in some plants whilst uncommon in others and can be seen when two seedlings emerge from one seed. Similarly in animals, little is known about the origins and causes of polyembryony in plants. In the reproductive strategy of plant monozygotic polyembryony, it has been observed that, usually, only one embryo in a polyembryonic seed survives while other subordinate embryos are eliminated via the mechanism of programmed cell death [47]. This mechanism ensures that one embryo survives by eliminating competition among the monozygotic embryos. In tobacco plants, Chen and Gallie [48] discovered that vitamin C promotes the production of monozygotic twins and triplets from a single seed. Like MZT in humans, their study shows that it is the first division of the zygote that results in two identical seedlings in plants. The researchers predict that vitamin C can produce twins and triplets in other plant species besides tobacco plants.

Data on MZT frequency in plants is less readily available. Development of plant embryos differ in many respects as compared to animals. In many cases, experimentally induced twinning by in-vitro methods are used to study twinning in plants. However, Berkovich [39] counted the number of seeds inside the nutshell of almonds and determined that pairs, i.e. two seeds inside a nutshell, occurred in about 1 in 16 cases, which corresponds to $(1/2)^4$. Although there may be a different mechanism in plants, the proportion of pairs as seen in almonds still support the hypothesis of the MZT probability of $(1/2)^K$.

Chapter 3: Model Design

3.1 Possible Cell Labeling Models

In order to depict the cell development of an evolving organism, we can use cell labeling schemes of a developmental tree. Three possible cell labeling models of increasing complexity are considered: the binary cell labeling scheme, the chromosomes labeling scheme, and the asymmetric labeling scheme. In the binary cell labeling scheme, we begin considering the concept of telomeres where new cells that are formed are not always the same. The labeling is a straightforward process. The chromosomes labeling scheme is an expansion of the binary cell labeling scheme where the semiconservative process of DNA replication is taken into account. The asymmetric labeling scheme is modified from the chromosomes labeling scheme where possible restorative mechanisms are considered. We derive the possible cell labeling combinations for both the chromosomes labeling scheme and the asymmetric labeling scheme and conduct our investigation into the mechanisms involved in twinning and cancerogenesis.

3.1.1 Binary Cell Labeling Scheme

When applied to biological systems, development models need to show how cell differentiation occurs where a single cell will eventually produce multiple cells that differ in its functionality after self-reproduction. Each cell divides into two individual cells with both of these new cells not necessarily being identical, thus creating a binary tree. The binary cell labeling scheme involves labeling the binary tree with 0's and 1's as shown in Figure 3-1, which illustrates the first three cell divisions. Each node in the tree

represents a cell. In this abstract scheme, its materialization by telomeres is considered, hence the binary labeling of the newly created cells. This scheme does not, however, take into consideration any other further complexities of labeling. For example, it does not distinguish between the cells all labeled as "0" and between the cells all labeled as "1". Rather, the scheme uses the sequence of 0's and 1's of the path from the root to a certain cell in the tree to make the distinction between cells. The binary sequence also describes the history of the cell's origin.

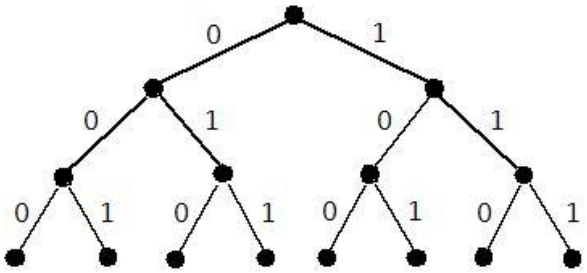

Figure 3-1: Binary cell labeling scheme.

The binary cell labeling scheme has many similarities to Huffman coding in lossless data compression. The method of labeling is analogous to the labeling on a Huffman tree built in Huffman coding where a left branch is labeled as a "0" and a right branch is labeled as a "1". The binary sequence is analogous to the codewords created in Huffman coding where the tree is traversed beginning at the root to a leaf node, and the sequence of 0's and 1's of that path forms the encoding of the respective symbol.

The binary cell labeling scheme illustrates the process of cell differentiation. Berkovich [49] explains that this scheme provides decentralized organization and cell system control for a developing system where control is spread across the entire population. A class of cells comprises of all cells with a common descendent. A class of cells, e.g. cells containing the sequence 10XX…X, may be differentiated into one type

whereas a class of cells within a larger class of cells, e.g. cells containing the sequence 100X…X, may be differentiated into a subtype. The symbol X may represent either a "0" or a "1", and the types and subtypes that the classes of cells differentiate into may become various tissues and cell types such as muscle cells, skin cells, nerve cells, etc.

3.1.2 Chromosomes Labeling Scheme

The chromosomes labeling scheme is one of the considered models in our investigation into twinning mechanisms and cancerogenesis. An analysis of the chromosomes labeling model is first provided followed by our implementation of the model in the Python programming language.

3.1.2.1 Model Analysis

The binary cell labeling scheme can be expanded upon to take into account DNA changes during replication and the asymmetry of cells during an organism's development. DNA carries predetermined instructions that are responsible for the functioning and characteristics of a living organism. Chromosomes are made of up of tightly coiled DNA wrapped around proteins. Before a cell divides, it must replicate, i.e. copy, its chromosomes so that all of these instructions can be passed onto the daughter cells when cell division occurs. During replication, a chromosome's two DNA strands unwind and separate and proceed through a process called semiconservative replication where a complementary strand is produced for each original strand. Telomeres, sequences of DNA at the ends of chromosomes, shorten after each cell division because of the space that is taken up by a short piece of RNA needed to begin the process of

creating the complementary strands. Therefore, the new complementary strands are slightly shorter in length as compared to the original strands.

Berkovich [39] suggests a model of the internal cellular clock that takes into account these chromosomal changes during semiconservative replication and provides a cell labeling mechanism in the process. Figure 3-2, taken from [39], shows this chromosome labeling scheme for the first three divisions. At the root, we have a chromosome in the original cell that contains two original strands of DNA, i.e. the two complementary halves of the double helix structure, labeled as AB. After the first division and the semiconservative replication process, we will get two daughter cells with each cell having a combination of one original DNA strand and one newly synthesized strand. Let us define this combination of one original and one new DNA strand to be a hybrid strand. These daughter cells are formed by the two original strands of DNA labeled as AB separating and a complementary strand is produced for A (which we label as B′) and another complementary strand is produced for B (which we label as A′). Thus, the two daughter cells will contain DNA labeled as AB′ and A′B, where ′ represents a newly formed complementary strand. The newly formed strands are slightly shorter than the original strands because of telomere shortening after the cell division. Hence, after the first division, both cells contain hybrid strands. The process continues with the hybrid strands separating and a new complementary strand is synthesized for each strand during the second division. These new cells will contain DNA labeled as AB′, A″B′, A′B″, and A′B after the second division. Thus, half of the cells contain hybrid strands whereas the other half contains completely new strands. This process of cell division and labeling continues where every subsequent division will result in fewer hybrid strands and more

completely new strands [50]. As Berkovich [39] mentions, the pattern of labeling is such that if a single strand accumulates i changes, then its complement can only have $(i - 1)$ or $(i + 1)$ changes.

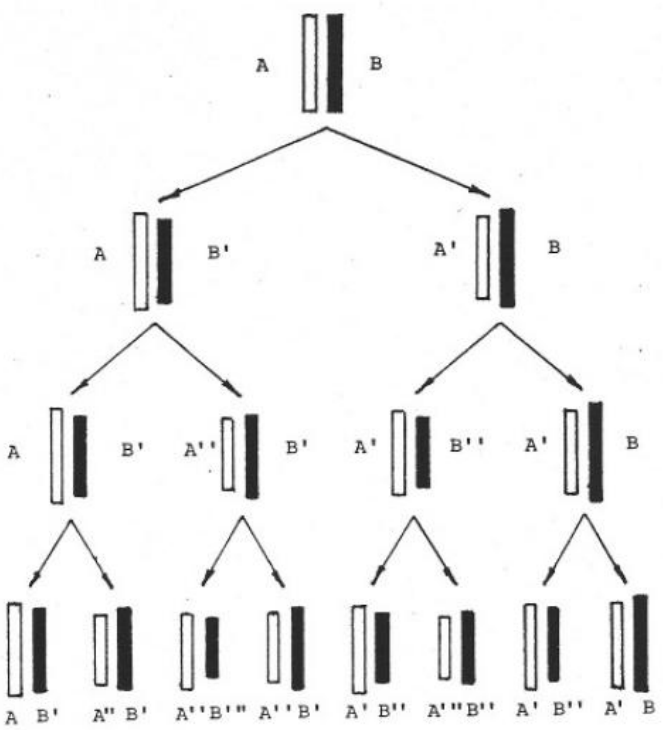

Figure 3-2: Chromosomes labeling scheme of the cellular clock model.

We investigate cell divisions up to and including the fourth division of the chromosomes labeling scheme in our study of the mechanisms involved in twinning and cancerogenesis. The first three divisions are instrumental in the examination of twinning mechanisms in humans and most mammals. The fourth division is not significant for humans since the probabilities of higher order multiples will be more than one in several billions, but it may be worthwhile to investigate for some animals such as fish. Consideration of the fifth cell division and beyond is impractical since twinning occurs in the very early stages of development.

3.1.2.2 Generating the Model Using Python

To formally generate the chromosomes labeling scheme up to any cell division, we use Python 2.7.8 to program the tree using recursion. The source code of the model can be found in Appendix A in its entirety as part of the source code used to generate all possible cell labeling combinations for a user-defined cell division. The main algorithm of the chromosomes labeling model consists of populating the tree to a specified depth. The specified depth must be greater than zero. If it does not satisfy this condition, then the tree does not populate. Once a valid depth is specified, the `populate(node)` function is called where population of the tree begins by checking to determine if both the left child and the right child of the current node exists. If both children of the parent node do not exist, then one of the following three conditions executes:

1. If the current node is the root of the tree, then populate the left child with a Chromosome object labeled as AB' and populate the right child with a Chromosome object labeled as A'B.
2. If the current node is the left child of its parent node, then populate the left child of the current node with a Chromosome object having the same labeling as the current node. In addition, populate the right child of the current node with a Chromosome object having the labeling where A increments by one the prime value of B in the current node and the labeling of B remains the same as its counterpart in the current node.
3. If the current node is the right child of its parent node, then populate the left child of the current node with a Chromosome object where the labeling of A remains the same as its counterpart in the current node and B increments by one the prime

value of A in the current node. In addition, populate the right child of the current node with a Chromosome object having the same labeling as the current node.

The process of populating the nodes of the tree continues recursively until the specified depth is reached. Figure 3-3 displays the pseudocode of the aforementioned main algorithm. The pattern of labeling as seen by the second and third conditions is prevalent throughout each of the subtrees in the chromosomes labeling scheme due to the newly synthesized complementary strands of DNA.

```
// Populates a depth of a chromosome tree using recursion.
// node is the root node of the tree to populate.
function populate(node)
    if node.left is null and node.right is null:
        if node is root:
            root.left = Chromosome(A=0, B=1)
            root.right = Chromosome(A=1, B=0)
        else if node is the left child:
            node.left = Chromosome(A=node.A, B=node.B)
            node.right = Chromosome(A=node.B + 1, B=node.B)
        else:
            node.left = Chromosome(A=node.A, B=node.A + 1)
            node.right = Chromosome(A=node.A, B=node.B)
        return
    populate(node.left)
    populate(node.right)
end function

// Populates the chromosome tree to the specified depth.
// depth is the amount of depths to populate the chromosome
// tree.
function populate_to_depth(depth)
    for 0 to depth:
        populate(root)
    end
end function
```

Figure 3-3: Pseudocode to generate the chromosomes labeling scheme.

The Chromosome object provides a representation of the chromosome labeling. It holds information about the labeling of the complementary strands of DNA, A and B, and whether they are newly formed strands, which is represented by a prime value. The

default prime value for both original complementary strands is zero, hence the labeling AB at the root of the tree. Other information in the Chromosome object, both of which are represented by boolean values, include the node's positioning in relation to its parent node and designation if the node is a maternal or paternal chromosome. The node's positioning indicates if the chromosome is the left child of the parent chromosome. Maternal chromosomes are represented by capital letters whereas paternal chromosomes are represented by lowercase letters in their string representations. This designation is important to distinguish between the maternal and paternal chromosomes when examining the cell labeling combinations.

Aside from the main algorithm of populating the tree, other functions include populating a lookup table and iterating over the chromosomes labeling model. The purpose of the lookup table is to provide a method to store each node with their respective children so that the likewise and crosswise pairings can be easily computed when determining all of the possible cell labeling combinations for a specified cell division as seen in Section 3.2. The lookup table is populated by traversing the tree in-order after the chromosomes labeling scheme has been generated to the specified depth. A function is included to perform an in-order traversal of the tree and display its contents to standard output.

3.1.3 Asymmetric Labeling Scheme

The asymmetric labeling scheme is another considered model in our investigation into twinning mechanisms and cancerogenesis. An analysis of the asymmetric labeling

model is first provided followed by our implementation of the model in the Python programming language.

3.1.3.1 Model Analysis

Although the chromosomes labeling scheme takes into account chromosomal changes during the replication process, it does not take into consideration any restorative mechanisms that certain cells may exhibit. Cells such as germ cells and stem cells contain the enzyme telomerase which helps prevent telomere shortening. In certain situations, DNA may not exhibit changes during replication. One such situation is seen in bacteria where Olovnikov [51] explains that because the chromosomes in bacteria are in a circular form, they do not undergo shortening since there are no ends. Another such situation is during certain stages of cell differentiation in germ cells where telomerase activity is present as shown in studies for both humans [52] and rats [53]. This suggests that restorative mechanisms such as telomerase are important during meiosis.

The chromosomes labeling scheme can be modified to account for possible restorative mechanisms in the DNA of cells. Figure 3-4 illustrates the asymmetric labeling scheme with such restorative mechanisms. It is assumed in this model that replicative changes in DNA are asymmetric, meaning that these changes are associated with only one of the pair of complementary strands. This assumption is based upon the ideas of the work in [54] where a general model for cellular differentiation is proposed in which there is a developmental asymmetry between sister cells as a result of DNA replication. In the asymmetric labeling scheme, the left branch of each branch pair reproduces its complement without changes while the right branch of each branch pair

reproduces its complement with changes. For example, in the first division, the DNA strand, A, reproduces its complement, B, without changes; hence the labeling of AB. On the other hand, the DNA strand, B, reproduces its complement, A, with changes; hence the labeling of A′B. Thus, some chromosomes can maintain the entirety of their initial information due to restorative mechanisms.

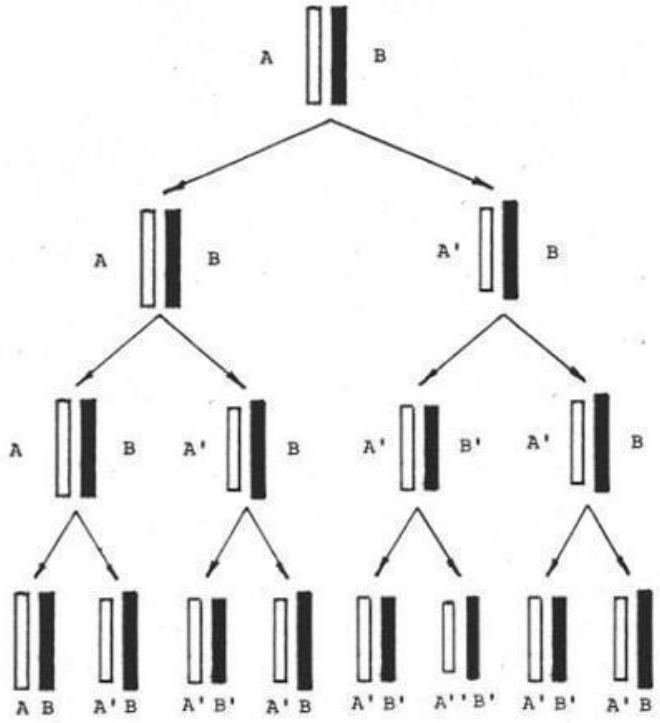

Figure 3-4: Asymmetric labeling scheme with restorative mechanisms.

3.1.3.2 Generating the Model Using Python

The asymmetric labeling scheme is generated similar to the chromosomes labeling scheme but with a few small differences when populating the nodes due to the differences in the labeling. In the asymmetric labeling scheme, one of the following three conditions executes if both children of the parent node do not exist:

1. If the current node is the root of the tree, then populate the left child with a Chromosome object labeled as AB and populate the right child with a Chromosome object labeled as A'B.
2. If the current node is the left child of its parent node, then populate the left child of the current node with a Chromosome object where the labeling of A remains the same as its counterpart in the current node and the labeling of B has the same prime value as A. In addition, populate the right child of the current node with a Chromosome object having the labeling where A increments by one the prime value of B in the current node and the labeling of B remains the same as its counterpart in the current node.
3. If the current node is the right child of its parent node, then populate the left child of the current node with a Chromosome object where the labeling of A remains the same as its counterpart in the current node and the labeling of B has the same prime value as A. In addition, populate the right child of the current node with a Chromosome object having the same labeling as the current node.

Similarly to the chromosomes labeling scheme, the process of populating the nodes of the tree for the asymmetric labeling scheme continues recursively until the specified depth is reached. The pseudocode of the aforementioned main algorithm is shown in Figure 3-5. The pattern of labeling as seen by the second and third conditions is prevalent throughout each of the subtrees in the asymmetric labeling scheme. The left branch of each branch pair exhibits telomerase in its complement whereas the right branch of each branch pair exhibits telomere shortening in its complement since it lacks the telomerase enzyme. The

source code used to generate the asymmetric labeling scheme can be found in Appendix C.

```
// Populates a depth of a chromosome tree using recursion.
// node is the root node of the tree to populate.
function populate(node)
    if node.left is null and node.right is null:
        if node is root:
            root.left = Chromosome(A=0, B=0)
            root.right = Chromosome(A=1, B=0)
        else if node is the left child:
            node.left = Chromosome(A=node.A, B=node.A)
            node.right = Chromosome(A=node.B + 1, B=node.B)
        else:
            node.left = Chromosome(A=node.A, B=node.A)
            node.right = Chromosome(A=node.A, B=node.B)
        return
    populate(node.left)
    populate(node.right)
end function

// Populates the chromosome tree to the specified depth.
// depth is the amount of depths to populate the chromosome
// tree.
function populate_to_depth(depth)
    for 0 to depth:
        populate(root)
    end
end function
```

Figure 3-5: Pseudocode to generate the asymmetric labeling scheme.

3.2 Cell Labeling Combinations

Cell labeling combinations can be used to determine equivalent labeling and the possibility of twins and other higher order multiples. The method of how likewise and crosswise pairings are accomplished is first explained followed by our implementation of generating all possible pairings for a specified cell division in Python.

3.2.1 Likewise and Crosswise Pairings

Using the chromosomes labeling scheme and the asymmetric labeling scheme, various possible cell labeling combinations can be deduced for each model. Humans

have 23 pairs of chromosomes for a total of 46 chromosomes. Out of these pairs, 22 pairs are homologous chromosomes while the 23rd pair is the sex chromosomes. Homologous chromosomes are similar since each contains the same genes in the same order, but they are not identical because the alleles for each trait may be different. Since humans are diploid organisms, each homologous pair contains one chromosome from the mother (maternal chromosome) and one from the father (paternal chromosome). Thus, in the course of division, the daughter cells receive random, independent combinations of chromosome copies from the parent cells. Two such combinations are possible: a likewise combination and a crosswise combination as shown in Figure 3-6. A likewise combination pairs the left labeling of the maternal and paternal subtrees together and pairs the right labeling of both subtrees together. A crosswise combination pairs the left labeling of the maternal subtree with the right labeling of the paternal subtree together and also pairs the right labeling of the maternal subtree with the left labeling of the paternal subtree together.

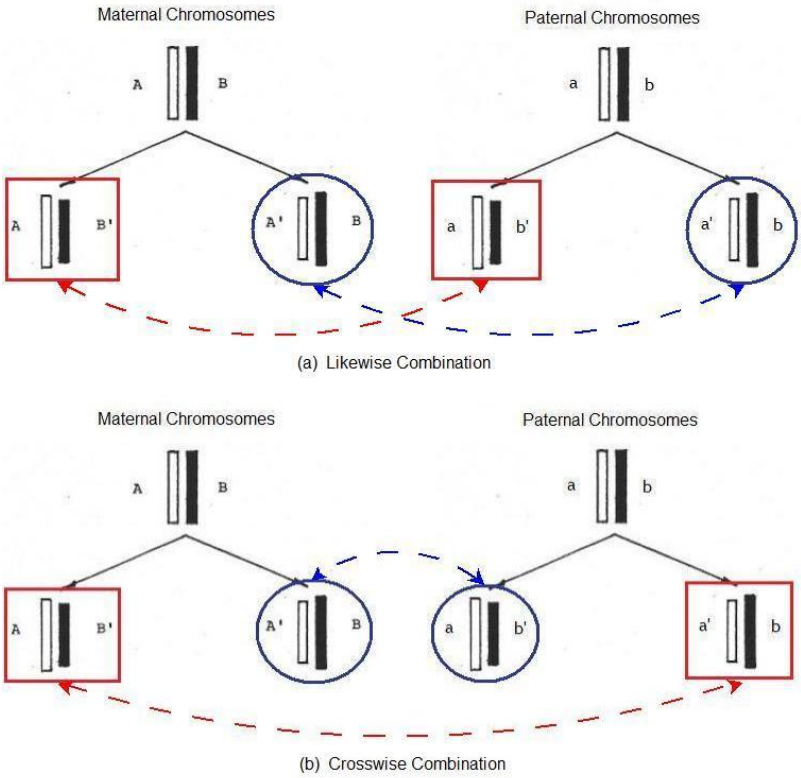

Figure 3-6: Two possible cell labeling combinations of (a) likewise combination and (b) crosswise combination.

As in the case of the initial state of the zygote at the root of the tree, the labeling of a pair of homologous chromosomes can be written as: [(A B), (a b)]. Hence, the cell labeling associated with the two possible combinations after the first division for the chromosomes labeling scheme is illustrated by Berkovich and Bloom [42] in Figure 3-7. Pairs of homologous chromosomes control the cellular clock mechanism. The states of the internal cellular clock are dependent upon the combinations of the chromosome changes as division occurs. Equivalent labeling can be achieved assuming homologous chromosomes influence the states of the internal clock in a similar way. In the labeling, we distinguish between maternal and paternal chromosomes by capital and lowercase lettering. For example, capital letters denote the maternal chromosomes whereas lowercase letters denote the paternal chromosomes. Therefore, the components of the

homologous chromosomes are equivalent where A ≡ a and B ≡ b. In this manner, the crosswise combination in Figure 3-7 will give equivalent labeling for the two daughter cells.

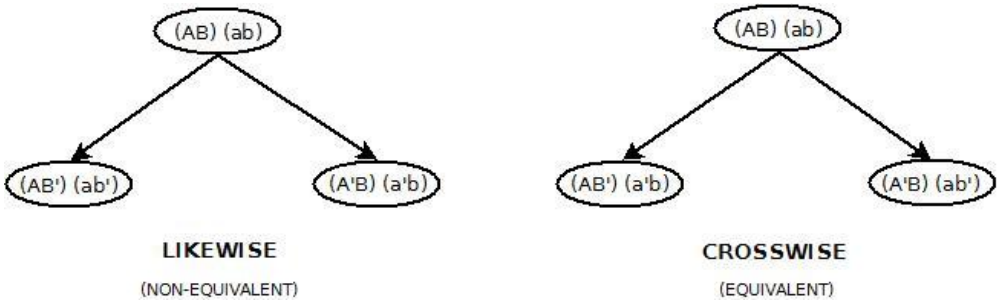

Figure 3-7: Cell labeling of possible combinations of homologous chromosomes at the initial zygote division for the chromosomes labeling scheme.

Correspondingly, the asymmetric labeling scheme yields the same result for the cell labeling of the likewise and crosswise combinations of homologous chromosomes after the first division. Figure 3-8 illustrates these combinations for the first division of the asymmetric labeling scheme. The crosswise combination will also give equivalent labeling for the two daughter cells in this scheme.

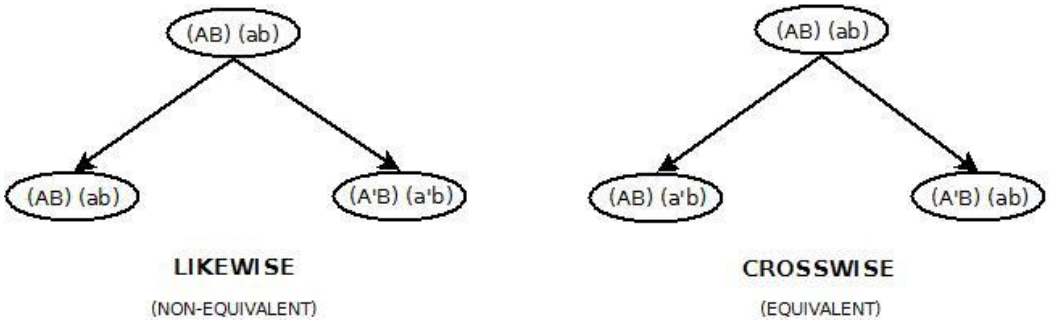

Figure 3-8: Initial zygote division of the asymmetric labeling scheme.

Assuming that K pairs of homologous chromosomes control the internal cellular clock, then there is a possibility of 2^K different ways of the organism beginning its development. This is due to the independent nature of the zygote's chromosomes being

passed on during its division. As noted above, the crosswise combination of each pair of homologous chromosomes is the only way out of the two possible combinations at the initial zygote division where there is equivalent labeling, i.e. identical states of the internal cellular clock. Thus, the probability of such an occurrence for MZT is $(1/2)^K$. Globally, the frequency of human monozygotic twins is approximately 1 in 250 to 300 births [40, 41]. The probability of $(1/2)^8$ falls into that range, suggesting that some set of 8 pairs of homologous chromosomes are essential in the control of a human zygote's development [39].

3.2.2 Generating the Combinations Using Python

Using Python 2.7.8, we can formally generate all possible cell labeling combinations for a user-defined cell division. The program in its entirety can be found in Appendix A while raw sample output of the resulting combinations after running the program can be found in Appendix B. The raw sample output is based on the chromosomes labeling scheme, but output for the asymmetric labeling scheme is generated using the same method. The user must provide a non-negative integer value as an argument for the desired cell division in order for the results to be computed for that division. Once a valid cell division value is inputted, the chromosome tree is generated as explained earlier in Section 3.1.2.2 for the chromosomes labeling scheme, or in Section 3.1.3.2 for the asymmetric labeling scheme, but for an extra division greater than that which is specified. The purpose of generating the tree for an extra division is to allow the chromosome lookup table to be populated with all possible chromosomes in the user-defined division.

Once the chromosome tree is generated, the actual computation of the results of all possible cell labeling combinations for the specified division is accomplished. We write all results of a cell division to a file and use those previous results to compute the next set of results for the subsequent division. The main algorithm is as follows. The file containing the current results, if it exists, is saved as the previous results file. Then, one of three conditions executes as we iterate through the computation of results for each division starting at division zero, i.e. the root of the tree, to the user-defined division:

1. If the cell division to compute is zero, then the root node of the chromosome tree is written to the current results file. This is a trivial case, and no computations are necessary.
2. If the cell division to compute is one, then the likewise result and the crosswise result are written to the current results file sequentially. The resulting combinations of the first cell division are hard-coded as it is trivial to compute.
3. If the cell division to compute is greater than one, then we read the previous division's results file and sequentially perform all likewise and crosswise combinations on each previous result. The resulting combinations are written to the current results file.

The pseudocode for the algorithm used in computing all possible cell labeling combinations for a given cell division can be found in Figure 3-9.

```
// Compute the cell division at the given division level.
// division is the division level to compute.
function compute_result(division)
    if current results file exists:
        save current results file as previous results file

    for div = 0 to division:
        if div is 0:
            write tree.root to current results file
        else if div is 1:
            write first likewise result to current results file
            write first crosswise result to current results file
        else:
            read previous results file as previous_results[]
            for i = 0 to number of previous results:
                result = previous_results[i]
                write all likewise and crosswise combos of result
                  to current results file
    end
end function
```

Figure 3-9: Pseudocode to generate all possible cell labeling combinations for a specified cell division.

Computing the likewise and crosswise pairings are done via the process shown in Figure 3-6. The chromosome lookup table built after the tree is generated is used to determine the children of the parent nodes needed to perform these pairings. The likewise and crosswise functions return a tuple representing the respective pairing from the maternal and paternal chromosomes.

To determine all of the possible likewise and crosswise combinations for each previous result as mentioned in the third condition, a truth table is implemented to accomplish this. Since all combinations have been computed up to that point, we only need to focus on calculating the combinations for the user-defined division. For example, if we want to compute all possible combinations for the third cell division, all possible combinations up to the second cell division are known by reading in the previous results file. Thus, a truth table the size of $2^{2^{(div-1)}}$ rows, where *div* is the cell division of the wanted results, is needed to compute all possible combinations for each previous result.

In the truth table, *False* corresponds to the likewise pairing whereas *True* corresponds to the crosswise pairing.

To ensure that all of the possible combinations can be computed for higher divisions, we store the results for each division in a file and use the Python `pickle` module, particularly the faster `cPickle` module if possible. The `pickle` module implements an algorithm to serialize and de-serialize Python objects, i.e. convert a Python object hierarchy to a byte stream and vice versa. The `cPickle` module is written in the C programming language, thereby making it faster and more optimized. Without using this method of processing all possible likewise and crosswise combinations, an issue with memory limitations may occur depending on the hardware used to run the program.

Chapter 4: Results and Discussion

The appearance of MZT and possible higher order multiples can be found by investigating the cellular labeling as seen in both the chromosomes labeling scheme and the asymmetric labeling scheme. MZT occurs at the beginning of cell differentiation in the very early stages of development. The different states of the internal cellular clock determine the beginning of an organism's formation. As we have previously seen, the simplest situation where MZT may occur is in the initial zygote division as in the case of the crosswise combination since the equivalent labeling of the daughter cells show the non-advancement of the internal cellular clock. In this case, a cell systems' control is distributed across both cells rather than centralized within the zygote, and the process of organism development have not yet been started. Each of these daughter cells will begin with identical states in their internal cellular clocks and eventually develop into two distinct organisms to create identical twins. Since twinning events and higher order multiples are rare in occurrence, this shows that the decentralized cell systems' control is less likely to happen as compared to centralized control in the zygote. We consider situations that occur after the initial zygote division and look at the possibility of higher order multiples in addition to MZT occurrences.

As we progress in the number of cell divisions, the total number of possible labeling combinations in each division increases exponentially. At each division, the number of possible labeling combinations is 2^i, where i = number of branch pairs. So, at the first division, there are $2^1 = 2$ possible labeling combinations; at the second division, there are $2^3 = 8$ possible labeling combinations; at the third division, there are $2^7 = 128$ possible

labeling combinations; at the fourth division, there are 2^{15} = 32768 possible labeling combinations, and etc.

4.1 Appearance of Quadruplet Twins as Identical Pairs

Investigating the second cell division, we observe the first appearance of quadruplets. In this case, the quadruplets are two pairs of monozygotic twins with the cells in each pair having identical states of the cellular clock as seen in Figure 4-1 for the chromosomes labeling scheme. We will coin the term quadruplet twins hereafter to describe this kind of situation. The quadruplet twins are monozygotic in the sense that the offspring are the result of one zygote splitting into two embryos, each of which further splits again. This is as opposed to multizygotic, i.e. fraternal, quadruplets where the four offspring are the result of four separate egg and sperm combinations. The formation of triplets may be possible as a result of having quadruplets where one of the offspring dies.

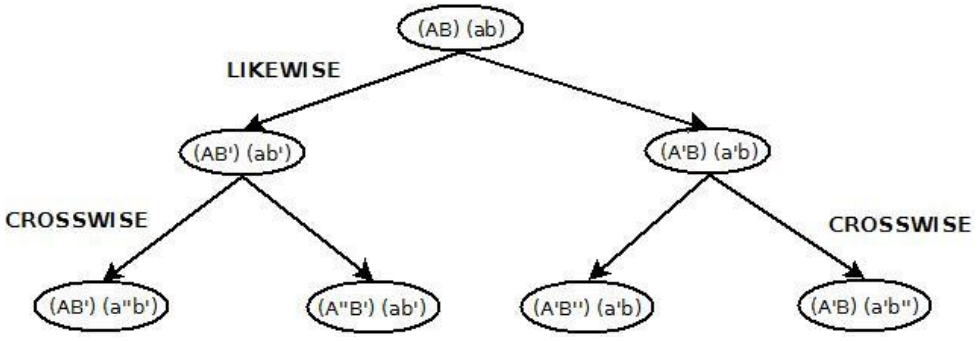

Figure 4-1: Quadruplet twins appearance in the second cell division of the chromosomes labeling scheme.

The possible cell labeling combinations in the second cell division of the asymmetric labeling scheme yield a similar result. Quadruplets appear as quadruplet twins as shown in Figure 4-2. Similar to the result from the chromosomes labeling scheme, the

quadruplets are two pairs of monozygotic twins. The cells in each pair have identical states of the cellular clock and equivalent labeling whereas the two pairs slightly differ in their labeling. No situation where all four daughter cells after the second cell division have equivalent labeling occurs. Therefore, as we can see from our findings, there cannot be monozygotic quadruplets where all four children are identical at this early stage of an organism's development, only pairs of monozygotic twins.

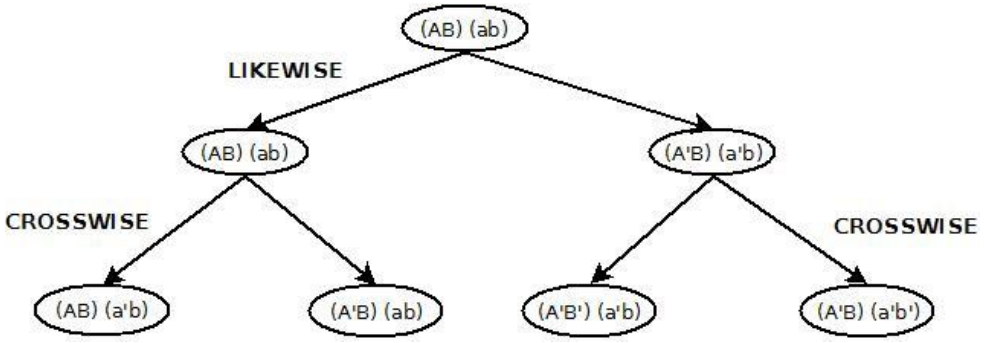

Figure 4-2: Quadruplet twins in the second cell division of the asymmetric labeling scheme.

Since there are eight possible states, i.e. labeling combinations of homologous chromosomes, of the organism in the second division, we discover that the probability of the occurrence of monozygotic quadruplets (or triplets), *P(MZQ)*, is

$$P(MZQ) = \left(\frac{1}{8}\right)^{K} \qquad (4.1)$$

where K = number of pairs of homologous chromosomes. Assuming that $K = 8$ as found from above for humans, this gives a probability of $(1/8)^8$, which is equivalent to a frequency of one in about 16 million for human quadruplet births or triplets if one offspring dies. This is close to the odds that are observed in [55, 56, 57, 58]. Because identical triplets, quadruplets, and other higher order multiples are extremely rare in

occurrence, it is difficult to accurately calculate its frequency. It is estimated that only about 60 sets of identical quadruplets exist worldwide [56, 58, 59]. In addition, it is hard to find this data in scientific literature, and it can usually be found in the common press when experts weigh in on the rarity of such an occurrence.

The origin of monozygotic quadruplets in humans, or triplets as in the case of the death of one embryo, can be explained by our results. Monozygotic quadruplets in humans may actually be two pairs of monozygotic twins where one pair may be a little bit different than the other as no case exists where there are four identical children in the early stages of development, only pairs of identical twins. This concept is a prediction of the theory and can be seen by our cell labeling mechanism of homologous chromosomes for both the chromosomes labeling scheme and the asymmetric labeling scheme. The cells of each pair of quadruplet twins have identical states of the internal cellular clock whereas the two pairs slightly differ in their genetic labeling. Testing and comparing the DNA of the quadruplet twins at various points on the genome and determining if any differences exist can provide insight into our prediction that monozygotic quadruplets are indeed different pairs of monozygotic twins. It would be expected that the two pairs of twins in a set of quadruplet twins will have some slight differences in their genome, but individuals within each pair will have more or less the same genomic structure. If this is the case, then these DNA results will corroborate with our considered models. The notion of pairs of monozygotic twins arising rather than four identical children can be of great significance for the whole concept of an organism's development.

There are rare instances, however, where monozygotic twins are identical but have some noticeable distinctions due to differing genomes rather than environmental factors.

Previous studies [60, 61] have suggested that epigenetic factors, i.e. changes in gene expression due to chemical modification, are the cause of the differences between monozygotic twins. However, recent studies [62, 63] have shown that genetic variation plays an important role in making identical twins different. There may be copy number variants or errors where a twin's DNA has a different number of copies of a particular gene at certain points in the genome. These somatic mutations can occur early in the development of the fetus. This situation may happen in the case of quadruplet twins where the pairs of twins slightly differ. If two of the twins die, then the indicated situation occurs.

Considering another organism other than humans, the nine-banded armadillo and five other closely related species in the genus *Dasypus* also exhibit the phenomenon of monozygotic quadruplet births. These six species of armadillo consistently produce genetically identical quadruplets with every litter to bypass the physical constraints of the uterus so that reproductive success can be achieved [41, 64]. The identical quadruplets are formed by an embryo that splits and each of which splits again similar to the structure seen in the second cell division of our chromosomes labeling scheme and asymmetric labeling scheme. According to our prediction of the theory on the origins of monozygotic quadruplets in humans, it may be the case that quadruplets in these six species of armadillo are not quadruplets but rather two different pairs of monozygotic twins like the occurrences seen in Figures 4-1 and 4-2. This, however, may be true for armadillos in South and Central America, in general, but not for the nine-banded armadillo in the United States. The nine-banded armadillo is the only species of armadillo found outside of South and Central America. The study in [64] found that

armadillos in Brazil have more genetic variability than the nine-banded armadillo in the United States where offspring in a quadruplet litter are genetic clones of one another. This may be due to the fact that armadillos originated from South America, and the nine-banded armadillo is adapting to unusual conditions in their recent arrival to the United States.

The six species of armadillos in the genus *Dasypus* always give birth to monozygotic quadruplets whereas species in other genera of armadillos usually have a litter of one offspring [64]. Currently, there are a total of 20 living species of armadillos. It is difficult to determine the actual frequency of quadruplet births across all species of armadillos since the nine-banded armadillo is the only frequently studied species and no accurate population counts exist. However, if we assume that the population of all species of armadillos is equal in ratio, then the frequency of monozygotic quadruplets in armadillos is 6 in 20 births. The closest K value to fit equation (4.1) for this frequency is $K = 1$ as determined by $6/20 = 3/10 \sim 1/8 = (1/8)^1$. This suggests that one pair of homologous chromosomes controls the zygote's development in armadillos.

4.2 Examination of Further Cell Divisions

The first to third cell divisions are critical in the examination of the mechanisms behind MZT and higher order monozygotic multiples in humans. The fourth cell division is not significant for humans since the odds of any occurrence of multiples are too low to be considered. However, inspecting the fourth division may be effectual for other animals like fish. We examine these further divisions in turn starting with the chromosomes labeling scheme and ending with the asymmetric labeling scheme.

4.2.1 Further Divisions of the Chromosomes Labeling Scheme

During investigation of the third cell division, we consider equivalent labeling for the eight cells that result from the division. Similar to the second division, the third division does not exhibit a situation where all eight cells have equivalent labeling. Rather, there is the possibility of four pairs of twins as seen in Figure 4-3. Each pair of twins is like a first level organism where each twin pair is identical. This scenario replicates that of the quadruplet twins appearance in the second cell division except that, in this case, we have octuplet twins. The octuplet twins are four pairs of monozygotic twins with the cells in each pair having identical states of the cellular clock. The third cell division does not show any equivalent labeling that will result in triplets and above.

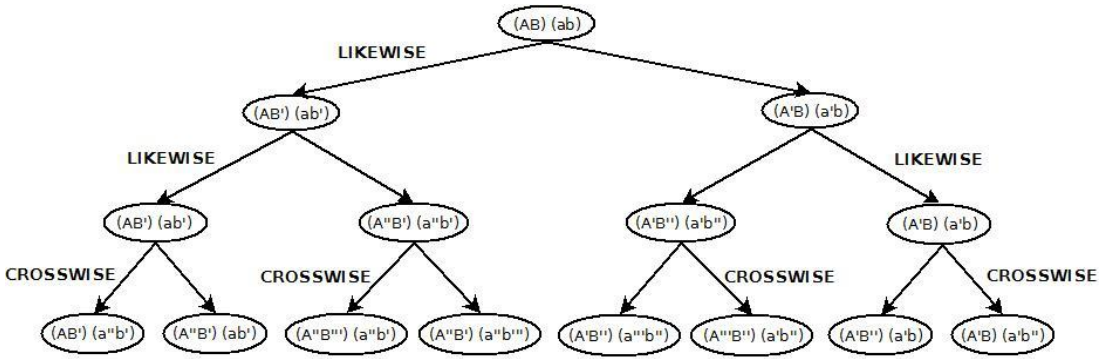

Figure 4-3: Octuplet twins appearance in the third cell division of the chromosomes labeling scheme.

In the fourth division, we consider equivalent labeling for the sixteen cells that result from the division. Similarly, a situation where all sixteen cells have equivalent labeling is not possible. Based on our model, this type of scenario where all resulting cells, i.e. all leaf nodes of the tree, have identical states of the cellular clock will never occur in any cell division except for the first division. Moreover, the fourth division does not exhibit any equivalent labeling that will result in quadruplets and above. The same scenario of

pairs of twins can be seen, and in the fourth cell division, eight pairs of twins with each pair having identical labeling is possible. Generally, this occurs in every division as long as all branch pairs exhibit the likewise combination except for the branch pairs of the final division all being the crosswise combination. We also begin to observe the possibility of sets of identical triplets where three cells have identical states of the cellular clock.

The odds of monozygotic twins and higher order multiples in the fourth cell division are too minuscule to be significant for humans but its consideration may be effectual for other animals such as in the investigation of fish spawn. Depending on the species and the size of the fish, a typical spawn can produce more than a million eggs for large fish whereas small fish may only produce a few hundred eggs per spawn. The species of fish also affect twinning rate, which varies widely. Stockard [65] investigated *Fundulus*, a genus of ray-finned fishes, eggs and never found twins among the eggs developing under normal conditions during a period of fourteen spawning seasons. However, when induced by arresting development in solutions, an occasional twin was discovered in the *Fundulus* eggs. On the contrary, a study on blue tilapia [66] shows a higher MZT rate where asymmetric conjoined twins are frequently encountered in both inbred and outbred embryos.

4.2.2 Further Divisions of the Asymmetric Labeling Scheme

The third cell division of the asymmetric labeling scheme has similarities to that of the chromosomes labeling scheme but also some differences as well. The third division also does not exhibit a situation where all eight cells have equivalent labeling. Instead,

we see the occurrence of octuplet twins, i.e. four pairs of monozygotic twins with the cells in each pair having identical states of the cellular clock. Figure 4-4 displays the occurrence of these octuplet twins. Peculiarly, two of these four pairs are identical in their labeling. This may result in the possibility of quadruplets since it is assumed that the components of the homologous chromosomes are equivalent where A ≡ a and B ≡ b and if we assume that the order of the homologous chromosomes in each cell does not matter. Unlike the chromosomes labeling scheme, however, we discover for the first time equivalent labeling that results in triplets in the third cell division of the asymmetric labeling scheme. These triplets are in their purest form where each component in all three cells is equivalent where A ≡ a and B ≡ b. An example of such a triplet is: [(A′ B′) (a′ b′)], [(A′ B′) (a′ b′)], [(A′ B′) (a′ b′)]. All three of these cells have identical states of the cellular clock. There are no circumstances of quadruplets or higher with this case of the purest form of labeling.

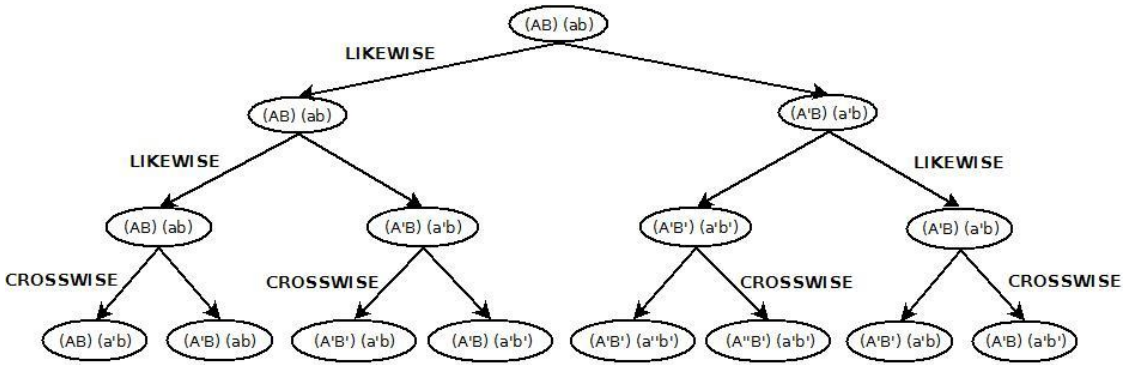

Figure 4-4: Octuplet twins in the third cell division of the asymmetric labeling scheme.

The fourth cell division of the asymmetric labeling scheme also reveals both similarities and differences as compared to the chromosomes labeling scheme. Namely, a situation where all sixteen cells have equivalent labeling is not possible. Like the

chromosomes labeling scheme, the same scenario of pairs of twins can be seen in the asymmetric labeling scheme where eight pairs of twins with each pair having identical labeling is possible in the case of the fourth cell division. The possibility of sets of identical triplets is also present. However, unlike the chromosomes labeling scheme, the asymmetric labeling scheme displays occurrences of identical quadruplets, quintuplets, and sextuplets where four, five, and six cells, respectively, have equivalent labeling and identical states of the cellular clock. These higher order multiples occur in their purest form where each component in all the respective cells is equivalent. The asymmetric labeling scheme seems to provide more opportunities for monozygotic higher order multiples up to and including sextuplets due to the restorative mechanisms that this scheme exhibits. No higher order multiples greater than sextuplets with labeling in the purest form appear in this scheme. As previously mentioned in Section 4.2.1, the consideration of the fourth cell division is not significant for humans but may be effectual for fish spawn.

How advantageous in an evolutionary sense, if at all, to produce monozygotic multiples varies among different species of animals and plants. In humans, twinning is not seen as an evolutionary advantage but rather as a failure of normal function to carry one fetus since there are increased risks of congenital anomalies in monozygotic twin babies [41]. Generally, in animals, the production of monozygotic multiples is a costly way to have offspring because it does not allow for genetic variation in the offspring, which is sometimes beneficial for better adaptation to environmental changes. However, polyembryony is advantageous when sperm are limited or when there are physical constraints in the female that limit the production of eggs as in the case with armadillos to

increase the success rate of reproduction [64]. For some species of plants, extra seedlings per seed help to increase the chances of survival per seed [67]. Twin and triplet seedlings per seed can also provide a boost in the production of high-value crops.

4.3 Implications of Telomeres on Aging

Telomeres are essential to preventing the strands of DNA on the ends of chromosomes from unraveling. They naturally shorten as cells divide and can lengthen by the aid of the enzyme telomerase, which is expressed by germ cells and certain stem cells but not by most human somatic cells. It is unclear whether telomere shortening causes aging or if aging causes telomere shortening. Although the common hypothesis is that telomere shortening contributes to human aging, literature reviews on this topic [68, 69] show that evidence supporting this claim is ambiguous. Telomere shortening with age in cells is difficult to measure and data on telomere length across a life span in various tissues is lacking. Determining which telomere length measure (e.g. longitudinal change or average telomere length) is most informative is important in determining if a relationship between telomeres and aging exists. In addition, there is a lot of variation in average telomere length amongst same-aged individuals and amongst different species of organisms making it even more difficult to establish a causal relationship between telomeres and aging.

Despite all of these uncertainties, it is known that telomeres play a significant role in the cellular response to traumas such as stress and DNA damage [69]. Short telomeres contribute to illness, cellular aging, diseases of premature aging, and other chronic diseases. Thus, telomere shortening does not seem to cause “normal” aging in humans,

but rather, short telomeres make humans more susceptible to various environmental, physical, and psychological stressors. It is possible that as telomeres shorten, cell sensitivity to DNA damage increases making these damage signals replication independent.

A recent study [70] reports that people who drink more regular soda have shorter telomeres in their white blood cells, and thus, accelerated cellular aging. Daily consumption of a 20-ounce soda was calculated to be equivalent to a telomere shortening of an average of 4.6 years. Researchers determined no link between diet sodas and fruit juices and cellular aging, however. They cautioned that the association between telomere length and sugar-sweetened soda consumption does not demonstrate causation since these two data points were only compared for each participant at a single time point. Furthermore, it is unknown whether lifestyle factors among the sample of adults or other variables may have influenced the telomere length of the individuals during the length of the study.

On the other hand, another recent small pilot study [71] following men who are in the early stages of prostate cancer shows that telomeres may actually lengthen over time by implementing some lifestyle changes. A healthy diet, moderate exercise, stress reduction, and social support all led to increased telomere length by about 10% in men in the test group whereas men in the control group saw a decrease in telomere length by about 3% after the end of the five-year study. The healthy lifestyle changes seem to increase telomerase activity causing the increases in telomere length which may help prevent certain chronic diseases.

Telomerase reverses telomere shortening and extends the replication potential of cells. Cells that express the telomerase enzyme can still experience cellular senescence, i.e. the cessation to divide, in response to certain toxins, oncogenes, or DNA damage. In the case of DNA damage, cells may also undergo apoptosis, or programmed cell death, if the repair of the damage is difficult. The only way that telomeres can contribute to aging is if cellular senescence also contributes to aging making telomerase the key to preventing aging. However, telomerase helps protect against cellular senescence by increasing replication potential but not when harmful factors are present which activates senescence and are telomere independent. Therefore, telomeres seem to mostly affect cell differentiation rather than aging because many different factors besides solely telomeres seem to contribute to aging.

Previously, it is thought that cellular senescence appears as a function of chronological age; however, the majority of recent studies that measured in vitro replicative capacity as a function of donor age fail to support this correlation [72]. Cellular senescence is believed to occur from changes in molecular structures that either results from deterioration or are preprogrammed. In principle, such changes are reversible. This inspires gerontology to a specific course of action in cellular research. However, at the system level the limitations of the lifespan of biological organisms must involve some factor of irreversibility. The system control including biological memory should incorporate an indispensable algorithm for resolution of multiple responses to ensure content-addressable access.

In the case of the Big Data situation arising in biological information processing, the pivotal operation of resolution of multiple responses is performed by a certain stream

algorithm [73]. The selection in this stream algorithm is determined by a prevalent fraction of replicated data. Since memory expands as the time goes by, this fraction will fall below an operational threshold. As a result, the control functions of biological information processing degrade irrevocably. The same algorithmic circumstance – low fraction of replication – deprives infants of future recollections of the events of their early life. The reason, however, is opposite: this fraction is low not because the denominator, the size of the memory, is large, but because the numerator, the extent of replication, is small. Thus, the irreversibility of aging at the end of life and some restrictions at the beginning of life are related; a famous French proverb says: *"les extrêmes se touchent"* (total opposites come into contact).

4.4 Implications of Considered Mechanism of Twinning on Cancerogenesis

Although the internal cellular clock and the concept of the Hayflick limit can be seen in humans and in most mammalian cells, this may not be the universal case across all animal species depending on some factors. Most notably, laboratory mice do not exhibit progressive telomere shortening due to long telomeres and telomerase activity [74]. Thus, the idea of the Hayflick limit occurring because of telomere shortening does not apply to mice, although this is the case as seen in humans. Rather, the behavior of certain cell types as if they are evading the Hayflick limit without reaching replicative senescence is an exception to particular types of cells and occurs under specific culture conditions. For example, the study in [75] claims that most rat oligodendrocyte precursor cells from the optic nerve are able to evade replicative senescence and do not stop dividing. This, however, only occurs under specific culture conditions that prevent cells from

differentiating and avoid activation of responses that arrest the cell cycle. In a separate study [76], some mouse cells have been found to behave as if they do not have a Hayflick limit when grown in 3% oxygen concentration levels as opposed to 20% oxygen. More DNA damage was seen in the mouse cells in 20% oxygen than in 3% oxygen.

Despite certain cells in mice dividing indefinitely under certain laboratory conditions, laboratory mice cells still have a Hayflick limit, albeit lower than that of humans [77]. It is explained in [78] that mice cells become senescent after 10 to 15 mitotic divisions in vitro and that mice have a short lifespan of about two years. Thus, the Hayflick limit is not determined by telomere shortening in mice, but rather, may be due to DNA damage in the cells. The universality of the Hayflick limit is dependent upon the type of cell studied and environmental conditions. Humans are naturally exposed to various environmental factors that may increase telomere shortening whereas laboratory mice are held in highly controlled and sterile environments which may disrupt effects of interactions among factors affecting telomere erosion [78]. Generally, it has been found that telomere length inversely correlates with lifespan in mammals, and telomerase expression inversely correlates with body size [74, 78]. This indicates that telomeres have different purposes depending on the species.

Apoptosis, i.e. programmed cell death, and the occurrence of Hayflick's limit are essential for the stabilization of constancy of cell agglomerations in working organisms. Remarkably, the considered mechanism of MZT at the beginning of cell development and apoptosis at the end provides insight towards the process of cancerogenesis. The emergence of cancerogenesis is a result of a breakdown of protection by apoptosis.

Namely, the probability of failure of the cell death, Z_T, between cell divisions is given by the formula from [39]:

$$Z_T \simeq \frac{\ln\left(3-\frac{2}{K}\right)}{M+\alpha\sqrt{M}} \sim \frac{1}{M} \qquad (4.2)$$

It is approximately inversely proportional to the Hayflick limit M. Here, K is the species-specific integer parameter representing the number of pairs of homologous chromosomes, and α is a standard deviation coefficient about 1. Equation (4.2) assumes that the Hayflick limit for a species is known regardless of whether the Hayflick limit is determined by telomere shortening as in the case of most mammals or by some other factor such as DNA damage as seen in the case of laboratory mice.

Thus, if some cancerization factor, such as exposure to radiation or pollution, is to somehow exceed a certain threshold, cancer results; otherwise, the number of potentially malicious cells that escape apoptosis will not reach the critical level for the formation of malignancy. The parameters of the phenomenon of MZT in certain laboratory animals, like mice, can be used to study the considered threshold effect for cancerization, and the results can be quantitatively transferred and compared with the observations on humans.

Although we consider the cell labeling models to determine the origins of pairs of monozygotic twins and to derive the parameter K, we can also see the situation of cancerogenesis by using these models. In this case, cancerogenesis may be seen when groups of cells in the models continuously divide past the Hayflick limit of 40 to 60 divisions as in the case for humans. This may especially be observed in the asymmetric labeling scheme because of the restorative mechanisms exhibited by telomerase activity. In order to distinguish between the cancer cells from the normal cells with active

telomerase, the cells can be tested for increased telomerase expression. Cancer cells will display telomerase in amounts more than that of normal cells. The increase in telomerase has been shown to be a common feature of human tumor cells, thus making investigations into inhibiting telomerase in cancer cells an attractive means to decreasing tumor growth and causing apoptosis in the tumors [79]. Therefore, the model of the developmental tree along with the parameter K can be considered in the two opposite circumstances of MZT and cancerogenesis.

The epigenetic clock developed by Steve Horvath [80] may provide more insight into the causes of aging in addition to cancer research by measuring the biological age of organs and tissues using DNA methylation levels. He provides a proof of concept showing that the biological clock can be reset to zero by reprogramming adult cells back into a stem-cell-like state. Nevertheless, uncovering the mechanism behind the internal cellular clock, of which we provided insight to, is the key to future developments in developmental biology, the human aging process, and cancerogenesis.

Chapter 5: Conclusion

We consider various possible cell labeling schemes and see that by investigating both the chromosomes labeling scheme and the asymmetric labeling scheme, some insight into the origins of pairs of monozygotic twins and higher order multiples can be gained. From the possible labeling combinations of the second cell division, the first appearance of monozygotic quadruplets as quadruplet twins can be seen. We discover that identical quadruplets may actually be two pairs of monozygotic twins where the pairs of twins are slightly different from one another. This theory can be tested by DNA analysis of the twin pairs and may apply to not only humans but animals like the armadillo as well. By exploring the next few cell divisions, we see that a situation where all cells at the same level of the developmental tree have identical states of the cellular clock will never occur. The only occurrence of this sort is in the crosswise combination of the initial zygote division where there is the possibility of monozygotic twins. The notion of pairs of monozygotic twins is prevalent throughout all of the explored cell divisions for both considered models, which can be of great significance for the concept of an organism's development as a whole.

From our analysis, the theoretical parameter, K, reveals the structure of the internal cellular clock and determines the threshold effect in cancerization. For example, cancer may occur if the intensity of radiation exposure exceeds a certain level. This theory can be tested on laboratory animals such as mice where $K = 6$. Depending upon the species, the number of pairs of homologous chromosomes, K, seems to be a determining factor in the zygote's development and in cell differentiation. If the internal cellular clock fails, then a cell may not have the limitation on replication resources that it normally has. The

internal cellular clock may fail because the enzyme telomerase prevents the telomeres from shortening or the ends of chromosomes fuse together if no more telomeres exist. The Hayflick limit and cellular senescence act as an additional protection from cancer [81]. Without it, cancer is more likely to occur but is not inevitable. The process of programmed cell death, or apoptosis, has been found to have a genetic basis where factors that control cell proliferation and differentiation can affect cell numbers, and mutations can disrupt cell death leading to cancer and other human diseases [82]. The phenomenon of biological aging may be to a considerable degree associated with irreversible factors in view of the complications in the organization of Big Data information processing.

References

[1] Leonard Hayflick. "The limited in vitro lifetime of human diploid cell strains." *Experimental Cell Research*, Vol. 37, No. 3, March 1965, pp. 614-636.

[2] A. M. Olovnikov. "Telomeres, telomerase, and aging: origin of the theory." *Exp. Gerontol.*, Vol. 31, No. 4, Jul.-Aug. 1996, pp. 443-448.

[3] J. H. Edwards, Tessa Dent, and Jacob Kahn. "Monozygotic Twins of Different Sex." *J. Med. Genet.*, Vol. 3, No. 2, June 1966, pp. 117-123.

[4] R. Schmidt, E. H. Sobel, H. M. Nitowsky, H. Dar, and F. H. Allen, Jr. "Monozygotic twins discordant for sex." *J. Med. Genet.*, Vol. 13, No. 1, Feb. 1976, pp. 64-68.

[5] T. Costa, M. Lambert, I. Teshima, P. N. Ray, C. L. Richer, and L. Dallaire. "Monozygotic twins with 45,X/46,XY mosaicism discordant for phenotypic sex." *Am. J. Med. Genet.*, Vol. 75, No. 1, Jan. 1998, pp. 40-44.

[6] B. Dallapiccola, C. Stomeo, G. Ferranti, A. Di Lecce, and M. Purpura. "Discordant sex in one of three monozygotic triplets." *J. Med. Genet.*, Vol. 22, No. 1, Feb. 1985, pp. 6-11.

[7] Renata Bortolus, Fabio Parazzini, Liliane Chatenoud, Guido Benzi, Massimiliano Maria Bianchi, and Alberto Marini. "The epidemiology of multiple births." *Human Reproduction Update*, Vol. 5, No. 2, 1999, pp. 179-187.

[8] Judith G. Hall. "Twinning." *Lancet*, Vol. 362, No. 9385, August 2003, pp. 735-743.

[9] Chantal Hoekstra, Zhen Zhen Zhao, Cornelius B. Lambalk, Gonneke Willemsen, Nicholas G. Martin, Dorret I. Boomsma, and Grant W. Montgomery. "Dizygotic twinning." *Human Reproduction Update*, Vol. 14, No. 1, 2008, pp. 37-47.

[10] Charles Choi. (2011, Oct. 11). "Countries With Most Twins Identified." *LiveScience*. [Online]. Available: http://www.livescience.com/16469-twins-countries-twinning-rates.html [Oct. 22, 2014].

[11] I. Blickstein. "Monochorionicity in perspective." *Ultrasound Obstet. Gynecol.*, Vol. 27, 2006, pp. 235-238.

[12] Gonzalo Herranz. "The timing of monozygotic twinning: a criticism of the common model." *Zygote*, June 2013, pp. 1-14.

[13] Hans-Werner Denker. "Comment on G. Herranz: The timing of monozygotic twinning: a criticism of the common model. *Zygote* (2013)." *Zygote*, November 2013, pp. 1-3.

[14] R. L. Gardner. "The timing of monozygotic twinning: a pro-life challenge to conventional scientific wisdom." *Reproductive BioMedicine Online*, Vol. 28, No. 3, March 2014, pp. 276-278.

[15] G. Herranz. "Response: The timing of monozygotic twinning: a pro-life challenge to conventional scientific wisdom." *Reproductive BioMedicine Online*, Vol. 28, No. 6, June 2014, p. 789.

[16] V. J. Baldwin. *Pathology of Multiple Pregnancy*. Springer-Verlag: New York, 1994, pp. 29-62.

[17] J. Conaghan. Available: http://stemcells.nih.gov/info/scireport/pages/chapter3.aspx [Oct. 24, 2014]

[18] Judith G. Hall. "Twinning: mechanisms and genetic implications." *Curr. Opin. Genet. Dev.*, Vol. 6, No. 3, June 1996, pp. 343-347.

[19] Judith G. Hall. "Twins and twinning." *Am. J. Med. Genet.*, Vol. 61, No. 3, January 1996, pp. 202-204.

[20] Henry Gray. "Embryology," in *Anatomy of the Human Body*, 20th Ed. Warren H. Lewis, Ed. Philadelphia and New York: Lea & Febiger, 1918, p. 39.

[21] Mina Alikani, Nicole Noyes, Jacques Cohen, and Zev Rosenwaks. "Monozygotic twinning in the human is associated with the zona pellucida architecture." *Human Reproduction*, Vol. 9, No. 7, July 1994, pp. 1318-1321.

[22] Maria Cheng. (2007, Jul. 3). "Study: Twins form after embryo collapses." *USA Today*. [Online]. Available: http://usatoday30.usatoday.com/news/health/2007-07-03-52152171_x.htm [Oct. 23, 2014].

[23] Joanita Monteiro, Catherine Derom, Robert Vlietinck, Nina Kohn, Martin Lesser, and Peter K. Gregersen. "Commitment to X Inactivation Precedes the Twinning Event in Monochorionic MZ Twins." *The American Journal of Human Genetics*, Vol. 63, No. 2, August 1998, pp. 339-346.

[24] Shai E. Elizur, Jacob Levron, Alon Shrim, Eyal Sivan, Jehoshua Dor, and Adrian Shulman. "Monozygotic twinning is not associated with zona pellucida micromanipulation procedures but increases with high-order multiple pregnancies." *Fertility and Sterility*, Vol. 82, No. 2, August 2004, pp. 500-501.

[25] Morey Schachter, Arieh Raziel, Shevach Friedler, Devorah Strassburger, Orna Bern, and Raphael Ron-El. "Monozygotic twinning after assisted reproductive techniques: a phenomenon independent of micromanipulation." *Human Reproduction*, Vol. 16, No. 6, June 2001, pp. 1264-1269.

[26] E. Scott Sills, Maureen Moomjy, Nikica Zaninovic, Lucinda L. Veeck, Monnie McGee, Gianpiero D. Palermo, and Zev Rosenwaks. "Human zona pellucida micromanipulation and monozygotic twinning frequency after IVF." *Human Reproduction*, Vol. 15, No. 4, April 2000, pp. 890-895.

[27] D. Hellin. *Die Ursache der Multiparität der uniparen Tiere überhaupt und der Zwillingsschwangerschaft beim Menschen insbesondere* [The causes of multiple maternities among uniparous animals and in man]. München: Seitz & Schauer. 1895.

[28] P. Strassmann. *Zur Lehre von der mehrfachen Schwangerschaft* [On multiple maternities]. Thesis, Berlin. 1889.

[29] Johan Fellman and Aldur W. Eriksson. "On the History of Hellin's Law." *Twin Research and Human Genetics*, Vol. 12, No. 2, April 2009, pp. 183-190.

[30] Johan Fellman and Aldur W. Eriksson. "Statistical Analyses of Hellin's Law." *Twin Research and Human Genetics*, Vol. 12, No. 2, April 2009, pp. 191-200.

[31] Johan O. Fellman and Aldur W. Eriksson. "Biometric Analysis of the Multiple Maternities in Finland, 1881-1990, and in Sweden since 1751." *Human Biology*, Vol. 65, No. 3, June 1993, pp. 463-479.

[32] Wilhelm Weinberg. "Beiträge zur Physiologie und Pathologie der Mehrlingsbeburten beim Menschen." *Arch. Ges. Physiol.*, Vol. 88, No. 6-8, December 1901, pp. 346-430.

[33] W. H. James. "Excess of Like Sexed Pairs of Dizygotic Twins." *Nature*, Vol. 232, July 1971, pp. 277-278.

[34] W. H. James. "Is Weinberg's differential rule valid?" *Acta Genet. Med. Gemellol. (Roma)*, Vol. 28, No.1, 1979, pp. 69-71.

[35] M. G. Bulmer. "Is Weinberg's method valid?" *Acta Genet. Med. Gemellol. (Roma)*, Vol. 25, 1976, pp. 25-28.

[36] Johan Fellman and Aldur W. Eriksson. "Weinberg's differential rule reconsidered." *Human Biology*, Vol. 78, No. 3, June 2006, pp. 253-275.

[37] Gert Bonnier. "Studies on monozygous cattle twins. II. Frequency of monozygous twins." *Acta Agr. Suecana*, Vol. 1, 1946, pp. 147-151.

[38] C. E. Meadows and J. L. Lush. "Twinning in Dairy Cattle and Its Relation to Production." *J. Dairy Sci.*, Vol. 40, No. 11, November 1957, pp. 1430-1436.

[39] Simon Y. Berkovich. "A Cybernetical Model of the Internal Cellular Clock." *Medical Hypotheses*, 1981, pp. 1347-1357.

[40] Jeroen Smits and Christiaan Monden. "Twinning Across the Developing World." *PLoS ONE*, Vol. 6, No. 9, 2011.

[41] David Cyranoski. "Developmental biology: Two by two." *Nature*, Vol. 458, 2009, pp. 826-829.

[42] Simon Y. Berkovich and Sherman Bloom. "Probability of Monozygotic Twinning as a Reflection of the Genetic Control of Cell Development." *Mechanisms of Ageing and Development*, Elsevier Scientific Publishers Ireland Ltd., 1985, pp. 147-154.

[43] Simon Y. Berkovich. "Informational Structure of the Developmental Tree of Multi-Cellular Organisms." *Evolution of Longevity in Animals*, Plenum Publishing Corporation, 1987, pp. 23-33.

[44] M. E. Wallace and D. A. Williams. "Monozygotic Twinning in Mice." *J. Med. Genet.*, Vol. 2, No. 26, 1965, pp. 26-31.

[45] L. Grapes, M. Malek, and M. F. Rothschild. "Identification of Identical Twins and Mutation Rate in Pigs." *Swine Research Report, 2000*, Paper 23, 2001, pp. 83-84.

[46] N. Silva del Rio, B. W. Kirkpatrick, and P. M. Fricke. "Observed frequency of monozygotic twinning in Holstein dairy cattle." *Theriogenology*, Vol. 66, 2006, pp. 1292-1299.

[47] L. H. Filonova, S. von Arnold, G. Daniel, and P. V. Bozhkov. "Programmed cell death eliminates all but one embryo in a polyembryonic plant seed." *Cell Death and Differentiation*, Vol. 9, No. 10, October 2002, pp. 1057-1062.

[48] Zhong Chen and Daniel R. Gallie. "Induction of Monozygotic Twinning by Ascorbic Acid in Tobacco." *PLoS ONE*, Vol. 7, No. 6, June 2012.

[49] Simon Y. Berkovich. "Mechanism for Controlling Development in Biological Systems." *Automation and Remote Control*. Vol. 38, No. 6, Part 2, 1977, pp. 945-947.

[50] Leslie A. Pray. "Semi-Conservative DNA Replication: Meselson and Stahl." *Nature Education*. Vol. 1, No. 1, 2008, p. 98.

[51] A. M. Olovnikov. "A Theory of Marginotomy: The Incomplete Copying of Template Margin in Enzymic Synthesis of Polynucleotides and Biological Significance of the Phenomenon." *Journal of Theoretical Biology*, Vol. 41, No. 1, September 1973, pp. 181-190.

[52] W. E. Wright, M. A. Piatyszek, W. E. Rainey, W. Byrd, and J. W. Shay. "Telomerase activity in human germline and embryonic tissues and cells." *Developmental Genetics*, Vol. 18, No. 2, January 1996, pp. 173-179.

[53] K. M. Eisenhauer, R. M. Gerstein, C. P. Chiu, M. Conti, and A. J. Hsueh. "Telomerase activity in female and male rat germ cells undergoing meiosis and in early embryos." *Biology of Reproduction*, Vol. 56, No. 5, May 1997, pp. 1120-1125.

[54] A. J. S. Klar. "The developmental fate of fission yeast cells is determined by the pattern of inheritance of parental and grandparental DNA strands." *The EMBO Journal*, Vol. 9, No. 5, June 1990, pp. 1407-1415.

[55] "Rare identical triplets born in Sacramento." Internet: http://www.upi.com/Top_News/US/2013/12/05/Rare-identical-triplets-born-in-Sacramento/UPI-49981386265337/, Dec. 5, 2013 [Aug. 24, 2014].

[56] Greg Wilson. "Woman Gives Birth to Identical Quadruplets." Internet: http://www.nbcwashington.com/news/weird/Woman-Gives-Birth-to-Identical-Quadruplets-137081473.html, Jan. 11, 2012 [Aug. 24, 2014].

[57] Rene Lynch. "Woman, 42, expecting triplets ends up delivering…quadruplets." *Los Angeles Times*, Feb. 17, 2014. [Online]. Available: http://www.latimes.com/nation/shareitnow/la-sh-woman-surprised-by-quadruplets-20140217-story.html#ixzz2tcobkNZq. [Aug. 24, 2014].

[58] "About Four of a Kind." Internet: http://www.mylifetime.com/shows/four-of-a-kind/about, [Aug. 24, 2014].

[59] Jacque Wilson. “Identical quadruplets surprise mom expecting triplets.” Internet: http://www.cnn.com/2014/02/17/health/identical-quadruplets-surprise/, Feb. 17, 2014 [Aug. 24, 2014].

[60] Sarah Graham. (2005, Jul. 5). “Identical Twins Exhibit Differences in Gene Expression.” *Scientific American*. [Online]. Available: http://www.scientificamerican.com/article/identical-twins-exhibit-d/ [Aug. 24, 2014].

[61] Robert Roy Britt. (2005, Jul. 8). “Identical Twins Not So Identical.” *LiveScience*. [Online]. Available: http://www.livescience.com/3905-identical-twins-identical.html [Aug. 24, 2014].

[62] Anne Casselman. (2008, Apr. 3). “Identical Twins’ Genes Are Not Identical.” *Scientific American*. [Online]. Available: http://www.scientificamerican.com/ article/identical-twins-genes-are-not-identical/ [Aug. 24, 2014].

[63] Tia Ghose. (2012, Nov. 9). “Identical Twins Are Genetically Different, Research Suggests.” *LiveScience*. [Online]. Available: http://www.livescience.com/24694-identical-twins-not-identical.html [Aug. 24, 2014].

[64] W. J. Loughry, Paulo A. Prodohl, Colleen M. McDonough, and John C. Avise. “Polyembryony in Armadillos.” *American Scientist*, Vol. 86, May-June 1998, pp. 274-279.

[65] Charles R. Stockard. "Developmental rate and structural expression: An experimental study of twins, 'double monsters' and single deformities, and the interaction among embryonic organs during their origin and development." *The American Journal of Anatomy*, Vol. 28, No. 2, Jan. 1921, pp. 115-277.

[66] A. Shirak, Y. Palti, O. Bern, T. D. Kocher, E. Gootwine, E. Seroussi, G. Hulata, M. Ron, and R. R. Avtalion. "A deleterious effect associated with UNH159 is attenuated in twin embryos of an inbred line of blue tilapia *Oreochromis aureus*." *Journal of Fish Biology*, 2012, pp. 1-12.

[67] "Link between vitamin C, twins can increase seed production in crops." Internet: http://phys.org/news/2012-06-link-vitamin-twins-seed-production.html, Jun. 18, 2012 [Nov. 10, 2014].

[68] Karen Anne Mather, Anthony Francis Jorm, Ruth Adeline Parslow, and Helen Christensen. "Is Telomere Length a Biomarker of Aging? A Review." *The Journals of Gerontology: Series A*, Vol. 66A, No. 2, Sept. 2010, pp. 202-213.

[69] Geraldine Aubert and Peter M. Lansdorp. "Telomeres and Aging." *Physiological Reviews*, Vol. 88, No. 2, April 2008, pp. 557-579.

[70] Cindy W. Leung, Barbara A. Laraia, Belinda L. Needham, David H. Rehkopf, Nancy E. Adler, Jue Lin, Elizabeth H. Blackburn, and Elissa S. Epel. "Soda and Cell Aging: Associations Between Sugar-Sweetened Beverage Consumption and Leukocyte Telomere Length in Healthy Adults From the National Health and Nutrition Examination Surveys." *American Journal of Public Health*, Vol. 104, No. 12, December 2014, pp. 2425-2431.

[71] Dean Ornish, Jue Lin, June M. Chan, Elissa Epel, Colleen Kemp, Gerdi Weidner, and et. al. "Effect of comprehensive lifestyle changes on telomerase activity and telomere length in men with biopsy-proven low-risk prostate cancer: 5-year follow-up of a descriptive pilot study." *The Lancet Oncology*, Vol. 14, No. 11, Oct. 2013, pp. 1112-1120.

[72] A. B. Maier and R. G. Westendorp. “Relation between replicative senescence of human fibroblasts and life history characteristics.” *Ageing Research Reviews*, Vol. 8, No. 3, July 2009, pp. 237-243.

[73] Simon Berkovich. “Organization of the Brain in Light of the Big Data Philosophy.” *2014 Fifth International Conference on Computing for Geospatial Research and Application*, IEEE, Washington, D.C., August 2014, pp. 91-92.

[74] Nuno M. V. Gomes, Oliver A. Ryder, Marlys L. Houck, Suellen J. Charter, William Walker, Nicholas R. Forsyth, Steven N. Austad, Chris Venditt, Mark Pagel, Jerry W. Shay, and Woodring E. Wright. “Comparative biology of mammalian telomeres: hypotheses on ancestral states and the roles of telomeres in longevity determination.” *Aging Cell*, Vol. 10, No. 5, Oct. 2011, pp. 761-768.

[75] D. G. Tang, Y. M. Tokumoto, J. A. Apperly, A. C. Lloyd, and M. C. Raff. “Lack of Replicative Senescence in Cultured Rat Oligodendrocyte Precursor Cells.” *Science*, Vol. 291, No. 5505, Feb. 2001, pp. 868-871.

[76] S. Parrinello, E. Samper, A. Krtolica, J. Goldstein, S. Melov, and J. Campisi. “Oxygen sensitivity severely limits the replicative lifespan of murine fibroblasts.” *Nature Cell Biology*, Vol. 5, No. 8, Aug. 2003, pp. 741-747.

[77] Joel C. Eissenberg. “The Methuselah Question: What are the Ultimate Limits of the Human Lifespan? Telomeres, Cancer and Aging: Live Long and Prosper?” *Missouri Medicine, the Journal of the Missouri State Medical Association*, March 2013.

[78] Rodrigo T. Calado and Bogdan Dumitriu. “Telomere dynamics in mice and humans.” *Seminars in Hematology*, Vol. 50, No. 2, Apr. 2013, pp. 165-174.

[79] Elizabeth H. Blackburn. "Telomerase and Cancer: Kirk A. Landon – AACR Prize for Basic Cancer Research Lecture." *Molecular Cancer Research*, Vol. 3, No. 9, Sept. 2005, pp. 477-482.

[80] Steve Horvath. "DNA methylation age of human tissues and cell types." *Genome Biology*, Vol. 14, No. 10, Oct. 2013, pp. R115.

[81] Thomas Kuilman, Chrysiis Michaloglou, Wolter J. Mooi, and Daniel S. Peeper. "The essence of senescence." *Genes & Development*, Vol. 24, 2010, pp. 2463-2479.

[82] Scott W. Lowe and Athena W. Lin. "Apoptosis in cancer." *Carcinogenesis*, Vol. 21, No. 3, 2000, pp. 485-495.

Appendix A: Program to Generate Model and Labeling Combinations

The entire program used to generate the chromosomes labeling scheme and all of the possible cell labeling combinations for a specified cell division can be found here. It consists of four total files:

- chromosome.py - provides a representation of a chromosome
- chromotree.py - generates the chromosome labeling scheme
- chromo_combo.py - determines all possible likewise and crosswise combinations
- cell_division.py - the main program

The program takes as input a user defined non-negative integer representing the cell division of the possible labeling combinations that the user wants computed. It outputs all of the possible likewise and crosswise pairings for the specified division. A sample of the raw output can be found in Appendix B.

File #1: chromosome.py

```
"""
Provides a representation of a chromosome.

Author: Carol Jim
"""


class Chromosome:
    """Represents a chromosome node in the chromosome tree."""

    def __init__(self, a=0, b=0, is_left=False, is_maternal=True):
        """
        Construct a new Cell.

        Parameters:
          a - Represents this chromosome's 'A' prime value (default 0).
          b - Represents this chromosome's 'B' prime value (default 0).
          is_left - Indicates if this chromosome is a left child of a
                    parent chromosome.
          is_maternal - Indicates that this chromosome represents a
                        maternal chromosome.
        """
        self.left = None
        self.right = None
        self.a = a
        self.b = b
        self.is_left = is_left
        self.is_maternal = is_maternal

    def __str__(self):
        """Return a string representation of this chromosome."""
        if self.is_maternal:
            return "[A" + "'" * self.a + ", B" + "'" * self.b + "]"
        else:
            return "[a" + "'" * self.a + ", b" + "'" * self.b + "]"

    def __repr__(self):
```

```
        """Return a string representation of this chromosome."""
        if self.is_maternal:
            return "[A" + "'" * self.a + ", B" + "'" * self.b + "]"
        else:
            return "[a" + "'" * self.a + ", b" + "'" * self.b + "]"
```

File #2: chromotree.py

```
"""
Provides methods for generating and iterating over a chromosome tree.

Author: Carol Jim
"""

from __future__ import print_function
from chromosome import Chromosome


class Tree:
    """Represents a chromosome tree."""

    def __init__(self, depth=0):
        """
        Construct and initialize a new chromosome tree.

        Parameters:
          depth - The depth of the tree to generate (default 0).
        """

        # Holds root of the chromosome tree.
        self.root = Chromosome(0, 0, False)

        # Holds a lookup table of nodes and their children.
        self.chromosome_lookup = dict()

        # Populate the tree according to the depth specified.
        self._populate_to_depth(depth)

    def _populate(self, root):
        """
        Populate the next depth of the chromosome tree using
        recursion.

        Parameters:
          root - The root of the chromosome tree to populate.

        Return:
          None.
        """
        if (root.left is None) and (root.right is None):
            if root is self.root:
                self.root.left = Chromosome(0, 1, True)
                self.root.right = Chromosome(1, 0, False)
            elif root.is_left:
                root.left = Chromosome(root.a, root.b, True)
                root.right = Chromosome(root.b + 1, root.b, False)
```

```
            else:
                root.left = Chromosome(root.a, root.a + 1, True)
                root.right = Chromosome(root.a, root.b, False)

            return

        self._populate(root.left)
        self._populate(root.right)

    def _inorder_traversal_print(self, root):
        """
        Traverse the chromosome tree in-order, displaying its
        contents to standard output.

        Parameters:
          root - The root of the chromosome tree to traverse.

        Return:
          None.
        """
        if root is None:
            return
        self._inorder_traversal_print(root.left)
        print(root)
        self._inorder_traversal_print(root.right)

    def _populate_lookup_table(self, root):
        """
        Traverse the chromosome tree in-order, populating the node
        lookup table.

        Parameters:
          root - The root of the chromosome tree to traverse.

        Return:
          None.
        """
        if root is None or root.left is None or root.right is None:
            return
        self._populate_lookup_table(root.left)
        self.chromosome_lookup[str(root)] = root
        self._populate_lookup_table(root.right)

    def _populate_to_depth(self, depth):
        """
        Populates the specified amount of depths of the chromosome
        tree.

        If the amount of depths specified is <= 0, this function does
        nothing.

        Parameters:
          depth - The amount of depths to populate the chromosome tree.

        Return:
          None.
        """
```

```
        # Populate the depths and generate the lookup table.
        for n in range(depth):
            self._populate(self.root)
        self._populate_lookup_table(self.root)

    def inorder_traversal_print(self):
        """
        Traverse the chromosome tree in-order, displaying its contents
        to standard output.

        Parameters:
          None.

        Return:
          None.
        """
        self._inorder_traversal_print(self.root)

    def display_lookup_table(self):
        """
        Displays the lookup table to standard output.

        Parameters:
          None.

        Return:
          None.
        """
        for k, v in self.chromosome_lookup.iteritems():
            print("Root '" + k + "': " + str(v.left) + "  |  " +
                  str(v.right))
```

File #3: chromo_combo.py

```
"""
Provides facilities to determine chromosome combinations through the
use of likewise and crosswise pairing.

Author: Carol Jim
"""

from __future__ import print_function
from chromosome import Chromosome
from chromotree import Tree

import os
import itertools

# Attempt to utilize the faster pickler.
try:
    import cPickle as pickle
except ImportError:
    import pickle
```

```
class ChromoCombo:
    """Represents a class to facilitate the determination of chromosome
    combinations."""

    def __init__(self, pickle_file):
        """
        Construct the class to determine chromosome combinations.

        Parameters:
          pickle_file - The file name to save pickled results.

        Throws:
          ValueError - If the pickle filename is invalid.
        """

        # This file holds the previous results.
        self._PREV_PICKLE = "prev.pik"

        # Save the pickle file name.
        if pickle_file is None:
            raise ValueError("'None' pickle filename given")
        else:
            self._curr_pickle = pickle_file

        # Holds the chromosome tree that will be used to help determine
        # chromsome combinations.
        self._tree = None

        # Generate the first cell division.  This is hard-coded as it
        # is trivial to calculate by hand.
        self._first_likewise = []
        self._first_crosswise = []
        self._first_likewise.append((Chromosome(a=0, b=1,
                                                is_maternal=True),
                                     Chromosome(a=0, b=1,
                                                is_maternal=False)))
        self._first_likewise.append((Chromosome(a=1, b=0,
                                                is_maternal=True),
                                     Chromosome(a=1, b=0,
                                                is_maternal=False)))
        self._first_crosswise.append((Chromosome(a=0, b=1,
                                                 is_maternal=True),
                                      Chromosome(a=1, b=0,
                                                 is_maternal=False)))
        self._first_crosswise.append((Chromosome(a=1, b=0,
                                                 is_maternal=True),
                                      Chromosome(a=0, b=1,
                                                 is_maternal=False)))

    def _generate_tree(self, division=0):
        """
        Generates a new chromosome tree.

        The tree is generated an extra division greater than the
        division specified.  This is to allow the chromosome
        lookup dictionary to be generated with all possible
        chromosomes in the specified division.
```

```
        Parameters:
          division - The cell division wanted computed (default 0).

        Return:
          None.
        """
        self._tree = Tree(division + 1)

    def _likewise(self, maternal, paternal):
        """
        Computes the likewise pairing of the given chromosomes.

        Parameters:
          maternal - The maternal chromosome.
          paternal - The paternal chromosome.

        Return:
          A tuple representing the likewise pairing of 'maternal' and
          'paternal'.

        Throws:
          ValueError - If either chromosome is not specified or is not
                       found in the chromosome lookup dictionary.
        """

        if maternal is None or paternal is None:
            raise ValueError("Cannot perform likewise pairing on "
                             "'None' chromosome.")

        if not isinstance(maternal, Chromosome) or not
               isinstance(paternal, Chromosome):
            raise ValueError("Not a chromosome.  Cannot perform "
                             "likewise pairing.")

        if (str(maternal).upper() not in self._tree.chromosome_lookup
         or str(paternal).upper() not in self._tree.chromosome_lookup):
            raise ValueError("Cannot locate chromosome in lookup "
                             "dictionary.")

        m = self._tree.chromosome_lookup[str(maternal).upper()]
        p = self._tree.chromosome_lookup[str(paternal).upper()]

        m_left = Chromosome(m.left.a, m.left.b, m.left.is_left, True)
        m_right = Chromosome(m.right.a, m.right.b, m.right.is_left,
                             True)

        p_left = Chromosome(p.left.a, p.left.b, p.left.is_left, False)
        p_right = Chromosome(p.right.a, p.right.b, p.right.is_left,
                             False)

        return (m_left, p_left), (m_right, p_right)

    def _crosswise(self, maternal, paternal):
        """
        Computes the crosswise pairing of the given chromosomes.
```

```
        Parameters:
          maternal - The maternal chromosome.
          paternal - The paternal chromosome.

        Return:
          A tuple representing the crosswise pairing of 'maternal' and
          'paternal'.

        Throws:
          ValueError - If either chromosome is not specified or is not
                       found in the chromosome lookup dictionary.
        """

        if maternal is None or paternal is None:
            raise ValueError("Cannot perform crosswise pairing on "
                             "'None' chromosome.")

        if not isinstance(maternal, Chromosome) or not
               isinstance(paternal, Chromosome):
            raise ValueError("Not a chromosome.  Cannot perform "
                             "crosswise pairing.")

        if (str(maternal).upper() not in self._tree.chromosome_lookup
         or str(paternal).upper() not in self._tree.chromosome_lookup):
            raise ValueError("Cannot locate chromosome in lookup "
                             "dictionary.")

        m = self._tree.chromosome_lookup[str(maternal).upper()]
        p = self._tree.chromosome_lookup[str(paternal).upper()]

        m_left = Chromosome(m.left.a, m.left.b, m.left.is_left, True)
        m_right = Chromosome(m.right.a, m.right.b, m.right.is_left,
                             True)

        p_left = Chromosome(p.left.a, p.left.b, p.left.is_left, False)
        p_right = Chromosome(p.right.a, p.right.b, p.right.is_left,
                             False)

        return (m_left, p_right), (m_right, p_left)

    def _compute_result(self, division=0):
        """
        Computes the result of cell division at the given division
        level.

        Parameters:
          division - The cell division to compute.

        Return:
          None.
        """

        num_results = 0

        # Compute the result given the cell division desired.  The cell
        # division value is incremented by one since the range function
        # is zero indexed.
```

```
for div in range(division + 1):

    # Save the current result if it exists.
    try:
        os.rename(self._curr_pickle, self._PREV_PICKLE)
    except OSError:
        pass

    # If the cell division to compute is zero, the result is
    # always the root node.
    if div == 0:
        num_results = 1
        with open(self._curr_pickle, "wb") as curr_f:
            pickle.dump(num_results, curr_f)
            pickle.dump(self._tree.root, curr_f)

    # If the first cell division to compute is one, the result
    # is trivially calculated.
    elif div == 1:
        num_results = 2
        with open(self._curr_pickle, "wb") as curr_f:
            pickle.dump(num_results, curr_f)
            pickle.dump(self._first_likewise, curr_f)
            pickle.dump(self._first_crosswise, curr_f)

    # Perform the cell division using the previous division's
    # result.
    else:

        # Calculate the number of entries in this division.
        num_results *= 2 ** (2 ** (div - 1))

        # Open the current and previous pickle files for
        # reading.
        with open(self._curr_pickle, "wb") as curr_f, \
                open(self._PREV_PICKLE, "rb") as prev_f:

            # The first line in a pickle file will be the
            # number of results.
            pickle.dump(num_results, curr_f)

            # Using the previous results, perform the likewise
            # and crosswise pairings.
            for _ in range(pickle.load(prev_f)):

                # Load a previous result.
                result = pickle.load(prev_f)

                # Perform likewise and crosswise pairings for
                # all combinations.
                for combos in itertools.product([False, True],
                                                repeat=2 ** (div - 1)):
                    new_result = []
                    for i, is_crosswise in enumerate(combos):
                        cell = result[i]

                        # Save the left and right cells.
```

```
                                left = cell[0]
                                right = cell[1]

                                # Compute the likewise and crosswise
                                # pairings.
                                if is_crosswise:
                                new_result.extend(self._crosswise(left,
                                                                  right))
                                else:
                                 new_result.extend(self._likewise(left,
                                                                  right))

                            # Save the new result as the current
                            # result.
                            pickle.dump(new_result, curr_f)

    def find_combinations(self, division=0):
        """
        Calculates then returns all cell combinations for the given
        cell division.

        Parameters:
          division - The cell division to compute (default 0).

        Return:
          A generator to the result of performing a cell division at
          the given cell division level.

        Throws:
          TypeError - If the given division is not an integer.
          IOError - If there was an issue opening the current pickle
                    file.
        """

        # Sanity check.
        if division < 0:
            division = 0

        # Generate the chromosome tree and compute the result.
        self._generate_tree(division)
        self._compute_result(division)

        # Return the results.
        try:
            with open(self._curr_pickle, "rb") as f:
                for _ in range(pickle.load(f)):
                    yield pickle.load(f)
                    # print(pickle.load(f))
        except IOError as ioe:
            raise ioe
```

File #4: cell_division.py

```
#!/usr/bin/env python

from __future__ import print_function
```

```
from cellutils.chromo_combo import ChromoCombo

import os
import sys
import getopt

# The developer intended program name.
DEV_PROGRAM_NAME = "cell_division.py"

# Save off the program name.
PROGRAM_NAME = os.path.basename(sys.argv[0])

# The pickle file containing the results.
PICKLE_FILE = "results.pik"


def _usage():
    """
    Prints usage information to standard output.

    Parameters:
      None.

    Return:
      None.
    """
    print("""\
Usage: {0} [OPTION] DIVISION
Calculates the result of performing a cell division DIVISION amount of
times.

Options:
  -h, --help                display this help message and exit
      --version             display version information and exit
  -o, --output=FILE         save the cell division results to FILE
                              (does not print to standard output)

Results are generated in the working directory in a Python pickle file
as 'results.pik', and are printed to standard output unless '--match'
is set.

Use the '--output' option if a configuration option is set and the
results are wanted in a file.

Examples:
  {1} 3
      Perform the third cell division.

  {2} -o results.txt 2
      Perform the second cell division, saving the results to
      'results.txt'."""
          .format(PROGRAM_NAME, PROGRAM_NAME, PROGRAM_NAME))


def _version():
    """
    Prints version information to standard output.
```

```
    Parameters:
      None.

    Return:
      None.
    """
    print("""\
{0} 1.0
Written by Carol Jim.""".format(DEV_PROGRAM_NAME))


def main():
    """
    The main program entry point.

    This function gathers and processes user input and initiates the
    cell division calculations.

    Parameters:
      None.

    Return:
      None.
    """

    # The working program name.
    prog = sys.argv[0]

    # Holds the file to save the results to (None means output to
    # standard output).
    save_in_file = None

    try:
        opts, args = getopt.getopt(sys.argv[1:], "ho:m",
                                   ["help", "version", "output=", "match"])
    except getopt.GetoptError as goe:
        print("{0}: {1}".format(prog, str(goe)), file=sys.stderr)
        print("Try '{0}' --help for more information.".format(prog),
              file=sys.stderr)
        sys.exit(1)

    # Parse the options.
    for opt, arg in opts:
        if opt in ("-h", "--help"):
            _usage()
            return 0
        elif opt in "--version":
            _version()
            return 0
        elif opt in ("-o", "--output"):
            save_in_file = arg
        else:
            print("{0}: unhandled option {1}".format(prog, opt),
                  file=sys.stderr)
            print("Try '{0}' --help for more
                  information.".format(prog), file=sys.stderr)
```

```
            return 1

    # Sanity check we have been given a single DIVISION value.
    if len(args) != 1:
        print("{0}: expected 1 argument, given {1}".format(prog,
                                                           len(args)),
              file=sys.stderr)
        print("Try '{0}' --help for more information.".format(prog),
              file=sys.stderr)
        return 1

    # The desired cell division level.
    div = int(args[0])

    # Sanity check the DIVISION value.
    if div < 0:
        print("{0}: negative division value".format(prog),
              file=sys.stderr)
        print("Try '{0}' --help for more information.".format(prog),
              file=sys.stderr)
        return 1

    # Compute and display the possible cell divisions at the given
    # DIVISION level.
    try:
        c = ChromoCombo(PICKLE_FILE)

        # If the output option was set, print the results to the file
        # specified.
        if save_in_file is not None:
            with open(save_in_file, "w") as f:
                for z in c.find_combinations(div):
                    print(z, file=f)
        else:
            for z in c.find_combinations(div):
                print(z)

    except ValueError as ve:
        print("{0}: {1}".format(prog, str(ve)), file=sys.stderr)
        print("Try '{0}' --help for more information.".format(prog),
              file=sys.stderr)
        return 1
    except TypeError:
        print("{0}: invalid division value".format(prog),
              file=sys.stderr)
        print("Try '{0}' --help for more information.".format(prog),
              file=sys.stderr)
        return 1
    except IOError as ioe:
        print("{0}: {1}".format(prog, str(ioe)), file=sys.stderr)
        return 1

if __name__ == '__main__':
    sys.exit(main())
```

Appendix B: Raw Sample Results

The following is a sample of raw output of the cell labeling combinations that result from the second cell division of the chromosomes labeling scheme after running the program shown in Appendix A. There are a total of eight possible combinations in the second division. Each grouping within parenthesis, i.e. `([X, Y], [x, y])`, represents a pair of homologous chromosomes in one cell.

```
[([A, B'], [a, b']), ([A'', B'], [a'', b']), ([A', B''],
[a', b'']), ([A', B], [a', b])]

[([A, B'], [a, b']), ([A'', B'], [a'', b']), ([A', B''],
[a', b]), ([A', B], [a', b''])]

[([A, B'], [a'', b']), ([A'', B'], [a, b']), ([A', B''],
[a', b'']), ([A', B], [a', b])]

[([A, B'], [a'', b']), ([A'', B'], [a, b']), ([A', B''],
[a', b]), ([A', B], [a', b''])]

[([A, B'], [a', b'']), ([A'', B'], [a', b]), ([A', B''],
[a, b']), ([A', B], [a'', b'])]

[([A, B'], [a', b'']), ([A'', B'], [a', b]), ([A', B''],
[a'', b']), ([A', B], [a, b'])]

[([A, B'], [a', b]), ([A'', B'], [a', b'']), ([A', B''],
[a, b']), ([A', B], [a'', b'])]

[([A, B'], [a', b]), ([A'', B'], [a', b'']), ([A', B''],
[a'', b']), ([A', B], [a, b'])]
```

Appendix C: Program to Generate Asymmetric Labeling Model

The program to generate the asymmetric labeling scheme can be found here. The differences between generating this scheme and the chromosomes labeling scheme occur when populating the nodes in the tree in the `populate(self, root)` function. The cell labeling combinations for the asymmetric labeling scheme can be generated using the program files found in Appendix A for the chromosomes labeling scheme.

```
"""
Provides methods for generating and iterating over a chromosome tree.

Author: Carol Jim
"""

from __future__ import print_function
from chromosome import Chromosome


class Tree:
    """Represents a chromosome tree."""

    def __init__(self, depth=0):
        """
        Construct and initialize a new chromosome tree.

        Parameters:
          depth - The depth of the tree to generate (default 0).
        """

        # Holds root of the chromosome tree.
        self.root = Chromosome(0, 0, False)

        # Holds a lookup table of nodes and their children.
        self.chromosome_lookup = dict()

        # Populate the tree according to the depth specified.
        self._populate_to_depth(depth)

    def _populate(self, root):
        """
        Populate the next depth of the chromosome tree using
        recursion.

        Parameters:
          root - The root of the chromosome tree to populate.

        Return:
          None.
        """
        if (root.left is None) and (root.right is None):
            if root is self.root:
                self.root.left = Chromosome(0, 0, True)
                self.root.right = Chromosome(1, 0, False)
            elif root.is_left:
                root.left = Chromosome(root.a, root.a, True)
```

```
                root.right = Chromosome(root.b + 1, root.b, False)
            else:
                root.left = Chromosome(root.a, root.a, True)
                root.right = Chromosome(root.a, root.b, False)

            return

        self._populate(root.left)
        self._populate(root.right)

    def _inorder_traversal_print(self, root):
        """
        Traverse the chromosome tree in-order, displaying its
        contents to standard output.

        Parameters:
          root - The root of the chromosome tree to traverse.

        Return:
          None.
        """
        if root is None:
            return
        self._inorder_traversal_print(root.left)
        print(root)
        self._inorder_traversal_print(root.right)

    def _populate_lookup_table(self, root):
        """
        Traverse the chromosome tree in-order, populating the node
        lookup table.

        Parameters:
          root - The root of the chromosome tree to traverse.

        Return:
          None.
        """
        if root is None or root.left is None or root.right is None:
            return
        self._populate_lookup_table(root.left)
        self.chromosome_lookup[str(root)] = root
        self._populate_lookup_table(root.right)

    def _populate_to_depth(self, depth):
        """
        Populates the specified amount of depths of the chromosome
        tree.

        If the amount of depths specified is <= 0, this function does
        nothing.

        Parameters:
          depth - The amount of depths to populate the chromosome tree.

        Return:
          None.
```

```
        """

        # Populate the depths and generate the lookup table.
        for n in range(depth):
            self._populate(self.root)
        self._populate_lookup_table(self.root)

    def inorder_traversal_print(self):
        """
        Traverse the chromosome tree in-order, displaying its contents
        to standard output.

        Parameters:
          None.

        Return:
          None.
        """
        self._inorder_traversal_print(self.root)

    def display_lookup_table(self):
        """
        Displays the lookup table to standard output.

        Parameters:
          None.

        Return:
          None.
        """
        for k, v in self.chromosome_lookup.iteritems():
            print("Root '" + k + "': " + str(v.left) + "  |  " +
                  str(v.right))
```