Technological Communication

Biosc.Biotech.Res.Comm Vol 12 (4) 1150-1158 (2019)

BBRC
Bioscience Biotechnology Research Communications

Modified Direct Differential Coding Using 2D-Dynamic Dictionary for Nucleotide Sequence

Raju Bhukya

Department of Computer Science and Engineering, National Institute of Technology,Warangal, India.

ABSTRACT

Bioinformatics is the application of computer technology for the management of biological information. As the exponential growth of DNA sequence is very big in size, ranging from 3 to 50 GB or more depending on the species. Being large in size, molecular data imposes restrictions in searching and retrieval of sequences. While modern hardware provides vast amounts of inexpensive storage. The compression of biological sequence data is still important in order to facilitate fast search and retrieval operations by reducing the original size of the sequence. The Differential Direct Coding technique is DNA sequence data compression algorithm that identifies sequence data and auxiliary data by including the extra codes for the symbols that are other than the set of nucleotide bases. The Differential Direct Coding algorithm is a single phase algorithm which removes the overhead of handling wildcard symbols. Here the algorithm works on construction dictionary based coding technique that provides better compression ratio in spite of little storage space with the reference of Differential Direct Coding algorithm.

KEY WORDS: AUXILIARY DATA, BIOINFORMATICS, COMPRESSION RATIO, DIFFERENTIAL DIRECT CODING.

Article Information:*Corresponding Author: raju@nitw.ac.in
Received 10/10/2019 Accepted after revision 18/12/2019
Published: 30th Dec 2019 Pp- 1150-1158
This is an open access article under Creative Commons License,
Published by Society for Science & Nature, Bhopal India.
Available at: https://bbrc.in/
Article DOI: http://dx.doi.org/10.21786/bbrc/12.4/39

INTRODUCTION

Bioinformatics is a multidisciplinary science that uses methods, principles from mathematics, computer science for analyzing the biological data. The computer databases and algorithms are developed to speed up and enhance biological research. A fundamental requirement for research in bioinformatics is the capacity to warehouse large amounts of biological sequence data (Bekas, Konstantinos.,2019). Genomic repositories contain a large amount of data, due to this some efficient algorithms have emerged to facilitate communication and storage issues. Out of this, cheap storage is not a big issue as due to limited bandwidth communication of modern world (Salomon and Motta 2010). The amount of DNA sequenced from organisms is increasing rapidly (Cao et al 2007). Sequencing initiatives are contributing exponentially increasing quantities of nucleotide data to databases such as GenBank (Williams, et.al., 1997, Benson, et.al., 2008).

As of January 2009, the Nucleic Acids Research online Molecular Biology Database Collection listed 1170 publicly available biological databases(Galperin and Cochrane, 2008). GenBank, a major sequence database and a component of the International Nucleotide Sequence Databases (INSD), doubles in size roughly et al. 2008). In January 2013 the real cost of sequencing human-size genome (according to NHGRI data) was about 5,700 dollars, while the cost of one-year storage at Amazon S3 and 15 downloads of raw reads and mapping results (225 GB of reads with 30-fold coverage and 500 GB of mapped data) was close to 1,500 dollar (Deorowicz and Grabowski 2013).

Compression of biological sequences is useful, not primarily for managing the genome database, but for modeling and learning about sequences (Gauthier, Jeff, et al., 2018). Since DNA is the "instruction of life", it is expected that DNA sequences are not random and can be compressible. While some DNA sequences are highly repetitive. A repeat subsequence is a copy of a previous subsequence in the genome, either forward or reverse complement. Most DNA repeats are not exact as nucleotides can be changed, inserted or deleted. It is estimated that 55% of the human genome is repeat DNA. Because of the particularity of the DNA sequence data, it is not very ideal with traditional compression algorithms, (Cao et al 2007). There are many methods to achieve the compression of the data, (Das et.al, 2005, Chen et.al. 2001,Galperin et.al., 2008 Jifeng, et.al. 2012, Cao et.al., 2007). This research particularly concentrate on the table method(dictionary method).

There exist many algorithms based on dictionary method like Ziv-Lampel (Vey, 2009, Das et al 2005). Also some other arithmetic encoding algorithms also like Huffman algorithm. However, these universal text compression algorithms are not suitable for compression of biological sequences as they consider the sequence as a pure text stream. Nucleotide symbols deal with four symbols representing nucleotide bases {A, C, T, G}. Earlier, many text compression techniques have been applied on biological data, which are less efficient. Interestingly, most general purpose text compression algorithms fail to compress DNA to below the naive 2 bits per symbol. That is because DNA regularities are different from those in text and are rarely

Table 1. Differential Direct Coding (2D) with Dictionary Model

Type of Data	Description	Range	Dictionary Part
Auxiliary Symbol (Wildcards)	ASCII	0 to 127	
Triplets	Set of three base characters	-1 to -64	Fixed Part
Multiple of Triplets	Set of 3,6,9,12, base characters	-65 to -127	Dynamic Part

modeled by those compressors(Cao et al 2007). Thus there appeared compression algorithms for DNA sequence specially. A number of special purpose compression algorithms for DNA have been developed recently. The algorithms can be divided into two kinds: based on substitution and based on statistic (Jifeng 2012). The former class replace a long repeated subsequence by a pointer to an earlier instance of the subsequence or to an entry in a dictionary. Examples of this category are the popular Lempel-Ziv compression algorithms and their variants (Laird, Sarah, and Rachel Wynberg 2018)

On the other hand, a statistical compression encoder such as prediction by partial match (PPM) (Lipmann et al. 2008) predicts the probability distribution of each symbol(Cao et al 2007). The Huffman coding (Salomon et al 1952) invented in 1952, is a statistical method, which assigns a sequence of bits (a codeword) to each alphabet symbol. The code words are of different length, and in accordance to the golden rule of data compression, rarer symbols are represented by longer code words. The given sequence is then encoded by replacing each symbol with its corresponding codeword. Huffman coding importance is its optimality, i.e., no other code leads to a shorter encoded sequence. In 1977–78 Ziv and Lempel (Salomon Motta, 2010, Ziv and Lempel 1977) invented dictionary methods. They have processed the sequence from left to right and encode possibly long repetitions of consecutive symbols as references to the already compressed part of data. Such an approach allows for higher compression ratios than Huffman coding as it looks for another type of redundancy. Not only in natural language (e.g., repeating word phrases), but also in multiple genome sequences or overlapping sequencing reads. Even better results are possible with combining dictionary methods and Huffman coding. While modern hardware can provide vast amounts of inexpensive storage, the compression of biological sequence data is still of paramount concern in order to facilitate fast search and retrieval operations, primarily by reducing the number of required I/O operations, (Galperin and Cochrane, 2008). Data compression requires two fundamental processes: modelling and coding (Williams and Zobel, 1996).

Table 2. Dictionary Table

Triplet	ASCII Value
AAA	1
AAC	2
-----	---
TTT	64
AAAAAC	65
AACCGT	66
------	---
AACCGTGTA	127

Modeling involves constructing a representation of the distinct symbols in the data, along with any associated data, like the relative frequencies of the symbols (Williams and Zobel, 1996). Coding involves applying the model to each symbol in the data to produce a compressed representation of the data, preferably by assigning short codes for frequently occurring symbols and long codes to infrequently occurring symbols (El Naqa et al. 2018). In the case of DNA sequences, the finite set of nucleotide symbols {A, C, G, T} can be efficiently modeled as a corresponding set of binary values {00, 01, 10, 11}(Williams and Zobel, 1996). This model constitutes an effective binary representation where each nucleotide base is directly coded by two bits. This assumes that sequence data are indeed composed solely from the four symbols of the nucleotide set. However, this assumption is not guaranteed to be met and a nucleotide sequence may include additional wildcard symbols, like N or S (Williams and Zobel 1996).

Therefore, to reconcile the potential occurrence of symbols other than the expected four nucleotide bases, any unexpected symbol is randomly converted into one of the valid symbols that it represents (Williams and Zobel 1996). Eliminated wildcards are subsequently restored during sequence decompression (Williams and Zobel 1996). This makes the process as 2 phase process. This paper presents substitution based modified direct differential coding algorithm with dynamic dictionary. It is a nucleotide sequence data compression algorithm based on substitution technique with the help of dynamic dictionary.

This algorithm considers triplets and sequences of length multiple of three and it is stored in fixed part and dynamic part of the dictionary. This algorithm is also capable to handle symbols other than four nucleotide bases. It provides better performance as compared to existing algorithm. The Differential Direct Coding is designed by considering fixed size dictionary i.e. it contains 64 dictionary entries for possible combinations of A, C, G and T with set of 3 characters and multiples of it. We have modified the dictionary to contain more than 64 entries where first 64 entries are for triplets and next entries are for sequences of length multiple of 3.

MATERIALS AND METHODS

Initially the method identifies repeat regions multiple of three in the individual sequence and the repeat regions are store in the dictionary table. The proposed algorithm compresses both repeat and non repeat sequences. It also handles the non base character and compresses any nucleotide sequence. If the case of formation of triplets is considered, with combination of four symbols {A, C, G, T} or {A, C, G, U} for DNA or mRNA respectively, then it encounter maximum 64 combinations. These 64 triplet combinations handled by 64 non-printable ASCII characters, whereas there exist total 127 non-printable ASCII characters. Therefore, the remaining 63 characters are used to store some other combinations of size more than 3, which can yield a better compression.

Model: We consider the ASCII characters between the ranges 0 to 127 for the Auxiliary symbols. The ASCII values from -1 to -64 are used for triplets of {A, C, G, and T} or {A, C, G, and U}. The ASCII values from -65 to -127 are used for the sequences which are multiples of 3. The first 64 entries of dictionary are fixed length entries, i.e. sequence of length 3. The entries from 65 to 127 are dynamic length entries, i.e. sequences of length multiple of 3 as shown in Table1. The sample dictionary view is shown in Table. 2.2. It contains 127 entries for different triplets and patterns of length multiple of three. The first 64 entries are for triplets only and remaining upto 127 are for patterns of length multiple of three.

Compression procedure: Apply the model described above, for the encoding of input sequence. While scanning the input sequence the algorithm always searches for the triplet from the fixed size Dictionary Table. It will further scan next triplet therefore the total scanned characters will be of length 6 now. If this combination of 6 characters is available in the dynamic part dictionary table, it will scan next triplet, otherwise it will write the ASCII character corresponding to the last match group in the output file and also insert the current combination into the dynamic part of dictionary table.

This process will be carried on till the whole sequence is encoded. Once the dictionary entries are filled completely, i.e. all 127 entries are filled; the algorithm will not add any more entries in the dictionary and just search for the combinations from both parts of the table. Also this will be followed by writing the ASCII value corresponding to that combination to the output file. The output file is the required compressed file, when the whole sequence is scanned. While scanning the input sequence, in some cases may occur where a triplet cannot be formed. Here the algorithm writes single character or a group of two into the output file as it is. Whenever a wildcard symbol is obtained, the algorithm stops scanning the triplets and writes wildcard character's ASCII encoded equivalent, followed by its positive integral repeats in its decimal form.

Algorithm: Differential Direct Coding (2D) Compression with Dynamic Dictionary Approach

Input: DNA Sequence data file.
Output: Compressed DNA Sequence data file.
Note: Fixed part of dictionary table is preprocessed.

```
Initially T=NULL, TP=NULL,SEQ=NULL;
Step 1:    Read first triplet (T).
IF (T!= NULL)
go to step 2.
ELSE IF (!triplet)
Go to step 5.
END IF
Step 2: Check that T has all nucleotide symbols.
```

```
IF (true)
goto step 3.
ELSE
IF (T has wildcard Symbol)
go to step 4
ELSE
go to step 5.
END IF
END IF
Step 3: IF (TP exists in the Dictionary Table)
SEQ=TP
ELSE
Output the code (character) for s
Add ST to the Dictionary Table;
SEQ=T;
END IF
Step 4:
a. Search wildcards and get count of successive wildcards.
b. Replace it with wildcard and count
c. Go to step 6.
Step 5: IF (! triplet)
Write remaining symbols
Go to step 6.
END IF
Step 6: Return to step 1 if EOF is not reached.
```

Decompression procedure: Decoding phase replaces each signed byte from compressed file with corresponding triplet or sequence of length multiple of three. Decoding starts with reading a byte by byte compressed file. Each read byte is checked whether it is a positive signed byte or negative signed byte. Suppose it is positive signed byte, then it is clear that particular byte is representing an uncompressed value which might be any of wildcard symbols. In this case write that particular byte as it is in output file, and read next byte from compressed sequence file. If read byte is representing a negative byte value then it is clear that MSB of the byte is set which is representing compression bit. It means that particular byte is representation for sequence. Check the dictionary entries against read byte and retrieve the sequence against that byte.

Write that sequence to output file and read next byte from compressed file. Whenever read byte is positive byte, it is representation of wildcard symbol and wildcard symbol follows the decimal number representing number of occurrences of that symbol and write these many numbers of symbols in output file.

Algorithm

```
Differential Direct Coding (2D) Decompression with Dynamic
Dictionary Approach
Input: Compressed DNA File.
Output: Decompressed Sequence File.
Note: Fixed part of dictionary table is preprocessed.
Step1: WHILE (!EOF)
Read byte b from input file and go to step 2.
IF (EOF)
go to step 5.
IF END
Step 2:    If (b is positive)   // (Representing Wildcard Symbol)
Read next bytes representing digits and store in an array arr
Convert array arr into integer i  and goto Step 3
Else,
go to step 3.
END IF
Step 3:    FOR(0 to  i)
Write wildcard symbol to output file.
END FOR
Go to Step 1
Step4: Search the Dictionary table for a value –(b) and retrieve corresponding triplet t.
Write triplet t in an output file and go to step 1.
END WHILE
Step 5: Stop.
```

Example of Differential Direct Coding Compression

Input Sequence: TGACGATGACGATTTNNNAG

Step 1: Read first triplet t=TGA and check for presence in the dictionary. It is present. Then read next triplet.

Step 2: Next triplet t=CGA, and check for presence in the dictionary. It is present so add earlier triplet TGA and CGA to form sequence TGACGA. Check for presence in the dictionary. It is not present in dictionary.

Add TGACGA=$ in the dictionary.

Output for earlier pattern i.e. TGA=#.

Step 3: Read next triplet t= TGA and check for presence in the dictionary. It exists, add earlier triplet CGA and TGA to form sequence CGATGA. Check for presence in dictionary. It is not present in dictionary.

Output for earlier pattern i.e. CGA =+. Add CGATGA =@ in the dictionary.

Step 4: Read next triplet t= CGA and check for presence in dictionary. It is present so add earlier triplet TGA and CGA to form sequence TGACGA. Check for presence in dictionary. It is present in the dictionary.

Read next triplet t=TTT and earlier pattern TGACGA and TTT to form TGACGATTT. Check for presence in dictionary. It is not present in the dictionary.

Add TGACGATTT=& in the dictionary.
Output for earlier pattern i.e. TGACGA=$.
Step 5: Next symbol is non-nucleotide (wildcard) symbol so write output for earlier triplet.
Output earlier pattern i.e. TTT=ˆ.

Step 6: Read wildcard symbols until nucleotide is encountered. Put wildcard symbol with count of occurrence.
Output NNN= N3.
Step 7: Read next triplet t=AG. We got less symbols for triplet, so write as it is to output file.
Output = AG.
Compressed Sequence: #+$ˆN3AG

As the input sequence size is of 20 characters in size whereas the compressed file size is of 8 characters. The compression ratio depends upon the type of the sequence.

Example of the Decompression
Compressed Sequence: #+$ˆN3AG
Step 1: Read first byte b=# and check for positive or negative byte value. It is negative, so put corresponding sequence form dictionary to output file.

Output=TGA.
and go to the next step.
Step 2: Read next byte b=+ and check for positive or negative byte value. It is negative, so put corresponding sequence form dictionary to output file.

Table 3. Table of 2D with Dictionary Approach

Input Sequence	Current Triplet	Sequence of length multiple of 3	Dictionary Table		Output Symbol
			Status of Triplet In Dictionary	Corresponding Entry from Dictionary	
TGACGATGAC GATTTNNNAG TGA	TGA	TGA	Found	TGA = #	
	CGA	TGACGA	Not Found	Add with TGACGA = $ And output TGA=#	#
CGA	CGA	CGA	Found	CGA=+	
	TGA	CGATGA	Not Found	Add CGATGA=@ And output CGA=+	#+
TGA	TGA	TGA	Found	TGA=#	
	CGA	TGACGA	Found	TGACGA=$	
	TTT	TGACGATTT	Not Found	Add TGACGATTT=& and output TGACGA=$	#+$
TTT	TTT	TTT	Found	TTT=ˆ (Next is wildcard)	#+$ˆ
NNN					#+$ˆN3
AG		<3		#+$ˆN3AG	

Output=CGA.
Step 3: Read next byte from b=$ and check for positive or negative byte value. It is negative, so put corresponding sequence form dictionary to output file.

Table 4. Experimental Results of 2D with Dynamic Dictionary with the 2D algorithm

Source Genome	Input File Size (bytes)	2D Algorithm				2D with Dynamic Dictionary(proposed)			
		Compre-sseed File Size	Compre-ssion Ratio	Compress-ion Time (ms)	Decomp-ression Time (ms)	Compress-ed File Size	Compre-ssion Ratio Time (ms)	Compress-ion Time (ms)	Decomp-ression Time (ms)
Bacillus Subtilis	4274764	1404994	3.0425	64466	34945	1376213	3.1061	64631	31741
Escherichia Coli K12 MG1655	4705920	1567898	3.0014	70950	38678	1513218	3.1098	70646	37372
Mycoplasm Genitalium G37	588365	193362	3.0428	5362	4683	185424	3.1730	8845	4267

Output=TGACGA.
Step 4: Read next byte from b=ˆ and check for positive or negative byte value. It is negative, so put corresponding sequence form dictionary to output file.

Output=TTT.
Step 5: Read next byte from b=N and check for positive or negative byte value. It is positive and a wildcard symbol, so read next decimal number i.e. 3 and write these many Ns to output file.

Output=NNN.
Step 6: Read next byte b=A and check for positive or negative. It is positive but not a wildcard, so write as it is to output file.

Output=A
Step 7: Read next byte b=A and check for positive or negative. It is positive but not a wildcard, so write as it is to output file.

Output=G
Decompressed Sequence:
TGACGATGACGATTTNNNAG

RESULTS AND DISCUSSION

Here, by referring the Differential Direct Coding (2D) algorithm, we have proposed the algorithm "Differential Direct Coding (2D) based on Dynamic Dictionary Approach", which is working on triplets of nucleotides and patterns of length multiple of three. We have modified the dictionary table of Differential Direct Coding algorithm to divide it in two parts as fix part and dynamic part. The performance of proposed algorithm is compared with base algorithm. The results table contains the original input file size in bytes, compressed file size in bytes by 2D algorithm, the compression ratio for 2D algorithm. It also contains compressed file size in bytes by 2D with Dynamic Dictionary algorithm, the compression ratio for 2D Dynamic Dictionary algorithm. The compression ratio is derived from following formula-

Compression Ratio = Original File Size / Compressed File Size.

It was observed that 2D with Dynamic Dictionary is producing an output file of small size as compared to 2D algorithm. For Bacillus subtilis 2D is generating output file of size 1404994 bytes resulting a compression ratio 3.0425. Whereas 2D with Dynamic Dictionary is generating an output file of size 1376213 which is 28781 bytes less than 2D output file resulting a compression ratio 3.1061. For Escherichia coli K12 MG1655 2D is generating output file of size 1567898 bytes resulting a compression ratio 3.0014. Whereas 2D with Dynamic Dictionary is generating output file of size 1513218 which is 54680 bytes less than 2D output file resulting a compression ratio 3.1098. For Mycoplasm genitalium G37 2D is generating output file of size 193362 bytes resulting a compression ratio 3.0428. Whereas 2D with Dynamic Dictionary is generating output file of size 185424 which is 7938 bytes less than 2D output file resulting a compression ratio 3.1730.

In the similar grounds the decompression ratios of 2D with dynamic dictionary algorithm when compared with 2D algorithm shows a significant decrease in all the three source genome types mentioned i.e., in Bacillus subtilis there is '9.1%' decrease in compression ratio,in Escherichia coli K12 MG1655 nearly '3.5%' decrease and in Mycoplasm genitalium G37 '8.9%' decrease is observed.The results from Table 4.1 clearly indicate better compression results for all three input files. The compression ratio for 2D with Dynamic Dictionary is also better than compression ratio for 2D. High compression ratio also suggests a highly repetitive sequence.

Thus proposed algorithm, "Differential Direct Coding (2D) with Dynamic Dictionary Approach", has high compression ratio to other existing DNA Sequence Compression algorithm, Differential direct coding: a compression algorithm for nucleotide sequence data. The Table 4.1 shows the execution time of proposed algorithm and Differential Direct Coding (2D) algorithm for three different input files. The execution time from table is the average of ten execution times. From Table 4.1 it is clear that the compression time for proposed algorithm is more than Differential Direct Coding algorithm for all input files. Decompression time of proposed algorithm for Bacillus subtilis and Escherichia coli K12 MG1655 is more than Differential Direct Coding algorithm, whereas for Mycoplasm genitalium G37 it is more than Differential Direct Coding algorithm.

CONCLUSION

A new DNA sequence data compression algorithms by dynamic dictionary approach technique have proposed. Here direct coding on triplet basis technique for modeling and encoding purposed. An experimental result gives efficient results compared with existing techniques and reduces the compression as well as the execution time. From these experiments, we conclude that algorithm "differential direct coding with dynamic dictionary approach" is improved comparing to differential direct coding (2d) algorithm. For improvements, we have partitioned dictionary into two parts as 'fix dictionary table' and 'dynamic dictionary table'.

REFERENCES

Vey, G., (2009) Differential direct coding: a compression algorithm for nucleotide sequence data. Database, 2009.

Williams, H. and Zobel, J., (1997)Compression of nucleotide databases for fast searching. Bioinformatics, 13(5), pp.549-554.

Das, D., Kumar, R. and Chakrabarti, P(2005) January. Dictionary based code compression for variable length instruction encodings. In 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (pp. 545-550). IEEE.

Chen, X., Kwong, S. and Li, M., (2001) A compression algorithm for DNA sequences. IEEE Engineering in Medicine and biology Magazine, 20(4), pp.61-66.

Galperin, M.Y. and Cochrane, G.R., 2008. Nucleic acids research annual database issue and the NAR online molecular biology database collection in 2009. Nucleic acids research, 37(suppl_1), pp.D1-D4.

Jifeng Sun,Wenping Xiong Li Tan, A(2012) Compression Algorithm for DNA Sequence Using Extended Operations. Journal of Computational Information Systems 8: 18 (2012) 7685–7691

Cao, M.D., Dix, T.I., Allison, L. and Mears, C., (2007)March. A simple statistical algorithm for biological sequence compression. In 2007 Data Compression Conference (DCC'07) (pp. 43-52). IEEE.

Benson, D.A., Karsch-Mizrachi,I., Lipman,D.J. et al. (2008), GenBank. Nucleic Acids Res., 36, D25–D30.

Williams, H. and Zobel, J., (1996). Practical compression of nucleotide databases. Australian CompP uter science Communications 18, pp.184-192.

Cao, M.D., Dix, T.I., Allison, L. and Mears, C., (2007)March. A simple statistical algorithm for biological sequence compression. In 2007 Data Compression Conference (DCC'07) (pp. 43-52). IEEE.

Galperin, M.Y. and Cochrane, G.R., (2008) Nucleic acids research annual database issue and the NAR online molecular biology database collection in 2009. Nucleic acids research, 37(suppl_1), pp.D1-D4.

Salomon D, Motta G,(2010) Handbook of data compression. London: Springer.

Huffman, D.A.(1952). A method for the construction of minimum-redundancy codes. Proceedings of the IRE, 40(9), pp.1098-1101.

Ziv, J. and Lempel, A(1977). A universal algorithm for sequential data compression. IEEE Transactions on information theory, 23(3), pp.337-343.

Deorowicz, S. and Grabowski, S.(2013)Data compression for sequencing data. Algorithms for Molecular Biology, 8(1), p.25.

Bekas, Konstantinos, Maria Gabrani and Antonio Foncubierta Rodriguez. (2019)Extract information from molecular pathway diagram. US. Patent No. 10,360,993. 23 Jul. 2019.

Gauthier, Jeff, et al. (2018) A brief history of bioinformatics. Brief. Bioinform 3 (2018).

Laird, Sarah, and Rachel Wynberg. (2018) A Fact-Finding and Scoping Study on Digital Sequence Information on Genetic Resources in the Context of the Convention on Biological Diversity and the Nagoya Protocol Secretariat of CBD (2018): 2-79.

El Naqa, Issam, et al. (2018) Biological data: The use of-omics in outcome models. Guide to Outcome Modeling In Radiotherapy and Oncology. CRC Press, 2018. 53-62.