ARRANGING CROSS-INTERSECTING GENOMIC SEGMENTS FOR CIRCOS-TYPE VISUALIZATION

BY

TAN SEE YEE

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONS)

Faculty of Information and Communication Technology
(Perak Campus)

JANUARY 2018

DECLARATION OF ORIGINALITY

I declare that this report entitled "**ARRANGING CROSS-INTERSECTING GENOMIC SEGMENTS FOR CIRCOS-TYPE VISUALIZATION**" is my own work except as cited in the references. The report has not been accepted for any degree and is not being submitted concurrently in candidature for any degree or other award.

Signature : _____

Name : _____TAN SEE YEE_____

Date : _____

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and appreciation to my supervisor, Dr. Ng Yen Kaow for giving me this opportunity to collaborate with a Hong Kong City University student to take this project. His enthusiasm in teaching and diligence towards research help me to complete this project. Without his invaluable guidance and patience, this project would not possible to complete.

Thanks to WenLong, Jia and HeChen, Lee from City University of Hong Kong who has been giving me insights about this project. They are so helpful and always available when I don't understand the project.

To my parents for their unconditional love, support and constant encouragement, I would able to continue this project. Finally I would like to thanks to all who have directly or indirectly helped me in the completion of this project.

ABSTRACT

Arranging cross-intersecting genomic segments for circos-type visualization is a topic of this project. There are many types of circos-type visualization, such as CIRCOS and RCircos. In this project, we use new circos-type visualization to display the relationship between human genomic segments and virus segments. Lack of research on allocating the genomic segments in a circular form is the main reason to develop this project. The algorithm to arrange the genomic segments in circular form is needed for the genetic researcher to visualize their data. With the circos-type visualization, they are able to present their data in more understandable and interesting way. The user can make an overview in which of their chromosome had been infected by the virus. From this circos-type visualization, the user may know where the gene had been mutated, so this can help the medical scientist investigate the methods to treat cancer. The existing allocation problem only focuses on allocating the blocks in a rectangle area. Therefore, this project is carried out to find a better allocation solution on the circle. The algorithm developed will test by a program with the genomic data. At the end of this project, an algorithm for arranging the cross-intersecting genomic segments for circos-type visualization will be developed.

TABLE OF CONTENTS

TITLE…….....…… i
DECLARATION OF ORIGINALITY ii
ACKNOWLEDGEMENTS iii
ABSTRACT.....iv
TABLE OF CONTENTS.....v
LIST OF TABLES.....vii
LIST OF FIGURES viii
LIST OF ABBREVIATIONS xi
CHAPTER 1 INTRODUCTION.....1
1-1 Motivation and Problem Statement 1
1-2 Background information.....1
1-3 Project Scope 10
1-4 Project Objectives.....16
1-5 Impact, significance and contribution 17
1-6 Report Organisation.....18
CHAPTER 2 LITERATURE REVIEW.....19
2-1 Dynamic Storage Allocation 19
2-2 CIRCOS.....21
2-3 RCircos 25
CHAPTER 3 SYSTEM DESIGN 27
3-1 System Setup 27
3-2 Platform Overview 30
3-3 Data Loading 35
3-4 Data Pre-processing.....36
3-5 Arrange Genomic Segments.....46
3-6 Drawing 51
3-7 Interaction.....55
CHAPTER 4 METHODOLOGIES AND TOOLS.....64
4-1 Design Specifications 64
4-1-1 Methodologies and General Work Procedures.....64
4-1-2 Tool to use.....64
4-1-3 User requirements 65
4-1-4 System Performance Definition 65

4-1-5 Verification Plan 66
4-2 Implementation Issues and Challenges 66
CHAPTER 5 RESULT 67
5-1 Results 67
CHAPTER 6 CONCLUSION 71
6-1 Project Review 71
6-2 Discussion 71
6-3 Future Work 72
BIBLIOGRAPHY 73

LIST OF TABLES

Table Number	Title	Page
Table 1-2-T1	Patients with HPV16 cancer	2
Table 1-2-T2	Numbering of sample T003 (1)	3
Table 1-2-T3	Numbering of sample T003 (2)	4
Table 1-2-T4	Numbering of sample T005	5
Table 1-2-T5	Detail of genomic segments of each sample	6
Table 1-2-T6	Sample T019 genomic segments	6
Table 3-4-T1	Sub-Segments file (T003)	40
Table 3-4-T2	Sub-Haplotypes file (T003)	44
Table 3-4-T3	Sample T003 data after pre-processing	44
Table 3-4-T4	Reference segment “chr4”	44

LIST OF FIGURES

Figure Number	Title	Page
Figure 1-2-F1	Sequence problem	4
Figure 1-2-F2	Repeat of the genomic segments	7
Figure 1-2-F3	Replacing of the same genomic segments	7
Figure 1-2-F4	Part of circos-type visualization (sample T019)	8
Figure 1-2-F5	Circos-type visualization of all samples	9
Figure 1-2-F6	Local genomic map of sample T019	10
Figure 1-3-F1	Circos-type visualization of chr8	11
Figure 1-3-F2	Example of circos-type visualization of chr8	12
Figure 1-3-F3	Segment ‘1’ with interval *(a, b)* (1)	12
Figure 1-3-F4	Segment ‘1’ with interval *(a, b)* (2)	13
Figure 1-3-F5	Multiple segments with interval *(a, b)*	13
Figure 1-3-F6	Two segments in same track (1)	14
Figure 1-3-F7	Two segments in same track (2)	14
Figure 1-3-F8	Distance between the center of 2 segments (1)	15
Figure 1-3-F9	Distance between the center of 2 segments (2)	15
Figure 2-1-F1	DSA of two blocks	19
Figure 2-2-F1	Rules added to the Circos image	21
Figure 2-2-F2	Procedures to create Circos image	22
Figure 2-2-F3	Simple Circos image of human chromosome	23
Figure 2-2-F4	Different types of 2D data tracks	23
Figure 2-3-F1	RCircos image of human chromosome	25
Figure 3-2-F1	Dovirus framwork	30
Figure 3-2-F2	Administration page	30
Figure 3-2-F3	Add analysis page (part 1)	31
Figure 3-2-F4	Add analysis page (part 2)	32
Figure 3-2-F5	Dovirus dashboard	32
Figure 3-2-F6	Project page (part 1)	33
Figure 3-2-F7	Project page (part 2)	33

Figure 3-2-F8	Dovirus file	33
Figure 3-2-F9	Dovirus db folder contents	34
Figure 3-2-F10	Analysis folder with js module name	34
Figure 3-3-F1	Data interface	35
Figure 3-3-F2	Data mapping	35
Figure 3-3-F3	Data loading	35
Figure 3-4-F1	Arc description	36
Figure 3-4-F2	Segment description	36
Figure 3-4-F3	D3 arc angle	37
Figure 3-4-F4	Start and end angle of virus reference segment	37
Figure 3-4-F5	Placement of the host reference segments	38
Figure 3-4-F6	Angle after one round	39
Figure 3-4-F7	Track length computation	39
Figure 3-4-F8	Reference segment start position and end position after calculation	43
Figure 3-4-F9	Segment id “B” start position and end position after calculation	43
Figure 3-5-F1	Description of the distance between two segments	50
Figure 3-6-F1	Reference segment inner and outer radius	51
Figure 3-6-F2	Inter-track distance of 4 tracks	52
Figure 3-6-F3	Genomic segment arc thickness, h	53
Figure 3-6-F4	Example genomic segment arc	53
Figure 3-7-F1	Movement of a segment	55
Figure 3-7-F2	Track calculation	56
Figure 3-7-F3	Radius calculation	56
Figure 3-7-F4	Angle concept	57
Figure 3-7-F5	Distance, r computation	58
Figure 3-7-F6	Distance, D_{02} greater than distance, R	60
Figure 3-7-F7	Distance, D_{02} smaller than distance, R	60
Figure 3-7-F8	Radius of the tracks	62
Figure 5-1-F1	Result of the circos-type visualization	67
Figure 5-1-F2	Changes when click event is triggered	68

Figure 5-1-F3	Changes when double click event is triggered	68
Figure 5-1-F4	Segment is dragged to the track that occupied	69
Figure 5-1-F5	Reposition of segments	70

LIST OF ABBREVIATIONS

DSA	*Dynamic Storage Allocation*

CHAPTER 1 INTRODUCTION

1-1 Motivation and Problem Statement

Circos-type visualization is a popular method for presenting genomic data. The advantages of using circos-type visualization are the relationship between the genome can be easily explored and it is attracting. Without the circos-type visualization, the presentation that related to genetic will degrade and it is hard for the genetic researcher to interpret data. The problem is lack of resources in arranging the genomic segments for circos-type visualization.

The motivation to solve this problem is to create an algorithm that can help in solving the problem of the allocation of cross-intersecting genomic segments in a circular. It is helpful for the genetic researcher to present their genomic data in the more understandable way. Circos-type visualization provides the platform for the genetic researcher to show the overview connection of the genomic segments of human that infected by the virus. Thus, the medical scientist can get the information in order to investigate the methods to treat the cancer. Genomic segments of multiple patients will show in the circos-type visualization, which helps the user to learn about the relationship between host chromosome and virus.

1-2 Background information

What is genomic segment? Genomic segment is a region of the genome. It is different in size. Genome is a collection of DNA. DNA is deoxyribonucleic acid, used in development, growth, functioning in all living organisms including human and viruses. The structure of DNA is a double helix. It is made up of two strands which bound to one another. Each strand of DNA contains a chain of nucleotides. There are different nucleotides, each nucleotide with varieties nitrogenous base which are adenine(A), thymine(T), cytosine(C) and guanine(G). The base nucleotides are always in a pair (base pair) which adenine-thymine and cytosine-guanine.

Human genome has roughly 3 billions base pairs which are organized in the 23 pair chromosome. Each chromosome has around 500 to 4000 genes. Genes have three different part – promoter, exons and introns. Promoter located at the beginning of a gene. Introns are non-coding and composed 98% which do not carry any information to the formation of protein. Exons are coding that used to translate into the protein.

Exons transcript into RNA which is one strand that contains base pairs of DNA without introns. RNA is then translated into protein. A codon (3 nucleotides in RNA) translate into an amino acid which is building blocks of the proteins. Multiple amino acids form a protein. The structure of a protein is based on the sequence of the nucleotides in RNA.

Many situations can cause the gene mutation. Mutation of genes can cause the disease. It will change the structure of the proteins and cause the function cannot perform well. Single-nucleotide polymorphism (SNP) is a variation of the nucleotide in a specific position. For example, adenine(A) is incorrectly paired with guanine(G): A-G; in fact, it should be adenine(A) pair with thymine(T): A-T. It will cause the deformation of the proteins. After a long period of time, they will cause the disease. Human genome has roughly 3 billions base pairs which are organized in the 23 pair chromosome. Each chromosome has around 500 to 4000 genes. Genes have three different part – promoter, exons and introns. Promoter located at the beginning of a gene. Introns are non-coding and composed 98% which do not carry any information to the formation of protein. Exons are coding that used to translate into the protein.

#SampleID	haplotype_NO	colour	contig_NO	repeat_time	regid_string
T003	1	purple	1	1	A,
T003	1	purple	2	9	B,C,D,E,r_b,G,H,
T003	1	purple	3	3	B,E,r_b,G,H,
T003	1	purple	4	1	B,E,r_b,G,D,E,r_b,G,H,I,
T005	1	green	1	1	A,F,
T005	1	green	2	42	G,r_b,r_a,r_d,C,D,E,F,
T005	1	green	3	4	r_a,r_d,C,D,E,F,
T005	1	green	4	1	G,H,
T019	1	orange	1	1	A,B,C,D,
T019	1	orange	2	67	r_C,r_B,H,I,c,d,a,K,C,D,
T019	1	orange	3	4	r_C,r_B,H,d,a,K,C,D,
T019	1	orange	4	1	E,F,G,H,I,J,K,L,F,
T019	1	orange	5	12	G,H,I,J,K,L,M,
T019	1	orange	6	1	N,

Table 1-2-T1 Patients with HPV16 cancer.

Sample ID is referred to the sample number of the different patient. For example, patient P019 has T019 sample. Each sample ID is presented by its own color and id. Moreover, each sample is made up by multiple contigs. The variable repeat_time show how many time the regid_string (contig) has been iterated. The variable contig_NO is

used to represent the ordering of the regid_string (contig) which means how the path flow. The variable regid_string show the formation, sequence and direction of the patient genomic segments. From the regid_string in Table 1-5-T1, there are uppercase and lowercase genomic segments; lowercase is the viral segments; uppercase is the host genomic segments.

The genomic segment can be inserted in two direction - direct and inverted. Direct means genomic segment in the original orientation, inverted means the genomic segment in the reversed orientation which defines as the genomic segment comes in the reversed direction. The inverted orientation is represented by alphabet "r" with the meaning reverse. For example, if the genomic segment "A" is inserted in reverse order then it will get the "r_A"; else it is inserted in normal order then "A" is using. The segment of genomic is reversed due to the breakage and rearrangement of the genomic – chromosomal inversion.

In order to transform the information into a circos-type visualization, one major aspect is needed to take care – resolution. Therefore, the genomic segments (alphabets) are converted to a series of digits to increase the clarity of the circos-type visualization, decrease the size of the circos-type visualization and indirectly increase the readability. How does it work? So, giving an example "B,C,D,E,r_b,G,H", the alphabets "B,C,D,E" are arranged in a sequence, hence they can stand for one digit; "r_b" is a reversed virus segment which replaced by the subsequent digit; "G,H" are arranged in sequence, hence they are changed to the digit next to the "r_b" digit. As a result, we can convert the genomic segments (alphabets) into a sequence of digits.

#SampleID	contig_NO	repeat_time	regid_string	numbering
T003	1	1	A,	**1**[A],
T003	2	9	B,C,D,E,r_b,G,H,	**2**[B,C,D,E], **3**[r_b], **4**[G,H],
T003	3	3	B,E,r_b,G,H,	**5**[B], **6**[E], **7**[r_b], **8**[G,H],
T003	4	1	B,E,r_b,G,D,E,r_b,G,H,I,	**9**[B], **10**[E], **11**[r_b], **12**[G], **13**[D,E], **14**[r_b], **15**[G,H,I] ,

Table 1-2-T2 Numbering of sample T003 (1)

Through the numbering case, we know that multiple genomic segments (alphabets) can be represented by a digit. In the other words, the numbering depends on the sequences of the genomic segments (alphabets). This is because if each genomic segment stands for a digit (A=1, B=2, C=3, …) as shown in the Table 1-5-T3, the circos-type visualization will be messed up due to increasing of the round-shape in circos-type visualization and indirectly the clarity will be decreased.

#SampleID	contig_NO	repeat_time	regid_string	numbering
T003	1	1	A,	**1**[A],
T003	2	9	B,C,D,E,r_b,G,H,	**2**[B], **3**[C], **4**[D], **5**[E], **6**[r_b], **7**[G], **8**[H],
T003	3	3	B,E,r_b,G,H,	**9**[B], **10**[E], **11**[r_b], **12**[G], **13**[H],
T003	4	1	B,E,r_b,G,D,E,r_b,G,H,I,	**14**[B], **15**[E], **16**[r_b], **17**[G], **18**[D], **19**[E], **20**[r_b], **21**[G], **22**[H], **23**[I]

Table 1-2-T3 Numbering of sample T003 (2)

There are total 23 numbers in Table 1-5-T3 and total 15 numbers in Table 1-5-T2. Hence to rise the clarity of the circos-type visualization, numbering in Table 1-5-T2 is the best choice.

If the genomic segments are in a sequence, then they are joining together. So, the ending position of an object is the beginning position of the next object. Given the example of X with starting position 1 and ending position 5; Y with starting position of 6 and ending position 12.

Figure 1-2-F1 Sequence problem

Both directions are in sequence. So “r_Y, r_X” can be represented as one single digit. Given the regid_string “G,r_b,r_a,r_d,C,D,E,F” from example of sample T005 in Table 1-5-T1, the “r_b,r_a” is considered as one single digit since they are in sequence.

#SampleID	contig_NO	repeat_time	regid_string	numbering
T005	1	1	A,F,	**1**[A], **2**[F],
T005	2	42	G,r_b,r_a,r_d,C,D,E,F,	**3**[G], **4**[r_b,r_a], **5**[r_d], **6**[C,D,E,F],
T005	3	4	r_a,r_d,C,D,E,F,	**7**[r_a], **8**[r_d], **9**[C,D,E,F],
T005	4	1	G,H,	**10**[G,H],

Table 1-2-T4 Numbering of sample T005

From the information of sample T005 shows in Table 1-5-T5, we know that the regid_string "A,F" is followed by "G,r_b,r_a,r_d,C,D,E,F" which is repeated 42 times. Then followed by "r_a,r_d,C,D,E,F" which is iterated 4 times and lastly "G,H".

#SampleID	Seg_type	Seg_ID	Ref_seg	left_pos	right_pos	size	res	copy_number
T003	host	A	chr4	1733000	1739090	6091	80	1.03
T003	host	B	chr4	1739090	1744503	5414	70	12.65
T003	host	C	chr4	1744503	1750435	5933	100	8.94
T003	host	D	chr4	1750435	1793573	43139	500	10.39
T003	host	E	chr4	1793573	1798238	4666	60	12.32
T003	host	F	chr4	1798238	1798257	20	1	0.71
T003	host	G	chr4	1798257	1804955	6699	100	13.68
T003	host	H	chr4	1804955	1808762	3808	100	12.97
T003	host	I	chr4	1808762	1813000	4239	100	0.97
T003	virus	a	HPV16	1	7128	7128	100	N/A
T003	virus	b	HPV16	7128	7743	616	100	N/A
T003	virus	c	HPV16	7743	7905	163	100	N/A
T005	host	A	chr16	73550000	73553794	3795	40	2.45
T005	host	B	chr16	73553794	74035944	482151	8000	0.82
T005	host	C	chr16	74035944	74040276	4333	100	105.36
T005	host	D	chr16	74040276	74044288	4013	100	92.73
T005	host	E	chr16	74044288	74081722	37435	600	93
T005	host	F	chr16	74081722	74095258	13537	100	96.18
T005	host	G	chr16	74095258	74098694	3437	30	87.91
T005	host	H	chr16	74098694	74110000	11307	200	4.36
T005	virus	a	HPV16	1	2131	2131	100	N/A
T005	virus	b	HPV16	2131	2345	215	100	N/A
T005	virus	c	HPV16	2345	3547	1203	100	N/A
T005	virus	d	HPV16	3547	7902	4356	100	N/A
T019	host	A	chr2	157130000	157136911	6912	250	3.05
T019	host	B	chr2	157136911	157137101	191	3	35.89
T019	host	C	chr2	157137101	157137102	2	1	21.16
T019	host	D	chr2	157137102	157137327	226	4	52.21
T019	host	E	chr2	157137327	157152104	14778	250	2.42
T019	host	F	chr2	157152104	157169152	17049	200	4
T019	host	G	chr2	157169152	157174830	5679	100	27.05
T019	host	H	chr2	157174830	157181909	7080	100	166.42
T019	host	I	chr2	157181909	157188503	6595	100	161.16

T019	host	J	chr2	157188503	157188504	2	1	70.32
T019	host	K	chr2	157188504	157198763	10260	150	168.95
T019	host	L	chr2	157198763	157211176	12414	250	26.32
T019	host	M	chr2	157211176	157215752	4577	100	24.11
T019	host	N	chr2	157215752	157220000	4249	70	2.95
T019	virus	a	HPV16	1	1255	1255	100	N/A
T019	virus	b	HPV16	1255	2523	1269	100	N/A
T019	virus	c	HPV16	2523	3622	1100	100	N/A
T019	virus	d	HPV16	3622	7904	4283	100	N/A

Table 1-2-T5 Detail of genomic segments of each sample.

Seg_type is the type of the segment whether it is host genomic segment or viral segment. Seg_ID is the alphabet that represented the genomic segment. Ref_seg is the detail about the genomic segment and virus segment. For example, c of sample T003 is the HPV16 virus segment and A of sample T003 is the human chromosome 4 segment.

In addition, left_pos is the starting position of the particular genomic segment or virus segment. Whereas, right_pos is the ending position of the particular genomic segment or virus segment. And size is the range of the genomic segment or virus segment. The sample T019 is taken as an example to draw a part of the circos-type visualization and explain the relationship of the genomic segments.

segment_no	genomic_segments	begin	end	repeat
1	A,B,C,D	157130000	157137327	1
2	r_C,r_B	157137102	157136911	67
3	H,I	157174830	157188503	67
4	c,d	2523	7904	67
5	a	1	1255	67
6	K	157188504	157198763	67
7	C,D	157137101	157137327	67
8	r_C,r_B	157137102	157136911	4
9	H	157174830	157181909	4
10	d	3622	7904	4
11	a	1	1255	4
12	K	157188504	157198763	4
13	C,D	157137101	157137327	4
14	E,F,G,H,I,J,K,L	157137327	157211176	1
15	F	157152104	157169152	1
16	G,H,I,J,K,L,M	157169152	157215752	12
17	N	157215752	157220000	1

Table 1-2-T6 Sample T019 genomic segments

Table 1-5-T6 shows the combined information of sample T019 from Table 1-5-T5 and Table 1-5-T1. The highlighted segment_strings are the segment_string that looped more than 1 time. The segment_no 2 to 7 are repeated 67 times; segment_no 8 to 13 are iterated 4 times; lastly segment_no 16 is repeated 12 times. This information is shown in the circos-type visualization as the guideline.

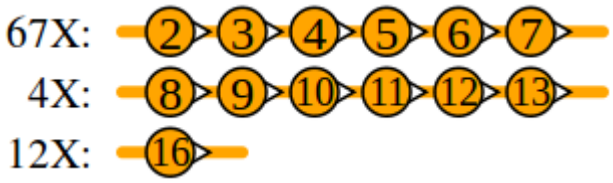


Figure 1-2-F2 Repeat of the genomic segments

There are some genomic_segments are same, example segment_no 5 and 11 have same genomic_segments “a”. Hence, they are replaced by a sign “+” to reduce the number of round-shape in the circos-type visualization in order to make the circos-type visualization more easily read. In the same way, segment_no 6 and 12 are substituted as “*” sign and segment_no 7 and 13 are substituted as “#”.

Figure 1-2-F3 Replacing of the same genomic segments

Segment_no 1 is the genomic_segments “A,B,C,D”, the begin and end value in Table 1-5-T6 are the starting and ending position of the segment_no 1. It aids in the drawing of the round-shape in circos-type visualization.

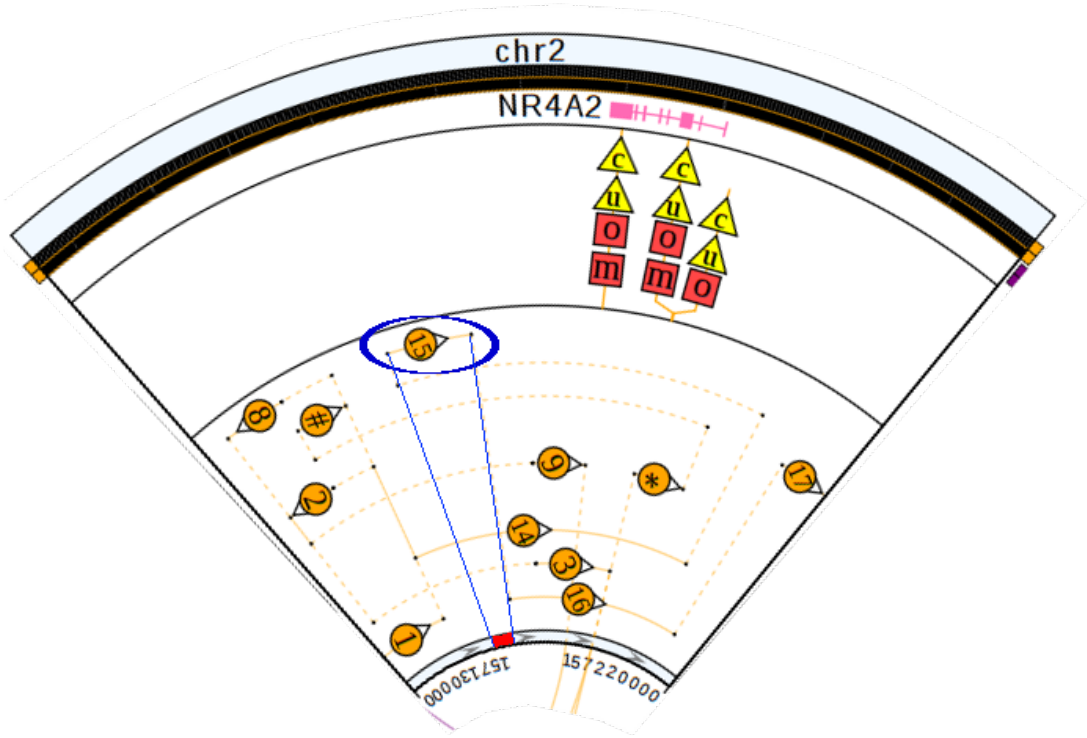


Figure 1-2-F4 Part of circos-type visualization (chr2)

From the Figure 1-2-T4, there are some small black dots. Take segment_no 15 as an example, there are two black dots at the track of segment_no 15. These black dots stand for the starting and ending position of the segment_no 15. Segment_no 15 is segment "F" with the beginning point of 157152104 and ending point of 157169152. The solid line and dashed line are used as track path but they have different usage. The solid line used in between two black dots when there is a round-shape segment_no while the dashed line with no round shape segment_no in between. Dashed line is used to aid the user easy to understand. The arrow beside the round-shape point to the direction where to go.

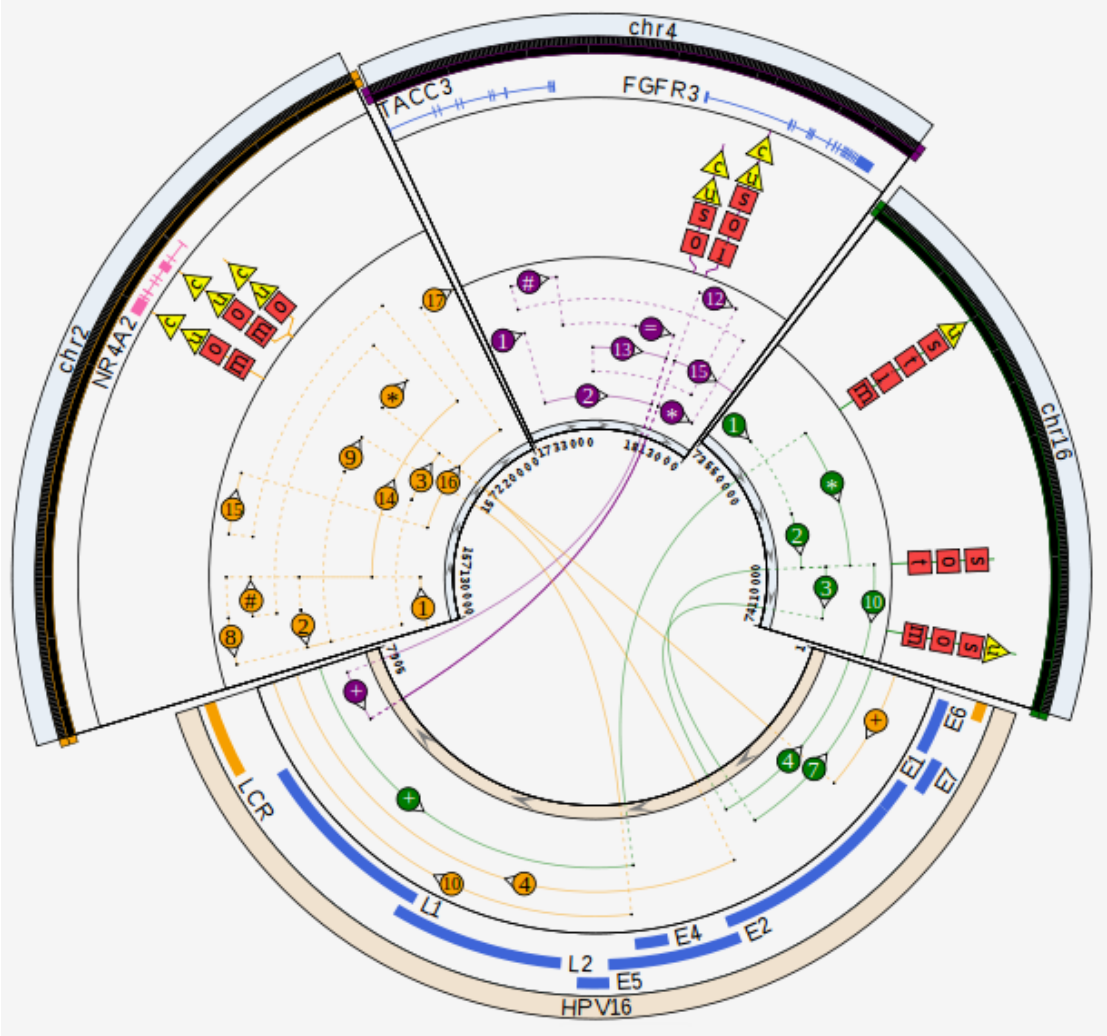


Figure 1-2-F5 Circos-type visualization of all samples.

Thus, the information of all the samples of HPV16 is drawn in the circos-type visualization as shown in Figure 1-2-T5. There are three chromosome pie sections and one HPV16 virus pie section. Total of 4 segments in a pie. From the circos, we know that there are 3 patients (3 different colors) with the HPV16 infection. The interaction of their own genomic segments and the HPV16 virus are established in this circos-type visualization. The arrangement of genomic segments in each pie section shown in Figure 1-2-F5 are not optimized. So, this is the reason why we carry out this project to increase the clarity of the circos.

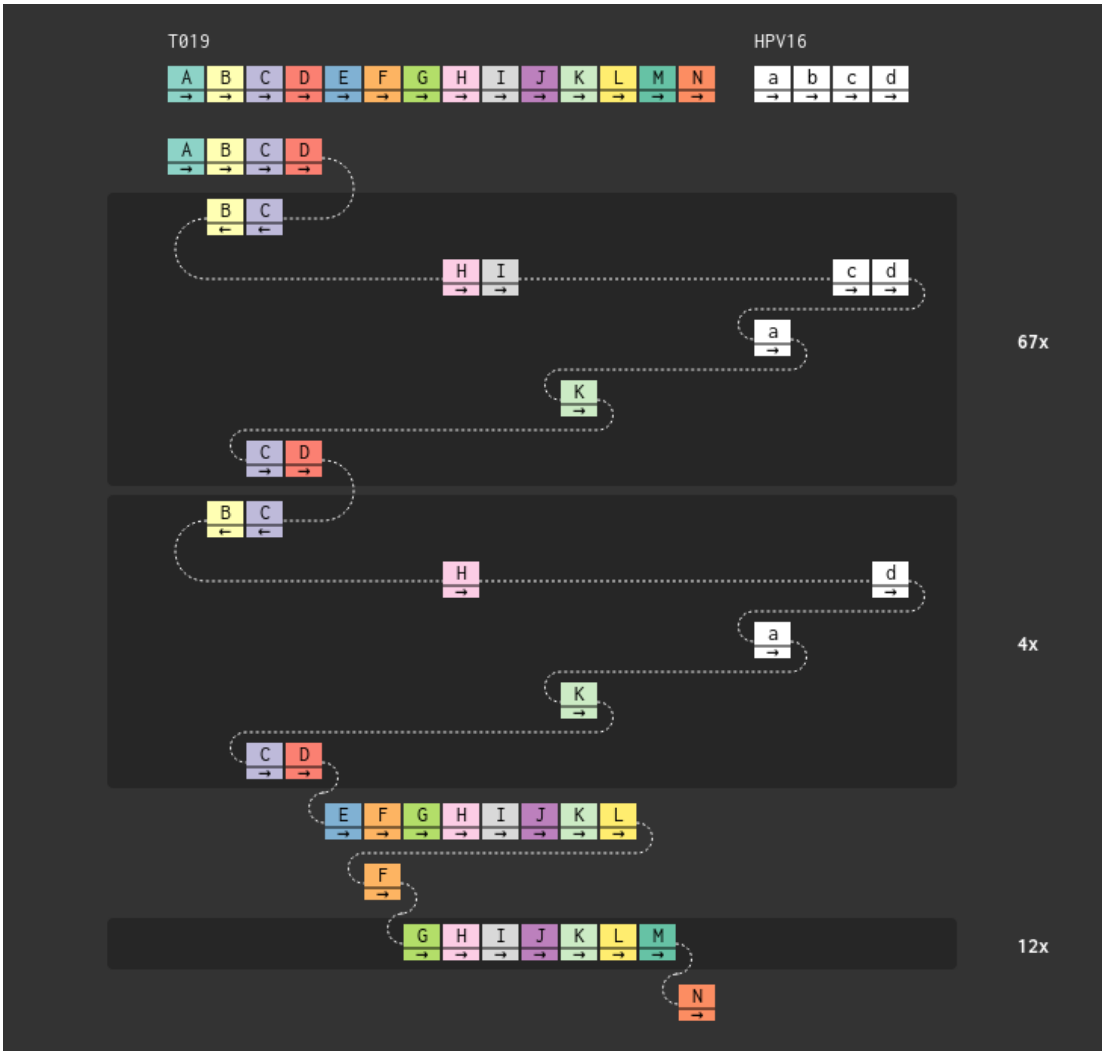


Figure 1-2-F6 Local genomic map of sample T019

Under each block of the genomic segment, there is an arrow. The forward arrow means that the direction is normal, but backward arrow means that the direction is inverted. So we can know that the flow of all the genomic segments. Firstly, “A,B,C,D” then it is inverse to “C,B” which can be represented by “r_C,r_B”. After that, “H,I,c,d,a,K,C,D”. The genomic segments “r_C,r_B, H,I,c,d,a,K,C,D” are repeated 67 times. Then follow by “r_C,r_B,H,d,a,K,C,D” with 4 times, “E,F,G,H,I,J,K,L,F”, “G,H,I,J,K,L,M” with 12 times and finally “N”.

1-3 Project Scope

This project focuses on the arranging genomic segments in circular form. The new algorithm will be designed to optimize the arrangement of the genomic segments. Moreover, the data processing on the genomic segments will be carried out. Besides,

the final circos-type visualization will have the interaction such as dragging and double clicking, user can interact with the circos-type visualization.

There are some variables prompted from user, so it is used as input to develop an algorithm for arranging the cross-intersecting genomic segments for circos-type visualization. The final algorithm will be tested with the practical genomic data in order to try out its effectiveness. The algorithm will give the best method to allocate all the genomic segments. The useful and interactive circos-type visualization is provided, in order to give the better explain and display the cross-intersecting genomic segments.

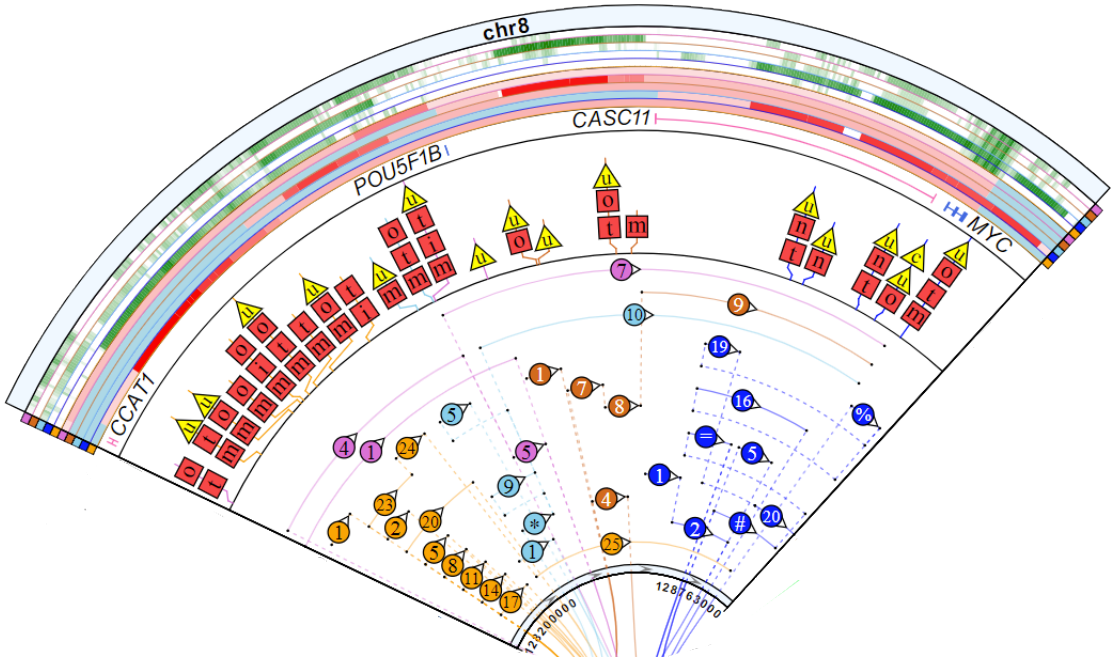


Figure 1-3-F1 Circos-type visualization of chr8.

As the figure shown, it is one of the pie regions of the circos-type visualization: human chromosome 8 of multiple patients (colors).

A *segment* is an interval (a, b), where $a \in N$ indicates the start position of the segment and $b \in N$ indicates the end position of the segment. So, $b > a$. N is the natural numbers – all positive integers starting from 1.

A track is a line $(0, MAX)$, where MAX is the largest value for segment positions. There are *n* tracks in a problem instance, numbered from 1 to *n*. The tracks are parallel to each

other, with a distance of D between each track i and its subsequent track $i + 1$. The total height of all tracks area is $(n - 1)D$, denoted as L. In order for the tracks to fit within the size of the paper, it is required that do not exceed some value.

The segment interval (a, b) has a length, x. Same interval has different length in different track. The higher the track, the larger the length of the line for the same interval. Figure 1-2-F3 shows that the same interval (a, b) in different tracks. There are 6 line of segment, $x_1 < x_2 < x_3 < x_4 < x_5 < x_6$.

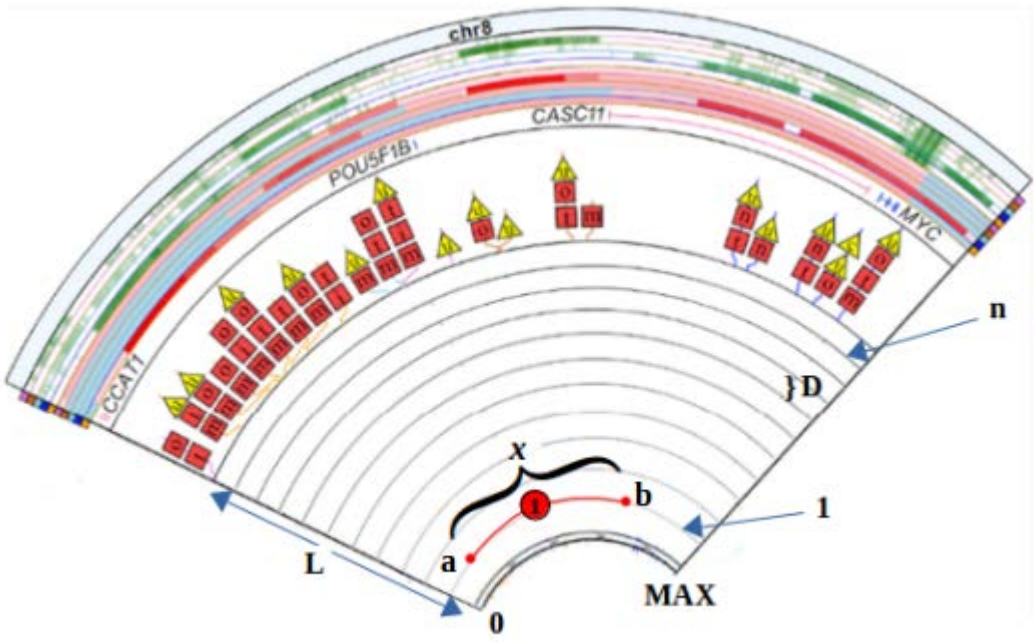


Figure 1-3-F2 Example of circos-type visualization of chr8.

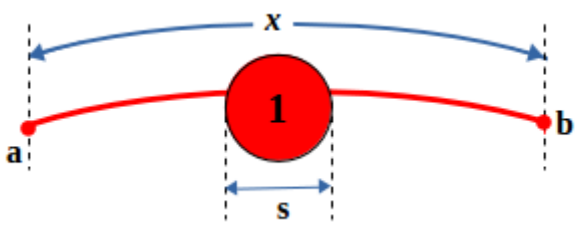


Figure 1-3-F3 Segment '1' with interval (a, b)(1).

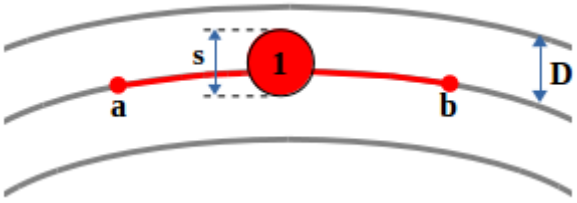


Figure 1-3-F4 Segment '1' with interval (a, b)(2).

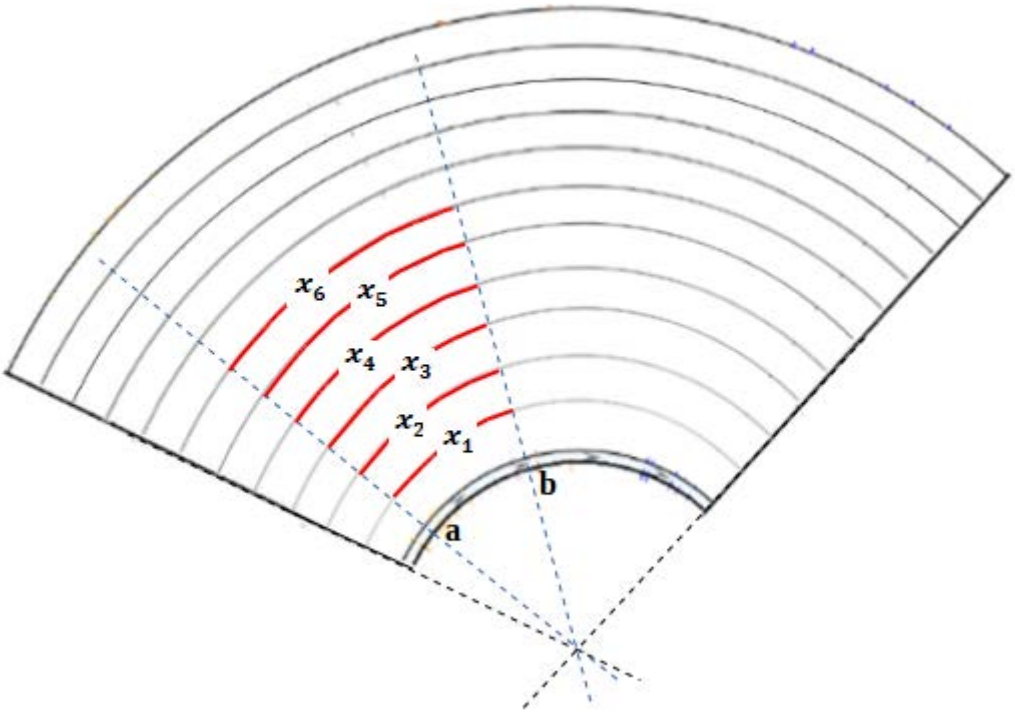


Figure 1-3-F5 Multiple segments with interval (a, b).

A *placement of segments* is a mapping $f: N \rightarrow N$ which maps each segment to a value from 1 to *n*. Namely, each segment is assigned to a track and multiple segments can assign in a track. If track $f(A) = f(B)$ for segments $A = (a_1, a_2)$ and $B = (b_1, b_2)$, then either $a_2 < b_1$ or $b_2 < a_1$.

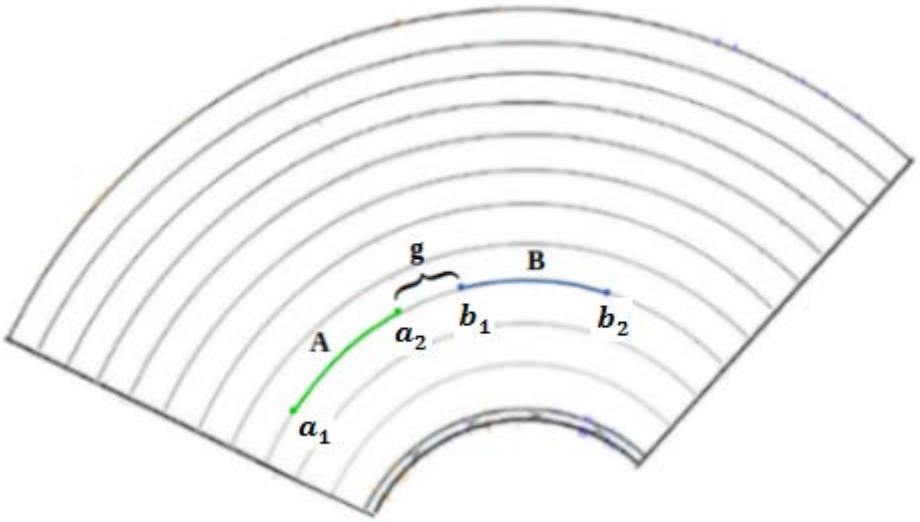


Figure 1-3-F6 Two segments in same track (1).

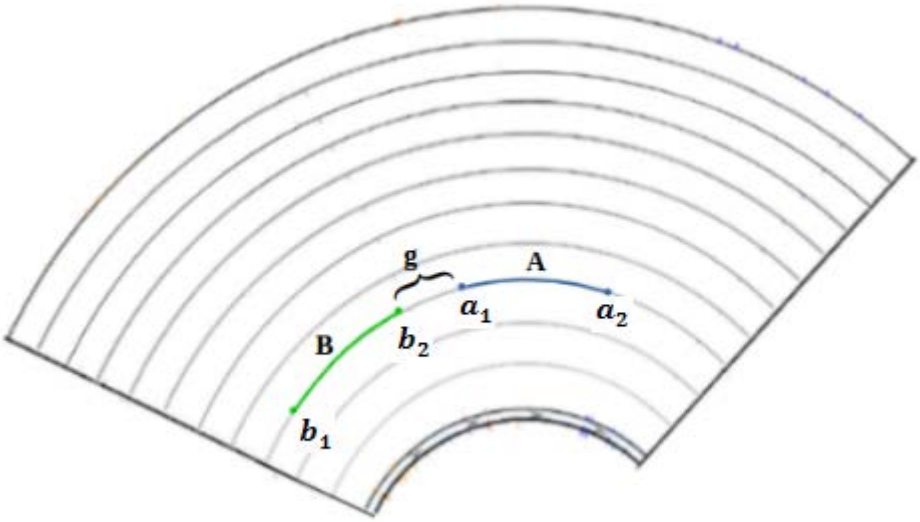


Figure 1-3-F7 Two segments in same track (2).

Given segments $A = (a_1, a_2)$ and $B = (b_1, b_2)$ on track *x* and *y*, we define the distance between *A* and *B*, $dist(A, B)$, as the distance between the center of *A* and the center of *B*. C_A is the center of the segment $A = (a_1, a_2)$ and C_B is the center of the segment $B = (b_1, b_2)$. So, $C_A = \frac{a_1+a_2}{2}$ and $C_B = \frac{b_1+b_2}{2}$.

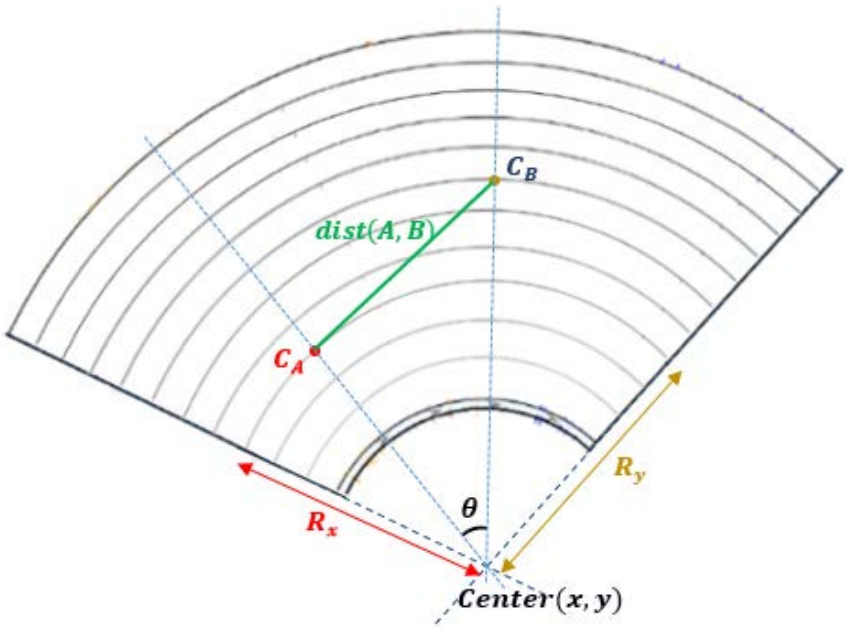


Figure 1-3-F8 Distance between the center of 2 segments (1).

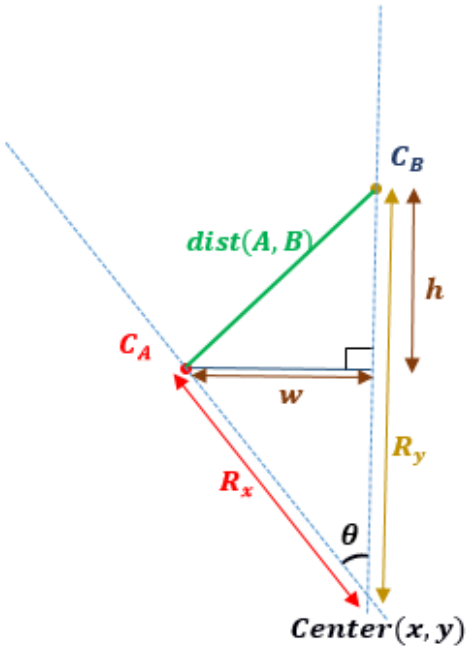


Figure 1-3-F9 Distance between the center of 2 segments (2).

To find the distance, the radius of the track that had been placed by the segments are found out. After that, the distance is calculated as shown in below:

$$w = R_x \sin\theta$$

$$h = R_y - R_x \cos\theta$$

$$dist(A,B) = \sqrt{w^2 + \mathrm{h}^2}$$

Problem A: Find a placement of segments such that $min_{(A,B)}dist(A,B)$ is larger than a given threshold *T*.

- If the problem cannot be fulfilled, the method should return no.
- However, this definition is weak against the situation above where 5, 8, 11, 14, 17 must either be placed on alternating tracks (with one blank track in between every two of them), or it becomes inevitable that.

Problem B: Find a placement of segments which maximizes

$$|\{min_B dist(A,B) > T\}||\{min_A dist(A,B) > T\}|$$

Additional requirements for advanced versions:

- Some segments *A* and *B* are to be placed near each other (on top or below). In this case, either $A \subseteq B$ or $B \subseteq A$. (These paired segments are to handle cases such as 15→16 and 18→19 in the example above which 15 and 18 are the symbol "%")
- Allow constant increment for tracks that are further from the center of the circle.
- The gap, g between the two segments in same track must greater than 0.

1-4 Project Objectives

This project aims to develop circos-type visualization with cross-intersecting genomic segments. Furthermore, to find out a more optimized method to arrange the genomic segments is also one of the objectives. In addition, purpose of this project is to form an interactive circos-type visualization.

The output of this project will give the advantages to the people whom interested in the genetic area such as genetic investigator. A genetic investigator can leverage the algorithm to develop a genetically related work.

With the optimized method, the circos-type visualization will become more clear and readable. Indirectly, the users are able to read the relationship between genomic segments more clearly. The users will know where the gene had been mutated due to the virus by referring the circos-type visualization.

The interactive circos-type visualization allows the user to interact with the genomic segments to have the better display of the circos-type visualization.

This project only focuses on the exploit a method to arranging the genomic segments but not finding the real-world data. Besides that, investigating the information of the genomic segments is also not covered by this project. The information of genomic segments fill into the algorithm, and the algorithm will find the best way to arrange the genomic segments.

1-5 Impact, significance and contribution

The projects on human DNA are becoming more and more well known, especially on the disease. Therefore, visualization is very important to interpret the genomic information. The readers will get the benefit of knowing how to solve the allocation problems in the circular shape. The genetic interpreters can use this project output as a guideline to arrange the genomic segments in circos-type visualization.

The circos-type visualization is more attractive and easy to understand. By having this algorithm, the cross-intersecting genomic segments can arrange in an optimized position. The size of the circos-type visualization is prompted from user to draw the circos-type visualization.

This project will generate a circos-type visualization with the interaction. Users are able to interact with the circos-type visualization and display the better explain of the circos-type visualization. From this project, the people will visualize how the genes arrange

when affected by the virus. This helps the medical scientist to study the ways to cure the cancer. Besides, the people will learn about arranging block in circular shape.

1-6 Report Organisation

This report includes of 6 chapters in total. The first chapter introduces the project background, the motivations for working on the project and also the project objectives. In the second chapter, the past researches on the topic of circos-type visualization and arrangement segments are studied and reviewed.

Chapter 3 explains the drawing method in detail. The definitions and equations of the mathematical model are in this chapter. Moreover, chapter 4 details the methodology and tools used in the project in addition to the timeline of the project. Chapter 5 describes the results of the project.

The final chapter concludes the project and explains on possible future improvements on the finished system.

CHAPTER 2 LITERATURE REVIEW

2-1 Dynamic Storage Allocation

Dynamic Storage Allocation is a popular topic in combinatorial optimization. It is a problem of arranging a set of blocks with different size in an area. The target of offline DSA is to minimize the area usage. It is used in many regions, example memory allocation (Robson 1974) and berth allocation. In offline DSA, the inputs (blocks) is known in advance, while in online DSA, the inputs (blocks) is known when running the algorithm.

According to the study of the Knuth (1973), the offline DSA problem was investigated. The main issue of DSA is the NP-completeness. NP-completeness can be examined in polynomial time, but there is no efficient method to solve the problem (Garey & Johnson 1979). This topic has received much attention from the public. Hence, there is some research on the DSA with the approximation algorithms. From the ratio of $h \ln \theta$ by Robson (1974); then 80 by Kierstead (1988); 6 from Kierstead (1991); and 5 by Gergov(1996) the most recent 3 by Gergov (1999).

Moreover, the offline DSA is formulated as: Given a set of blocks $B = \{B_1, B_2, \dots, B_n\}$. For $1 \leq i \leq n$, $B_i(s_i, a_i, d_i)$ where s_i is the size of the block, a_i is the arrival time and d_i is the departure time. The allocation of the blocks B is $A(B) = \{A_1, A_2, \dots, A_n\}$ is the position/non-negative value that assign to each block B_i. Given that $i < j$, $[a_i, d_i) \cap [a_j, d_j) = \emptyset$ and $(A_i, A_i + s_i) \cap (A_j, A_j + s_j) = \emptyset$.

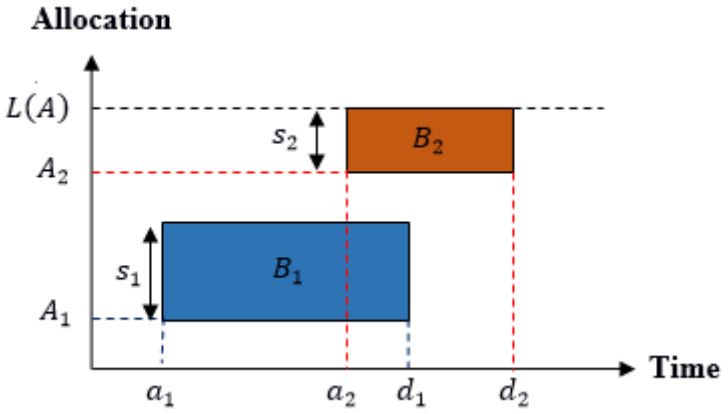


Figure 2-1-F1 DSA of two blocks.

DSA can be interpreted in x-axis and y-axis where x-axis stands for the time and y-axis stand for the area of storage. The x-position of each block is fixed, so the block can only move vertically. Thus, allocation position $A(B)$ is merely the y-axis point. Given a block B_i is ready for allocating, the x-extend of the B_i is the time interval $(a_i, d_i)$and the y-extend is the position to allocate $(A_i, A_i + s_i)$. The y-axis is the accumulation of the allocation $A(B)$ of all blocks.

The highest y-extend denoted as $L(A) = max_{(1 \le i \le n)}\{A_i + s_i\}$. $L(A)$ can be said as the minimum height to allocate all the blocks *B*. Li (2004) stated that there are two versions of DSA: optimization and decision. The objective of optimization DSA is to find the minimum height to allocate blocks or minimise $L(A)$ to arrange all the blocks. In addition, decision DSA is to find a method to allocate all the blocks given that the threshold Lmax is provided, $L(A) \le L_{max}$.

DSA is allocation problem on the rectangle area, but the arrangement genomic segments is the allocation problem on the circle. So, the solution of DSA is not suitable to solve the problem. Moreover, the blocks of DSA problem vary in size, on the other hand, the genomic segments are always with the size 1.

2-2 CIRCOS

Circos is a software for visualizing data and information (Martin et al. 2009). The data are displayed in a circular layout. This helps the user to explore the relationship between the information easily. Circos is written in Perl which available in any of the operating systems whose support the Perl. The final outcome of the Circos is bitmap (PNG) or vector (SVG).

At first, Circos is used to display genomic data peculiarly comparative genomics and cancer genomics. Due to the large data set of the genomic information, it is suitable to use Circos which encompass the information in one circular layout. Contemporarily, there are many other fields apply Circos to figure their data, such as visualizing the flow of the customer to the auto industry. Circos is automated system. The creation of the graphics is controlled by plain-text configuration file. There is no graphical user interface (GUI), whereas Circos is controlled by command line.

User can compose scripts to produce the data and configuration file. After write scripts, user merely needs to call the function of Circos to generate the image. Furthermore, user can create different output image with same configuration file and data. Images can be restructured by using the dynamic formatting in run time rules which change the color of the output image. For example, highlight the bin with value exceed 12 with blue color.

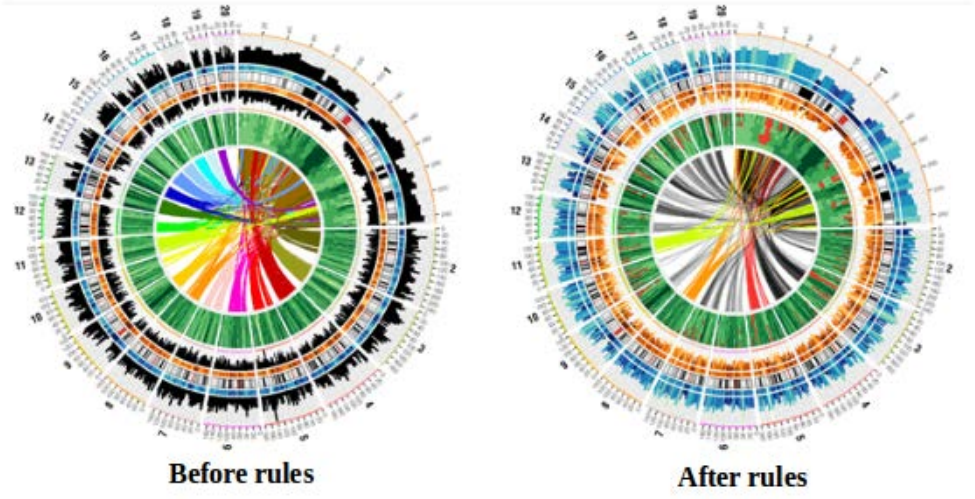


Figure 2-2-F1 Rules added to the Circos image. (Circos 2009)

Procedures to create the image in Circos are shown as below:

- deciding how the data is to be presented.
- changing the format of data file into Circos format.
- creating the configuration file.
- running Circos to generate PNG and SVG files.
- editing PNG and SVG files by adding the text labels.

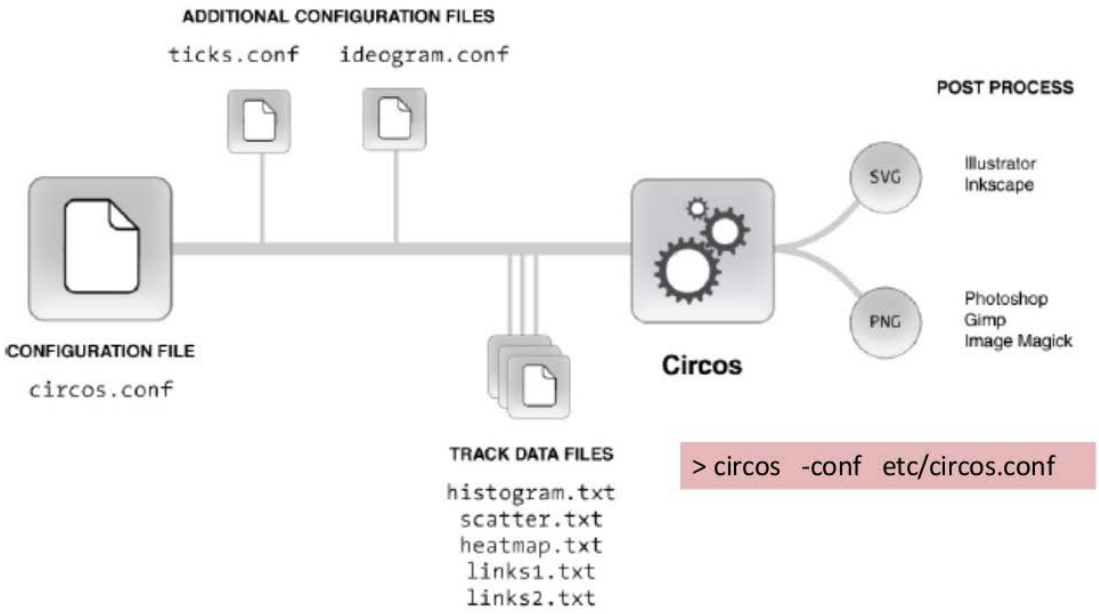


Figure 2-2-F2 Procedures to create Circos image. (Zhou 2014)

Circos include varies of 2D data tracks such as histograms, heat maps, scatter plots, tiles, line plots, connectors, texts and glyphs. It also contains ideograms, links, ticks and labels. Figure 2-2-F3 shows the simple image form by Circos with only little features. Symbol ‘A’ is the ideogram of chromosome; ‘B’ is the ticks; ‘C’ is the labels; and ‘D’ is the links.

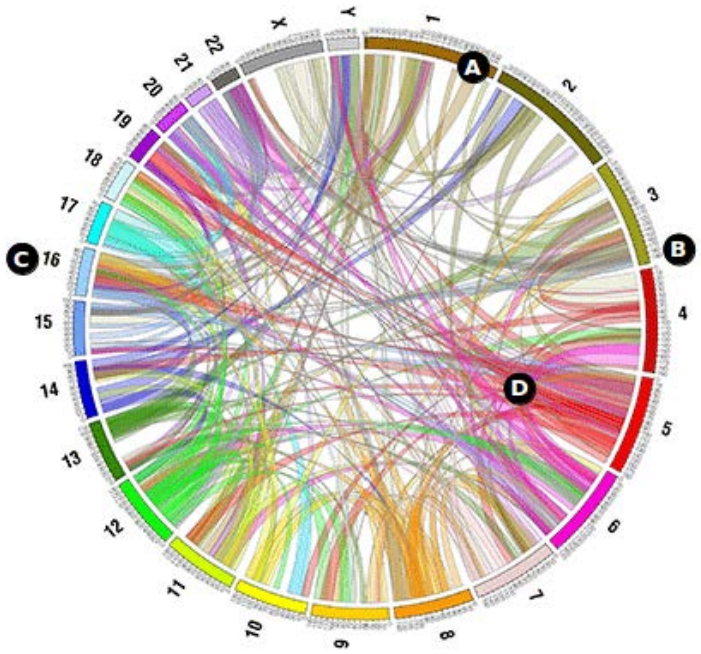


Figure 2-2-F3 Simple Circos image of human chromosome. (Circos 2009)

Figure 2-2-F4 shows the complex image form by Circos. 'A' is the scatter plot track; 'B' is the histograms; 'C' and 'D' are the heat maps; 'E' is the link bundles; 'F' is the ticks; 'G' is the scatter plot which correspond to the 'C'.

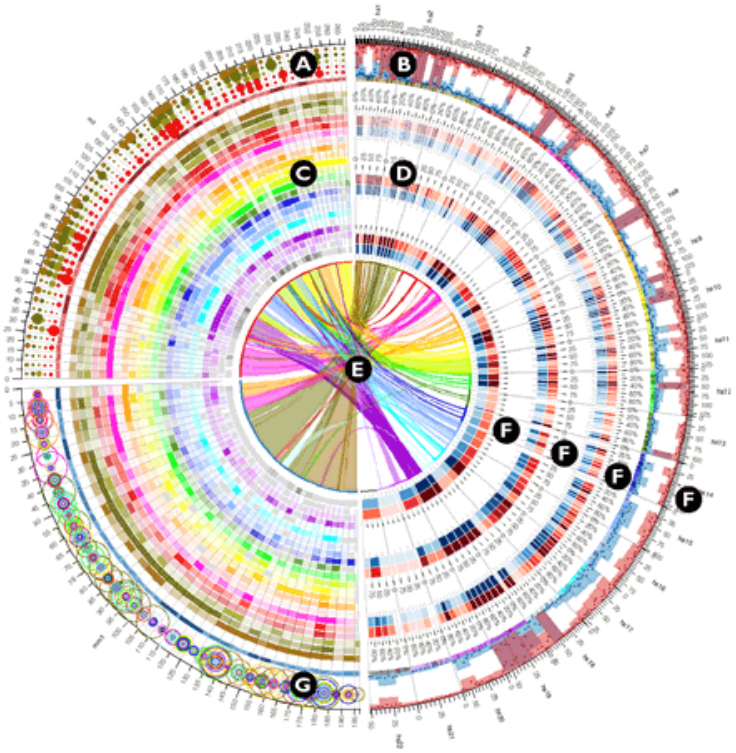


Figure 2-2-F4 Different types of 2D data tracks. (Circos 2009)

Circos is the most powerful software to visualize the data. Circos can produce an attractive image due to the multiple colors but the user may be dazzled when see the image. The multiple lines cause the relationship of the data hard to determine. User require to create more images with different rules added such as image which certain portion of the data is enlarged.

2-3 RCircos

RCircos is the R packages for the circos-type plots of genomic data (Zhang, Meltzer & Davis 2013). It can be integrate directly with R software for data processing and graphic management pipelines for generating the genomic data. Therefore, the steps of data processing to generate data files and configure files from database are not require in RCircos.

There are many types of Circos 2D data tracks plots include in RCircos. For instances, heat maps, histograms, ideograms with cytoband, lines, scatter plots and tiles. Plot items for decorations include text labels, links and connectors.

RCircos follow the layout example introduced by Circos. In addition, it also arrange data plots by tracks. The dedicated function is called to draw each type of Circos 2D data tracks plot.

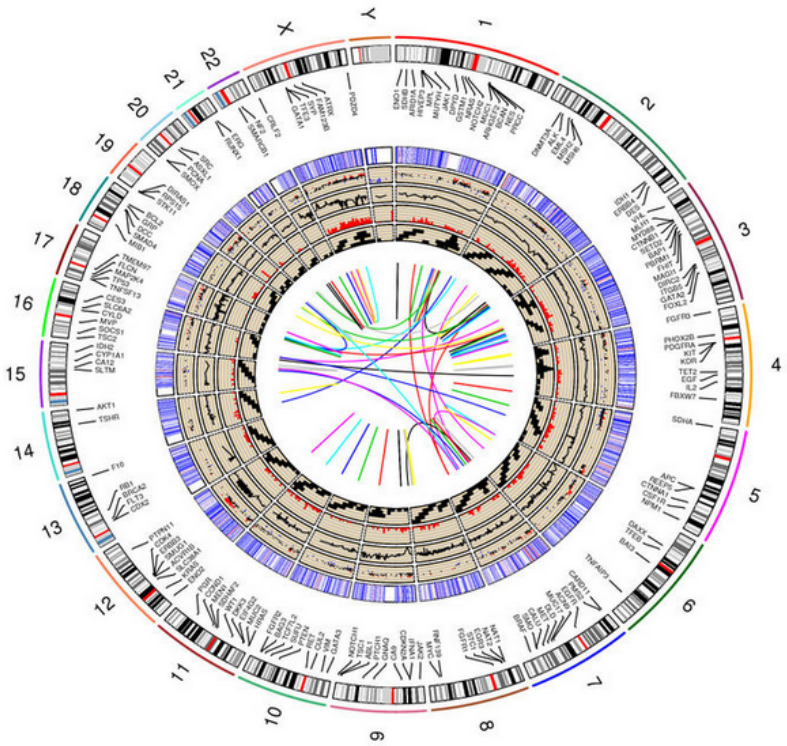


Figure 2-3-F1 RCircos image of human chromosome. (Zhang , Meltzer & Davis 2013)

Figure 2-3-F1 shows the RCircos image that contains ideogram with data tracks for connectors, labels, scatter plots, line plots, histogram, heat maps, tiles and links.

Moreover, RCircos currently support the human, mouse and rat genomic data to produce image. This is due to RCircos produce the ideogram images from the chromosome ideogram tables of UCSC genome browser.

RCircos is the tools for visualizing the genomic data, but it is not powerful enough. RCircos lacks of the functions of filter, format, adjust the data of the image. User cannot enlarge the image and has to run the functions required to generate the input genomic data.

CHAPTER 3 SYSTEM DESIGN

3-1 System Setup

The research on the allocation problem is found out. The combination of the idea is the used to develop the algorithm. The testing of the algorithm is carried out. Before the testing part, the system is set up. The following steps are carried out:

1. Ubuntu operating system is installed on the laptop.
2. Git is downloaded as the commands shown below:

```
$ sudo apt-get update
$ sudo apt-get install git
```

3. Tabix is installed as the commands shown below:

```
$ sudo apt-get update
$ sudo apt-get install tabix
```

4. Python2.7/pip is installed by various steps:
   - Install dependencies.

```
$ sudo apt-get install build-essential checkinstall
$ sudo apt-get install libreadline-gplv2-dev libncursesw5-devlibssl-devlibsqlite3-dev tk-dev libgdbm-dev libc6-dev libbz2-dev
```

   - Download Python2.7.13.

```
$ version=2.7.13
$ cd ~/Downloads/
$ wget https://www.python.org/ftp/python/$version/Python-$version.tgz
```

   - Extract and go to the downloaded file directory.

```
$ tar -xvf Python-$version.tgz
$ cd Python-$version
```

   - Install Python2.7. The command "./configure" used to check whether the application is installed and show the errors if the installation is failed. The command "checkinstall" is used to ease the uninstallation if needed.

```
$ ./configure
$ make
$ sudo checkinstall
```

- Install Easy Install.

```
$ sudo apt-get install python-setuptools python-dev build-essential
```

- Install pip.

```
$ sudo pip install --upgrade pip
```

- Install virtualenv.

```
$ sudo pip install --upgrade virtualenv
```

5. Postgres SQL is installed. "-contrib" package adds some additional utilities and functionality.

```
$ sudo apt-get update
$ sudo apt-get install postgresql postgresql-contrib
```

6. Node.js and npm are installed:

   Add Node.js PPA. PPA is for non-standard software or updates.

```
$ sudo apt-get install python-software-properties
$ curl -sL  | sudo -E bash -
```

   Install latest node.js and npm. Npm installed along with node.js.

```
$ sudo apt-get install node.js
```

   Check the version of the node.js. The version of node will show.

```
$ node -v
```

Check the version of the npm.

```
$ npm -v
```

7. The project dovirus is cloned by the command below:

```
$ git clone git@git.lhc.moe:dovirus
```

8. Change the working directory to dovirus:

```
$ cd dovirus
```

9. Install the packages that are needed to run the file:

```
$ pip install -r requirements.txt
$ npm install
```

10. The project is initialized:

```
$ echo -e "BVD3_ENABLE_SAMPLES =
False\nBVD3_INDEX_PACKAGE = 'dovirus'" > bvd3/settings_bvd3.py
```

11. Set up database:

    i. Create a user named virus.

```
$ sudo -u postgres createuser virus --createdb
```

ii. Create database. “-O” means the owner.

```
$ sudo -u postgres createdb -O virus virus_dev
```

iii. Switch to the server by postgres account to change the password. The password is changed to “virus_test”.

```
$ sudo -u postgres psql
#ALTER USER virus WITH PASSWORD ‘virus_test’;
```

iv. Migrate existing py file to packages.

```
$ python manage.py migrate
```

v. Create superuser for the project

```
$ python manage.py createsuperuser
```

12. The server is run and listen to the file changes.

```
$ python manage.py runserver
$ npm run watch
```

3-2 Platform Overview

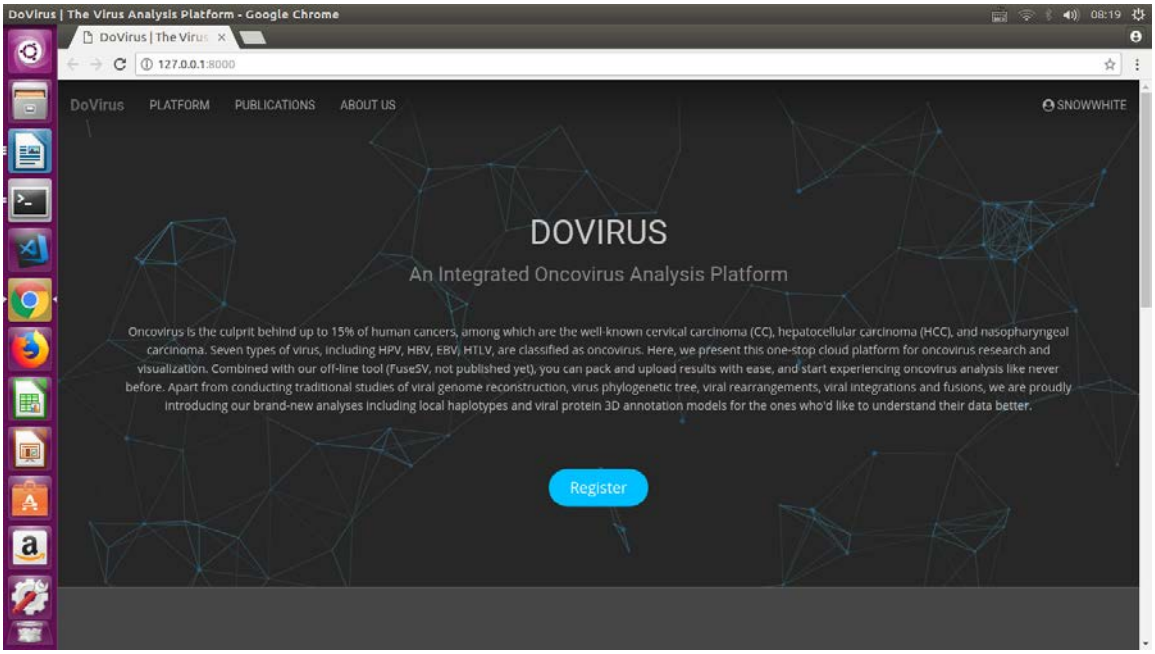


Figure 3-2-F1 Dovirus framework

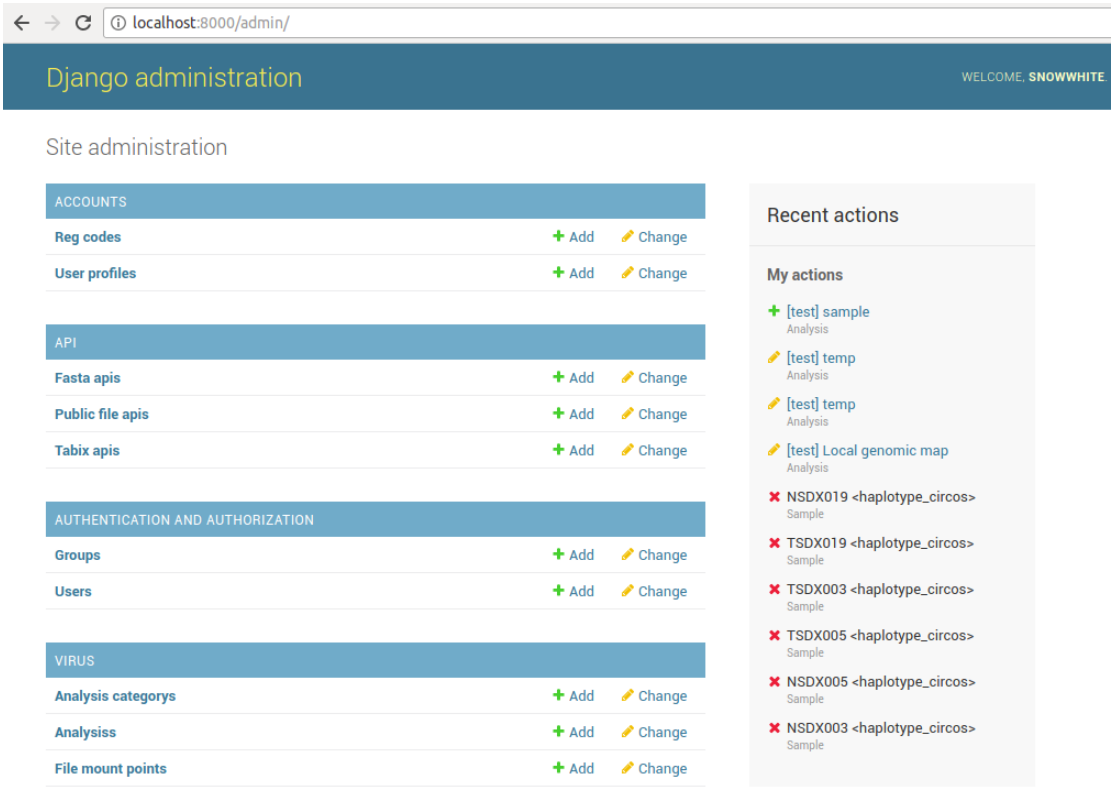


Figure 3-2-F2 Administration page

To start the visualization drawing, we have to:

Step	Action
1	Add Project
2	Add Analysis Category
3	Add Analysis

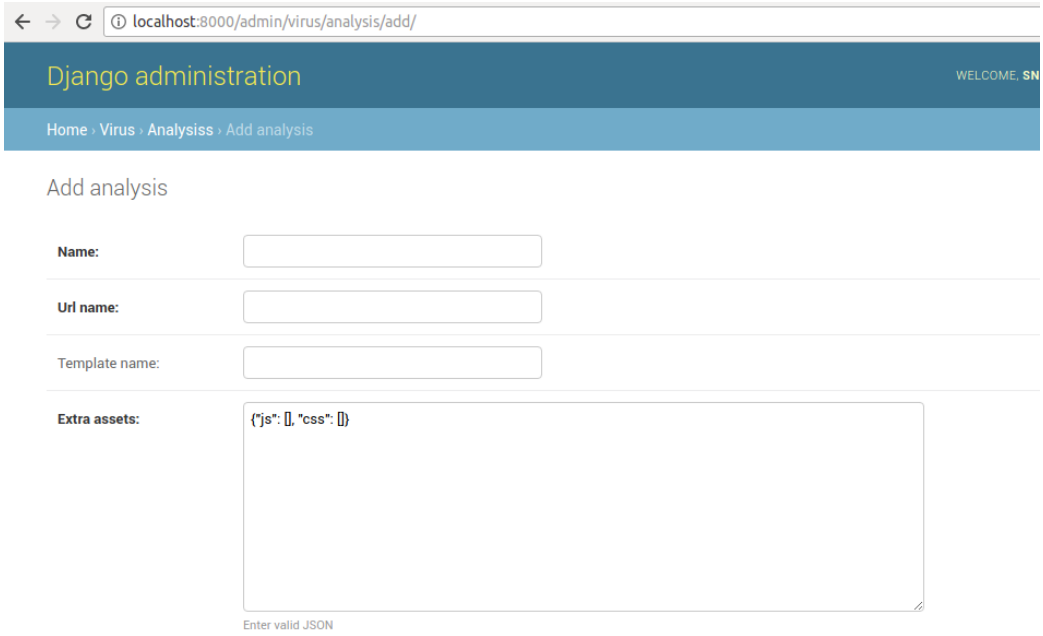


Figure 3-2-F3 Add analysis page (part 1)

When add analysis, the analysis name, url name and template name is needed to be entered. Url name is the name of the analysis that show in URL.

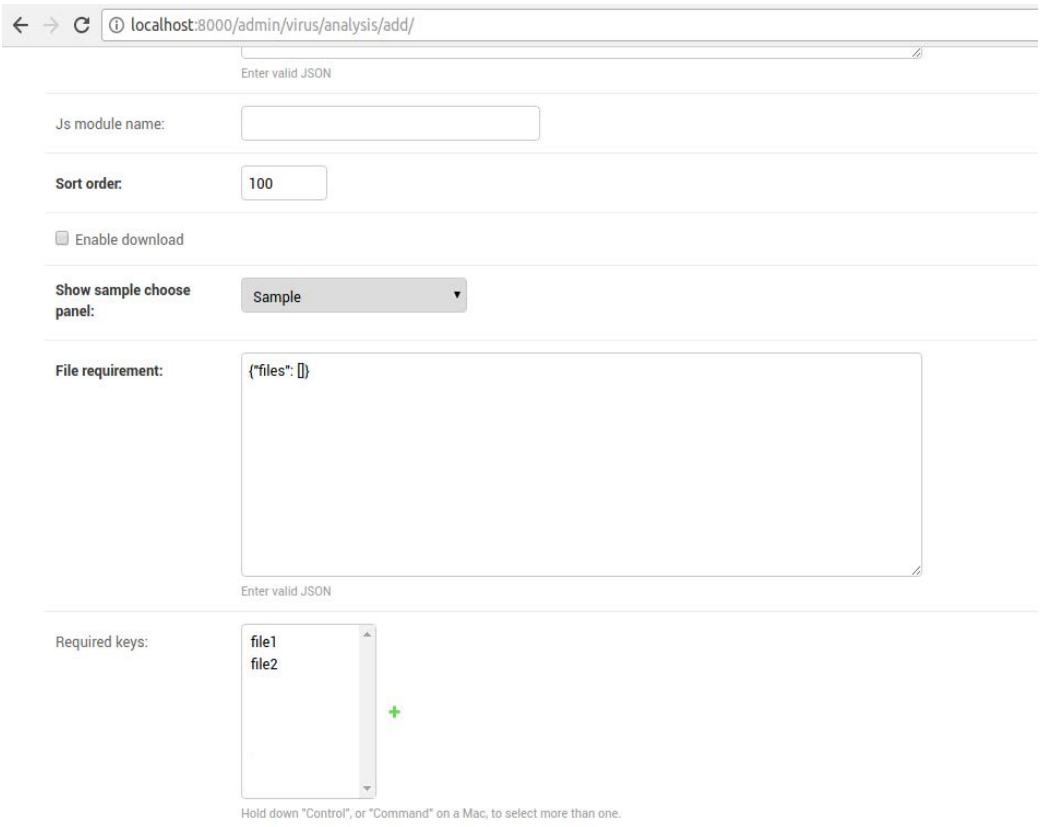


Figure 3-2-F4 Add analysis page (part 2)

Besides, we have to enter js module name which is the name of the directory to store all the file of visualization, such as .typescipt, .sass and so on. One js module represents one graph. Required key is the file key that indicate the key name of the file that need to be uploaded.

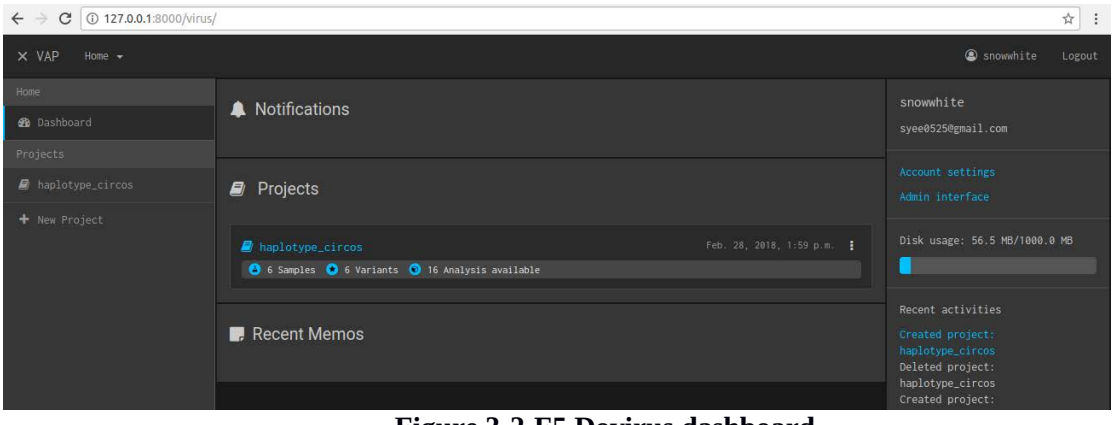


Figure 3-2-F5 Dovirus dashboard

Figure 3-2-F5 is the dashboard which will show the projects that are created by the user.

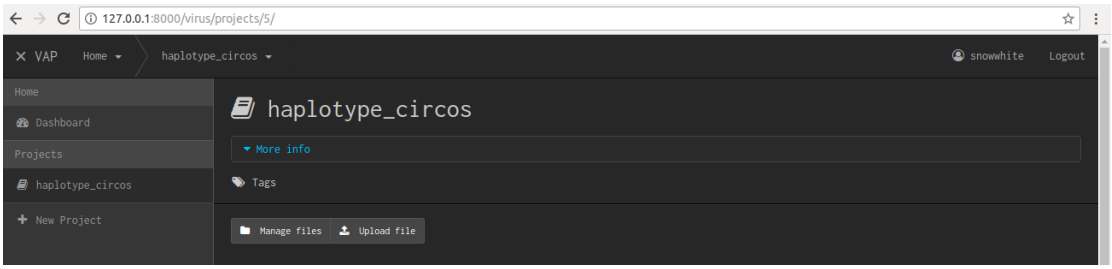


Figure 3-2-F6 Project page (part 1).

Figure 3-2-F6 shows the project named “haplotype_circos”. User can upload the file and manage the file.

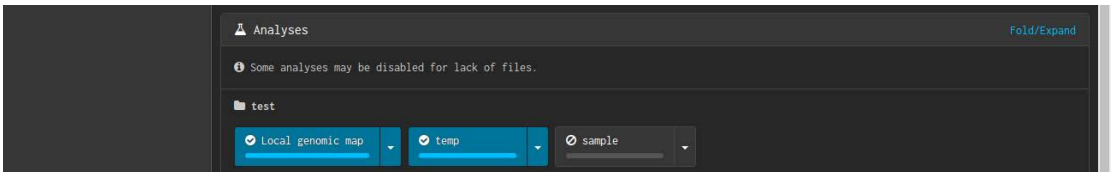


Figure 3-2-F7 Project page (part 2).

The created analysis will show in the project page.

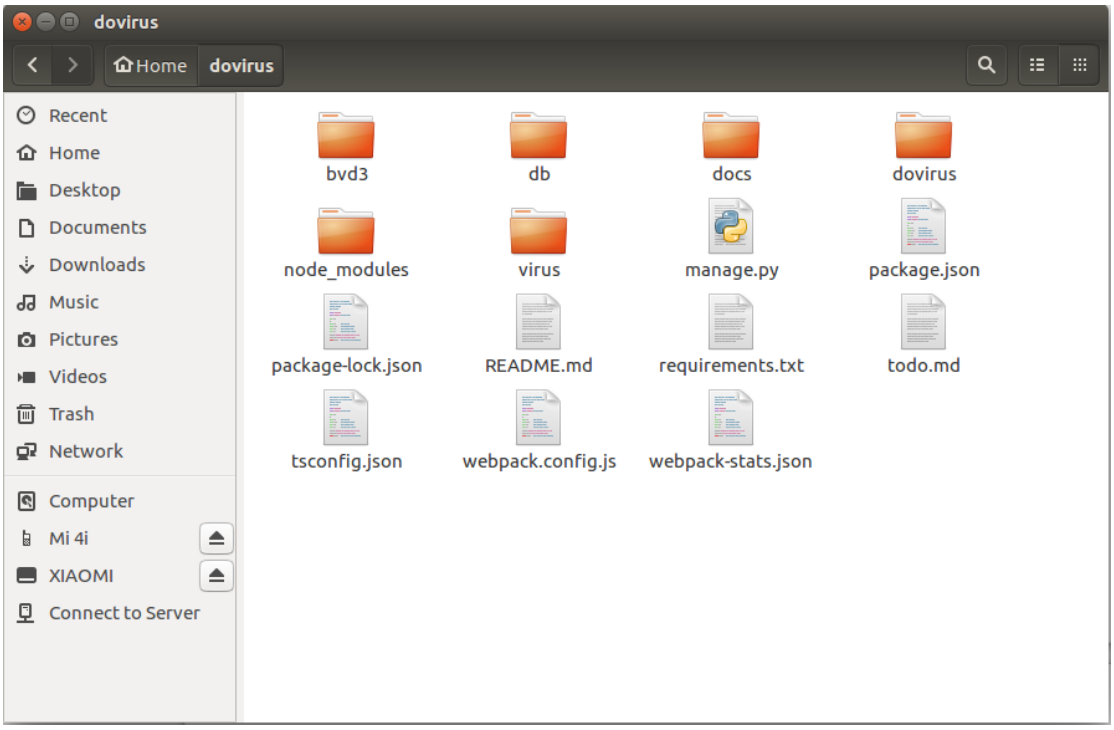


Figure 3-2-F8 Dovirus file

Figure 3-2-F8 shows the dovirus directory. This dovirus file will show after cloning the dovirus project. The folder bvd3 stored all the information needed for the platform. The

folder db stores the project created. Moreover, node_modules is the folder outcome of the installation of node.js. The folder virus stores the visualization files. The requirement.txt is the text file which shows the requirements of software after the project clone. Furthermore, manage.py checked the installation of Django.

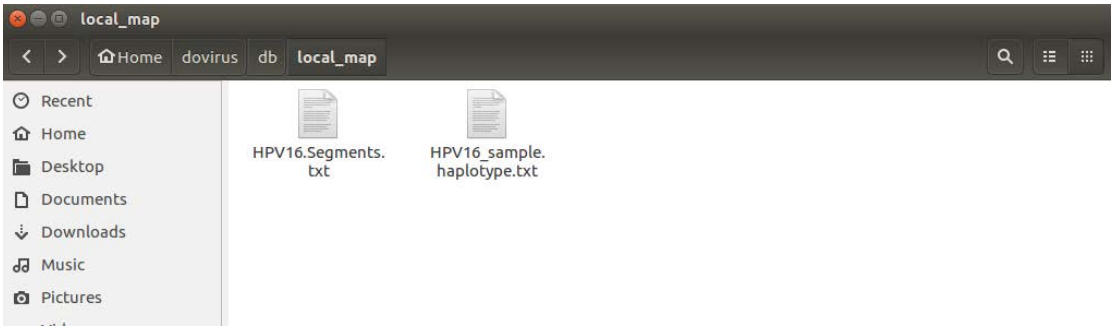


Figure 3-2-F9 Dovirus db folder contents

The db folder contains the uploaded datasets which use to generate the drawing for an analysis.

Figure 3-2-F10 Analysis folder with js module name.

The folder location of generated analysis is /dovirus/virus/static/app/[js_module_name]. There are 3 SASS and 3 TypeScript files which added to develop the graph. The visualization of the graph and data loading are done in the visualizer.ts file.

After the framework is set up, user may visualize the graph. Firstly, data loading is carried out. Then process the data to be used to draw the graph. With the processed data, the graph can be visualized. Interactions are added on to enable user to interact with the graph for further information. D3.js used to visualize the graph.

3-3 Data Loading

```
interface Data {
    haplotypes: { [id: string]: DataHaplotype[] };
    segments: { [id: string]: DataSegment[] };
```

Figure 3-3-F1 Data interface

Interface Data contains variables haplotypes and segments, which then used to load the data into it.

```
public static apiRoot: string = "/api";
public static dataTypes = ["haplotypes", "segments"];
public static apiMap = {
    haplotypes: "haplotype/HPV16",
    segments: "segments/HPV16",
};
```

Figure 3-3-F2 Data mapping

The data from the entered file are mapped to the created variables in interface Data respectively. So, the haplotypes map to the data from file “haplotype/HPV16”, and segments map to the data from file “segments/HPV16”.

```
public run() {
    this.loadData("haplotypes", (d) => {
        this.data.haplotypes = d.data;
    });

    this.loadData("segments", (d) => {
        this.data.segments = d.data;
        this.orderedSamIds = Object.keys(this.data.segments);
    });
}
```

Figure 3-3-F3 Data loading

In run() function, data are loaded to the respective variables. There are two files used to create the graph. Before drawing the graph, data pre-processing is done.

3-4 Data Pre-processing

The inputs of inner radius, outer radius, angle span, track thickness, number of tracks for each reference segments are used to generate the graph. Each arc represents one reference segment chr2/chr4/… Inner radius, $\boldsymbol{R_{inner}}$ is the radius of the arc near to the $\boldsymbol{P_0(x_0, y_0)}$, while outer radius, $\boldsymbol{R_{outer}}$ is the radius of the arc far from the $\boldsymbol{P_0(x_0, y_0)}$. Angle span, $\boldsymbol{\theta}$ is how big the arc in degree. Number of tracks, $\boldsymbol{N}$ show how many tracks in one arc. Moreover, track thickness, $\boldsymbol{h}$ indicate the segment thickness.

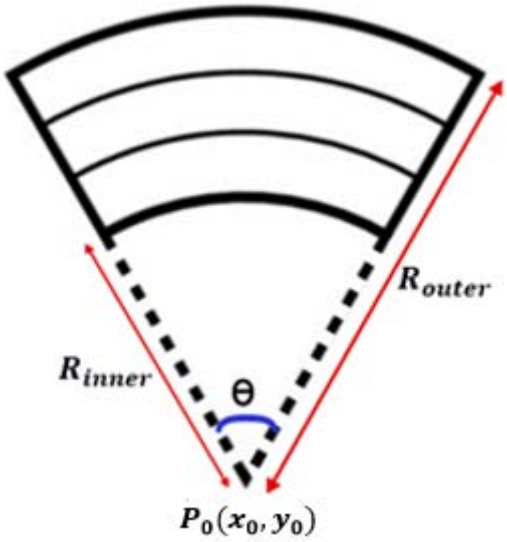


Figure 3-4-F1 Arc description.

Figure 3-4-F2 Segment description.

The start angle and end angle of the reference segments are calculated. There are many reference segments in one graph, which from host or virus. Host reference segment is come from human chromosome, example chromosome 1 to 22. Virus reference segment is come from virus, example HPV16, HPV18 and so on. Each graph is to analyse the disease that affect the human chromosome, so it is made up by only one virus reference segment and many host reference segments. Therefore we place the

virus reference segment at the bottom to aid in visualization. Since the d3.arc angle is calculated as shown in the figure below, hence the pre-processing on the angle is done.

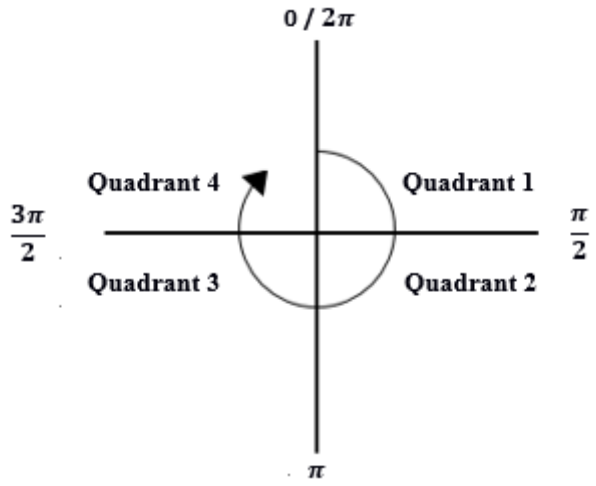


Figure 3-4-F3 D3 arc angle.

First, we assume the virus reference segment (HPV16) placed at the bottom as shown in below. Then the start angle and end angle can be found by using its angle span. $\boldsymbol{\theta}$ is the angle span in radian of the HPV16 reference segment.

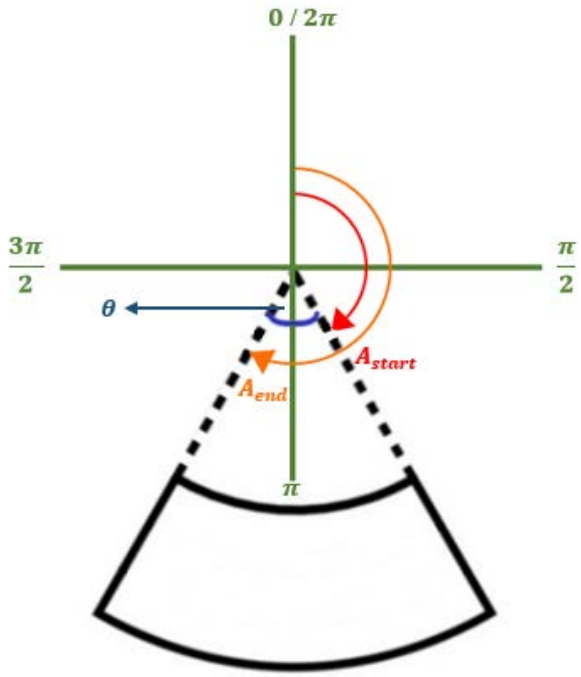


Figure 3-4-F4 Start and end angle of virus reference segment.

Start angle, $\boldsymbol{A_{start}}$ and end angle, $\boldsymbol{A_{end}}$ of virus reference segment are calculated as shown in below:

$$A_{start} = \pi - \frac{\theta}{2}$$

$$A_{start} = \pi + \frac{\theta}{2}$$

After finding the start and end angle of the virus reference segment, the host reference segments are placed gradually. Therefore, the angle can be calculated one by one. Figure 3-4-F5 shows the placement of the host reference segments after the virus reference segment. The bottom arc with blue label is the virus reference segment. And the arc is continuously added after the virus reference segment. The figure below shows there are one virus reference segment with three host reference segments.

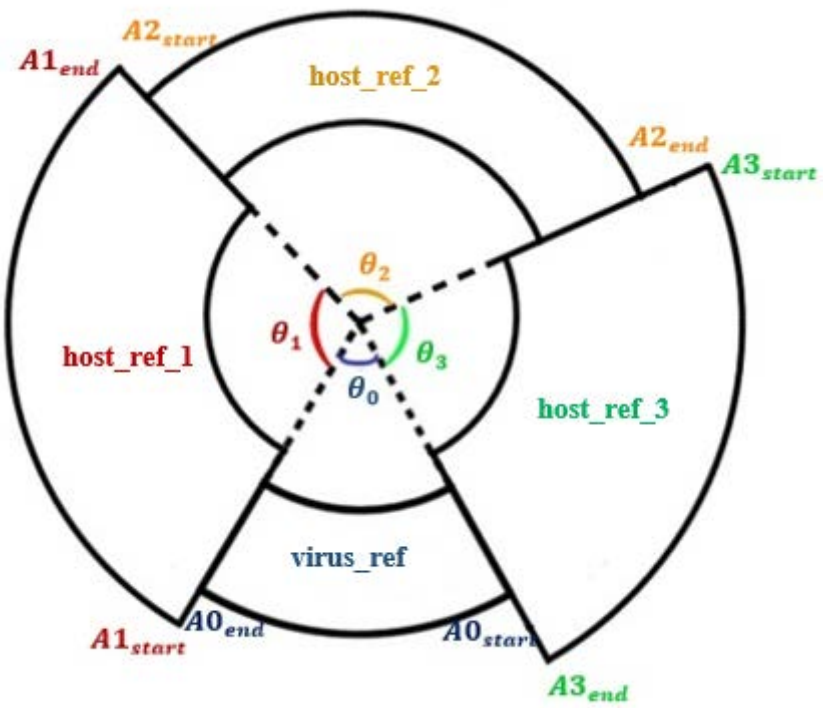


Figure 3-4-F5 Placement of the host reference segments.

In d3 arc, the angle is increasing gradually after one circle as figure shown in below:

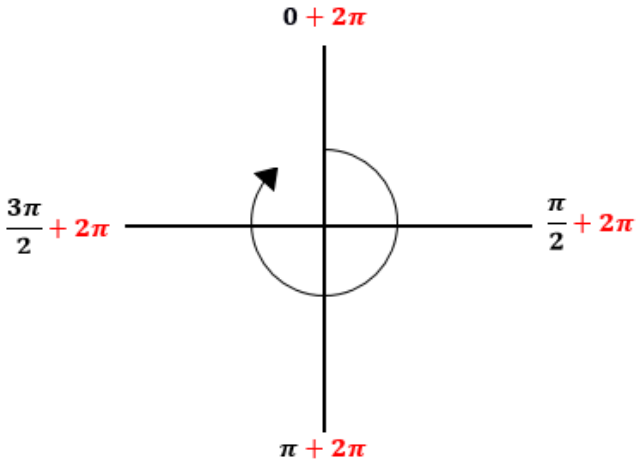


Figure 3-4-F6 Angle after one round.

Hence, the angle can be calculated as shown:

$$A[n+1]_{start} = A[n]_{end}$$

$$A[n+1]_{end} = A[n+1]_{start} + \theta_{[n+1]}$$

After computing the start and end angle, the track length of the of each reference segment is calculated. The track length is then used to find the position of the segment id "A, B, C, D, …". Figure 3-4-F7 explains about the track length. $\boldsymbol{R_{mid}}$ is the middle radius of the arc. $\boldsymbol{L_{mid}}$ is the track length based on the middle radius, $\boldsymbol{R_{mid}}$.

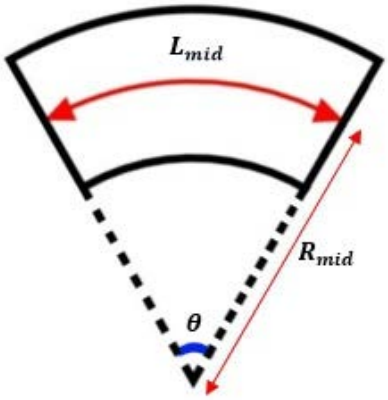


Figure 3-4-F7 Track length computation.

The track length is computed by using $\boldsymbol{R_{mid}}$ and $\boldsymbol{\theta}$:

$$R_{mid} = (R_{outer} + R_{inner})/2$$

$$L_{mid} = R_{mid} * \theta$$

Then, we need to find the span of each reference segment. Span, *S* indicates how long the reference segment is. It is calculated by using the end position subtract start position of the reference segment. The end and start point are computed by the segment id ("A"/ "B"/ "C"/….) left position and right position. The segments id are different from the reference segments. The examples of the segments id are "A", "B", "C", "D", …, while the examples of the reference segments are "chr2", "chr4", …..

Given the sub Segments file of sample id "T003" as shown in below. We can find the start and end point of the reference segment *"chr4"*.

#SampleID	Seg_type	Seg_ID	Ref_seg	left_pos	right_pos	size	res	copy_number
T003	host	A	chr4	1733000	1739090	6091	80	1.03
T003	host	B	chr4	1739090	1744503	5414	70	12.65
T003	host	C	chr4	1744503	1750435	5933	100	8.94
T003	host	D	chr4	1750435	1793573	43139	500	10.39
T003	host	E	chr4	1793573	1798238	4666	60	12.32
T003	host	F	chr4	1798238	1798257	20	1	0.71
T003	host	G	chr4	1798257	1804955	6699	100	13.68
T003	host	H	chr4	1804955	1808762	3808	100	12.97
T003	host	I	chr4	1808762	1813000	4239	100	0.97
T003	virus	a	HPV16	1	7128	7128	100	N/A
T003	virus	b	HPV16	7128	7743	616	100	N/A
T003	virus	c	HPV16	7743	7905	163	100	N/A

Table 3-4-T1 Sub-Segments file (T003).

Procedure 1 Find the starting position and ending position of the reference segment chr4

Input: segments data from Segments file, $dataSegments$
Output: Start position, $from$ and end position to of reference segment chr4.

I. $\boldsymbol{from \leftarrow -1}$
II. $\boldsymbol{to \leftarrow 0}$
III. **FOR EACH** $\boldsymbol{seg \in dataSegments}$["$\boldsymbol{T003}$"] **DO**

```
        IF seg.seg_type == "host" THEN
            IF seg.left_pos < from OR from < 0 THEN
                from ← seg.left_pos
            END IF
            IF seg.right_pos > toTHEN
                to ← seg.right_pos
            END IF
        END IF
    END FOR
```

From above procedure, we can know that the start position, $\boldsymbol{from}$ is the minimum left_pos of all the segment ids in reference segment "chr4", while the end position, $\boldsymbol{to}$ is the maximum right_pos of all the segment ids in reference segment "chr4".

With these reference segment start position, $\boldsymbol{from}$ and end position, $\boldsymbol{to}$, the span, $\boldsymbol{S}$ is calculated as shown:

$$S = to - from$$

For easy understand the concept, we take only one reference segment as an example to explain on the rest procedure. So, after finding the span of the reference segments, the segment id left position, $\boldsymbol{p_{left}}$ and segment id right position $\boldsymbol{p_{right}}$ are the left_pos and right_pos of the segment id in Segment file according to its reference segment. For example, $\boldsymbol{p_{left}}$ of the segment id "A" in reference segment "chr4" of sample id "T003" is equalled to the left_pos of segment id "A" in reference segment "chr4" of sample id "T003".

$$p_{left} = left\_pos$$

$$p_{right} = right\_pos$$

After all the informations such as track length, $\boldsymbol{L_{mid}}$, segment id left position, $\boldsymbol{p_{left}}$, segment id right position, $\boldsymbol{p_{right}}$, reference segment span, $\boldsymbol{S}$, start angle of the reference segment, $\boldsymbol{A_{start}}$, end angle of the reference segment, $\boldsymbol{A_{end}}$ and angle span, $\boldsymbol{\theta}$ are ready

to use. The start angle and end angle of each segment id ("A"/"B"/"C"/ …) are computed.

Before finding the angle, the segment id start, $\boldsymbol{p_{start}}$ and end, $\boldsymbol{p_{end}}$ are found out by using the track length, $\boldsymbol{L_{mid}}$, left position, $\boldsymbol{p_{left}}$, right position, $\boldsymbol{p_{right}}$, reference segment from position, $\boldsymbol{from}$ and the reference segment span, $\boldsymbol{S}$. The start and end imply the start, $\boldsymbol{p_{start}}$ and end, $\boldsymbol{p_{end}}$ position of the segment id in track length for drawing purpose, but the left position, $\boldsymbol{p_{left}}$ and right position, $\boldsymbol{p_{right}}$ indicate the actual segment id position in the real chromosome.

$$p_{start} = L_{mid} * \frac{(p_{left} - from)}{S}$$

$$p_{end} = L_{mid} * \frac{(p_{right} - from)}{S}$$

Figure 3-4-F8 demonstrates the positions of reference segment are calculated with respect to the track length for drawing purpose. Reference segment start position, $\boldsymbol{from}$ and end position, $\boldsymbol{to}$ are reflected to **0** and $\boldsymbol{L_{mid}}$ respectively:

1. $from$:

   $L_{mid} * \frac{(from - from)}{S}$

   $= L_{mid} * \frac{0}{S}$

   $= 0$

2. to:

   $L_{mid} * \frac{(to - from)}{S}$

   $= L_{mid} * \frac{S}{S}$

   $= L_{mid}$

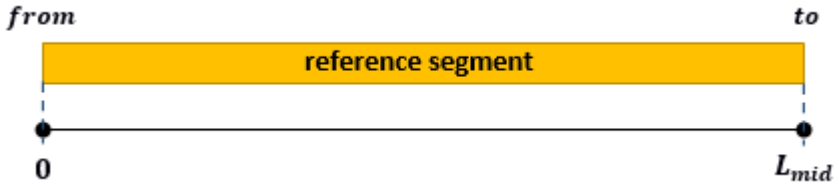


Figure 3-4-F8 Reference segment start position and end position after calculation.

Figure 3-4-F9 shows the example of the segment id "B" start position, $\boldsymbol{p_{start}}$ and end position, $\boldsymbol{p_{end}}$ according to the track length, $\boldsymbol{L_{mid}}$. The left position, $\boldsymbol{p_{left}}$ in the chromosome is translated to the $\boldsymbol{p_{start}}$ and the right position, $\boldsymbol{p_{right}}$ in the chromosome is translated to the $\boldsymbol{p_{end}}$.

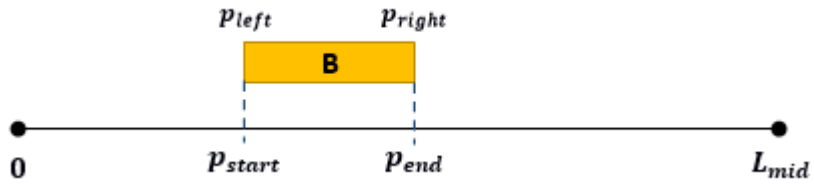


Figure 3-4-F9 Segment id "B" start position and end position after calculation.

So, the start position, $\boldsymbol{p_{start}}$ and end position, $\boldsymbol{p_{end}}$ of the segment id are used with other informations (start angle, $\boldsymbol{A_{start}}$ of reference segment and track length, $\boldsymbol{L_{mid}}$ of the reference segment) to figure the start radian, $\boldsymbol{a_{start}}$ and end radian, $\boldsymbol{a_{end}}$ of the segment id.

$$a_{start} = A_{start} + \theta * \frac{p_{start}}{L_{mid}}$$

$$a_{end} = A_{start} + \theta * \frac{p_{end}}{L_{mid}}$$

Due to there is a haplotype file which show the flow of the segment id, the pre-processing data is needed. Given that the sub-Haplotype file is provided in below, the data will be processed for drawing the graph.

#SampleID	haplotype_NO	colour	contig_NO	repeat_time	regid_string
T003	1	purple	1	1	A,
T003	1	purple	2	9	B,C,D,E,r_b,G,H,

T003	1	purple	3	3	B,E,r_b,G,H,
T003	1	purple	4	1	B,E,r_b,G,D,E,r_b,G,H,I,

Table 3-4-T2 Sub-Haplotypes file (T003)

From the table 3-4-T3, the flow of the segment id in sample T003 is recognized and also the position and the radian are stated.

segment no	genomic segments	ref_seg	start_pos	end_pos	start_radian	end_radian
1	A	chr4	$p_{start}(A)$	$p_{end}(A)$	$a_{start}(A)$	$a_{end}(A)$
2	B,C,D,E	chr4	$p_{start}(B)$	$p_{end}(E)$	$a_{start}(B)$	$a_{end}(E)$
3	r_b	HPV16	$p_{start}(b)$	$p_{end}(b)$	$a_{start}(b)$	$a_{end}(b)$
4	G,H	chr4	$p_{start}(G)$	$p_{end}(H)$	$a_{start}(G)$	$a_{end}(H)$
5	B	chr4	$p_{start}(B)$	$p_{end}(B)$	$a_{start}(B)$	$a_{end}(B)$
6	E	chr4	$p_{start}(E)$	$p_{end}(E)$	$a_{start}(E)$	$a_{end}(E)$
7	r_b	HPV16	$p_{start}(b)$	$p_{end}(b)$	$a_{start}(b)$	$a_{end}(b)$
8	G,H	chr4	$p_{start}(G)$	$p_{end}(H)$	$a_{start}(G)$	$a_{end}(H)$
9	B	chr4	$p_{start}(B)$	$p_{end}(B)$	$a_{start}(B)$	$a_{end}(B)$
10	E	chr4	$p_{start}(E)$	$p_{end}(E)$	$a_{start}(E)$	$a_{end}(E)$
11	r_b	HPV16	$p_{start}(b)$	$p_{end}(b)$	$a_{start}(b)$	$a_{end}(b)$
12	G	chr4	$p_{start}(G)$	$p_{end}(G)$	$a_{start}(G)$	$a_{end}(G)$
13	D,E	chr4	$p_{start}(D)$	$p_{end}(E)$	$a_{start}(D)$	$a_{end}(E)$
14	r_b	HPV16	$p_{start}(b)$	$p_{end}(b)$	$a_{start}(b)$	$a_{end}(b)$
15	G,H,I	chr4	$p_{start}(G)$	$p_{end}(I)$	$a_{start}(G)$	$a_{end}(I)$

Table 3-4-T3 Sample T003 data after pre-processing.

After all the sample data are processed, we know each reference segment contains what genomic segments. Table 3-4-T4 show the genomic segments of reference segment "chr4".

No.	genomic segments	ref_seg	start_pos	end_pos	start_radian	end_radian
1	A	chr4	$p_{start}(A)$	$p_{end}(A)$	$a_{start}(A)$	$a_{end}(A)$
2	B,C,D,E	chr4	$p_{start}(B)$	$p_{end}(E)$	$a_{start}(B)$	$a_{end}(E)$
3	G,H	chr4	$p_{start}(G)$	$p_{end}(H)$	$a_{start}(G)$	$a_{end}(H)$
4	B	chr4	$p_{start}(B)$	$p_{end}(B)$	$a_{start}(B)$	$a_{end}(B)$
5	E	chr4	$p_{start}(E)$	$p_{end}(E)$	$a_{start}(E)$	$a_{end}(E)$
6	G,H	chr4	$p_{start}(G)$	$p_{end}(H)$	$a_{start}(G)$	$a_{end}(H)$
7	B	chr4	$p_{start}(B)$	$p_{end}(B)$	$a_{start}(B)$	$a_{end}(B)$
8	E	chr4	$p_{start}(E)$	$p_{end}(E)$	$a_{start}(E)$	$a_{end}(E)$
9	G	chr4	$p_{start}(G)$	$p_{end}(G)$	$a_{start}(G)$	$a_{end}(G)$
10	D,E	chr4	$p_{start}(D)$	$p_{end}(E)$	$a_{start}(D)$	$a_{end}(E)$
11	G,H,I	chr4	$p_{start}(G)$	$p_{end}(I)$	$a_{start}(G)$	$a_{end}(I)$

Table 3-4-T4 Reference segment "chr4"

The variable ***segments*** is used to store the genomic segments for each reference segment.

$$segments[n][0] = start\_pos\ [n]$$

$$segments[n][1] = end\_pos\ [n]$$

where n = sequence number of genomic segments

$start\_pos$ = start position of the genomic segments

end_pos = end position of the genomic segments

After recognizing all the genomic segments of each reference segment, the sorting on genomic segments is carried out. The sorting process is based on the center of the genomic segments. We sort the ***segments*** in ascending order.

Procedure 2 Sort the *segments* according to their center

Input: Unsorted *segments*
Output: Sorted *segments*

```
I.  iTotal ← 0
II. jTotal ← 0
III. FOR i = 0 TO segments.length − 1 STEP 1 DO
        FOR j = i + 1 TO segments.length STEP 1 DO
            iTotal ← segments[i][0] + segments[i][1]
            jTotal ← segments[j][0] + segments[j][1]
            IF iTotal > jTotal THEN
                temp = segment[i]
                segments[i] = segments[j]
                segments[j] = temp
            END IF
        END FOR
    END FOR
```

3-5 Arrange Genomic Segments

For arranging the genomic segments, the idea from my supervisor, Dr Ng Yen Kaow is implemented to continue the project.

The threshold value, $\boldsymbol{T}$ for the distance is prompted from user. The value is then used to compute the possible solution of the placement of the genomic segments. The segments, $\boldsymbol{S}$ are sorted in increasing order according to their center, then the placement can carry out. Every placement of the segment is recorded as a prior state. When we place the first segment, $\boldsymbol{S_{[n-n]}}$, the solution, *state* is produced. The solution, *state* is stored for later use when placing next segment, $\boldsymbol{S_{[n-n+1]}}$.

Each *state* is an array with size of number of tracks, $\boldsymbol{N}$ which store the solution. The genomic segments are denoted as $\boldsymbol{i}$ according to their order in the segments, $\boldsymbol{S}$. The definition of the *state* is $(\boldsymbol{i_0}, \boldsymbol{i_1}, \dots, \boldsymbol{i_{N-1}})$. The initial state is set to $(-1_0, -1_1, \dots, -1_{N-1})$. For example, given that there are total three tracks, $\boldsymbol{N = 3}$, and the first segment, $\boldsymbol{S_0}$ is placed at the track 2, $\boldsymbol{k = 1}$, then the *state* is $(-1, 0, -1)$. After all the solutions, *state*s to place a segment are found out, the multiple *state*s are then used as the prior state to place the next segment. The *state* is only stored the latest segment in each track. If the new segment is placed on the track that same as the previous, the value on that track is changed to the new segment. For example if the *state* $(-1, 0, -1)$, then the segment 1 is placed to the second track, the new *state* with value $(-1, 1, -1)$ will create. Therefore, *state* (3,2,5) shows the segment 3 is the last segment in track 1, segment 2 is the last segment in track 2 and also the segment 5 is the last segment in track 3.

The set of the states that created right after the segment, $\boldsymbol{S_P}$ is placed are denoted as $\boldsymbol{\Phi_p}$. The prior set of the states, $\boldsymbol{\Phi_{p-1}}$ is used to compute $\boldsymbol{\Phi_p}$. To find the *state*, the conditions must meet. One of the conditions is the distance between the center of two segments must greater than the threshold value, $\boldsymbol{T}$.

$$(\forall_k, 0 \leq k \leq N-1) dist(A, B) > T, A \in state[k], B = S_P$$

Moreover, the segments cannot be overlapped with each other. If there is no overlapping when placing the segment on the track then the solution is possible.

Procedure 1 Find the solutions that fulfill conditions

Input: Segments, S, number of track, N, distance threshold, T and the states, Φ, size of the segments S, n
Output: The placement of the segment that fulfill conditions.

I. $\Phi_0 \leftarrow \{(\mathbf{0}, -\mathbf{1}, \dots, -\mathbf{1}), (-\mathbf{1}, \mathbf{0}, \dots, -\mathbf{1}), \dots (-\mathbf{1}, -\mathbf{1}, \dots, \mathbf{0})\}$
II. $\boldsymbol{paths_0} \leftarrow \{(\mathbf{0}, -\mathbf{1}, \dots, -\mathbf{1})\{(\mathbf{0})\}, (-\mathbf{1}, \mathbf{0}, \dots, -\mathbf{1})\{(\mathbf{1})\}, \dots (-\mathbf{1}, -\mathbf{1}, \dots, \mathbf{0})\{(\boldsymbol{N}-\mathbf{1})\}\}$
III. **FOR** $\boldsymbol{p} \leftarrow \mathbf{1}, \dots, \boldsymbol{n}-\mathbf{1}$ **DO**
    $\Phi_p \leftarrow \{\,\}$
    FOR $\boldsymbol{\alpha} \in \Phi_{p-1}$ **DO**
        FOR $\boldsymbol{k} \leftarrow \mathbf{0}, \dots, \boldsymbol{N}-\mathbf{1}$ **DO**
            $\boldsymbol{distance} \leftarrow \boldsymbol{minDist}(\boldsymbol{\alpha}, p, \boldsymbol{k})$
            IF $distance < T$ **OR** $distance == -1$ **THEN**
                CONTINUE
            END IF
            $\boldsymbol{\alpha'} \leftarrow \boldsymbol{\alpha}$
            $\boldsymbol{\alpha'}[\boldsymbol{k}] \leftarrow \boldsymbol{p}$
            $\Phi_p \leftarrow \Phi_p \cup \boldsymbol{\alpha'}$
            IF $paths[\boldsymbol{\alpha'}]$ **IS UNDEFINED THEN**
                $paths[\boldsymbol{\alpha'}] \leftarrow \{\,\}$
            END IF
            IF SIZEOF $paths[\boldsymbol{\alpha}] > 0$ **THEN**
                FOR $\boldsymbol{l} \leftarrow \mathbf{0}, \dots,$ **SIZEOF** $paths[\boldsymbol{\alpha}]$ **DO**
                    $newPath \leftarrow paths[\boldsymbol{\alpha}][l]$
                    APPEND k **TO THE** $newPath$
                    APPEND $newPath$ **TO THE** $paths[\boldsymbol{\alpha'}]$
                END FOR
            END IF
        END FOR
    END FOR
END FOR

Procedure 2 Find the minimum distance between the segments - $\boldsymbol{minDist}(\boldsymbol{\alpha}, p, \boldsymbol{k})$

Input: A state, $\boldsymbol{\alpha}$, segments, S, track position, $\boldsymbol{j}$, track length, $\boldsymbol{d}$, user input, $\boldsymbol{allow\_adjacent\_parallel}$
Output: The minimum distance between the segments $\boldsymbol{minDist}(\boldsymbol{\alpha}, p, \boldsymbol{k})$.

I. $distance \leftarrow \boldsymbol{d}$
II. **FOR** $\boldsymbol{k'} \leftarrow \boldsymbol{0}, \dots, \boldsymbol{N-1}$ **DO**
    $p' \leftarrow \alpha[\boldsymbol{k'}]$
    IF $p' < 0$ **THEN**
        CONTINUE
    END IF
    IF $\boldsymbol{k} == \boldsymbol{k'}$ **AND** $S_P[1] > S_{P'}[\boldsymbol{0}]$ **THEN**
        RETURN -1
    END IF
    IF $\boldsymbol{allow\_adjacent\_parallel}$ **AND** $\boldsymbol{isParallel}(S_P, S_{P'})$ **THEN**
        CONTINUE
    END IF
    $dist \leftarrow dist(S_P, S_{P'})$
    IF $dist < distance$ **THEN**
        $distance \leftarrow dist$
    END IF
END FOR

The user input, ***allow_adjacent_parallel*** means the segments are allowed to place in parallel. If there are no segments placed on the tracks, then the calculation of the distance between the segments, $dist$ is ignored. The calculation of the distance between the segments, $dist$ is also ignored if they are parallel to each other.

The ***isParallel*** function checks whether the segments are parallel. If so, the function will return true, if not, then it will return false. The two segments are parallel if there are overlapping. Hence, parallel means the overlap of the segments. The ***isParallel*** function return true if the percentage of overlapping is greater than 80%. Besides, the segment range is inside the largest segment then it will return true, which means the segment A and B are parallel, if $\boldsymbol{B} \in \boldsymbol{A}$.

Procedure 3 Check whether two segments are parallel - $isParallel(S_P, S_{P'})$

Input: Segment, S_P, and segment, $S_{P'}$
Output: Whether two segments are parallel.

I. **IF** $S_P[0] \leq S_{P'}[0]$ **AND** $S_P[1] \geq S_{P'}[1]$ **THEN**
    RETURN true
   END IF
II. **IF** $S_{P'}[0] \leq S_P[0]$ **AND** $S_{P'}[1] \geq S_P[1]$ **THEN**
    RETURN true
   END IF
III. **IF** $S_P[0] > S_{P'}[0]$ **THEN**
    $\boldsymbol{start} \leftarrow S_P[0]$
   ELSE
    $\boldsymbol{start} \leftarrow S_{P'}[0]$
   END IF
   IF $S_P[1] < S_{P'}[1]$ **THEN**
    $\boldsymbol{end} \leftarrow S_P[1]$
   ELSE
    $\boldsymbol{end} \leftarrow S_{P'}[1]$
   END IF
   IF $start > end$ **THEN**
    RETURN false
   END IF
   $\boldsymbol{overlap} \leftarrow |\boldsymbol{start} - \boldsymbol{end}|$
   IF $overlap \geq 0.8 * (S_{P'}[1] - S_{P'}[0])$ **OR** $overlap \geq 0.8 * (S_P[1] - S_P[0])$ **THEN**
    RETURN true
   END IF
   RETURN false

The distance between two segments, $\boldsymbol{dist}$ is calculated from the definition on the project scope. The Figure 3-5-F1 shows the description of the distance between two segments.

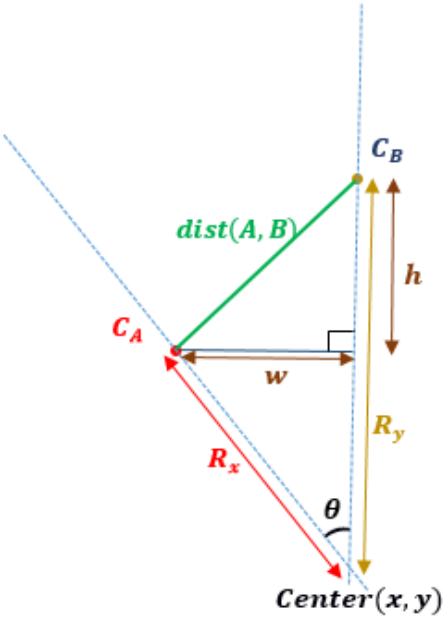


Figure 3-5-F1 Description of the distance between two segments.

The center of the segment is used to calculate the angle, $\boldsymbol{\theta}$. Given that the angle of segment **A** ($\boldsymbol{a_{start}[A], a_{end}[A]}$) and angle of segment **B** ($\boldsymbol{a_{start}[B], a_{end}[B]}$), the center angle of the segment is computed:

$$a_{center}[A] = \frac{a_{start}[A] + a_{end}[A]}{2}$$

$$a_{center}[B] = \frac{a_{start}[B] + a_{end}[B]}{2}$$

$$\theta = |a_{center}[A] - a_{center}[B]|$$

With the angle, we can compute the distance between two segments, $\boldsymbol{dist(A, B)}$.

$$w = R_x \sin\theta$$

$$h = R_y - R_x \cos\theta$$

$$dist(A, B) = \sqrt{w^2 + \mathrm{h}^2}$$

3-6 Drawing

After finding the placement of each genomic segments. The graph is pictured by the entered inputs and processed data. The inputs include the inner radius, $\boldsymbol{R_{inner}}$, outer radius, $\boldsymbol{R_{outer}}$, angle span, $\boldsymbol{\theta}$, track thickness, $\boldsymbol{h}$, number of tracks, $\boldsymbol{N}$ for each reference segments.

Each genomic segment is drawn by using d3 arc. Moreover, each reference segment is also drawn by using d3 arc. D3 arc function require the inner radius, outer radius, start angle in radian, end angle in radian. From the data pre-processing, we know each genomic segment and each reference segment start angle and end angle. So, the inner radius and outer radius of each genomic segments and each reference segments are computed.

For the reference segments, the inner radius, $\boldsymbol{ref_{inner}}$ and outer radius, $\boldsymbol{ref_{outer}}$ are calculated as shown:

$$ref_{inner} = R_{inner} - h$$
$$ref_{outer} = R_{outer} + h$$

Figure below explains the inner radius and outer radius of the reference segments.

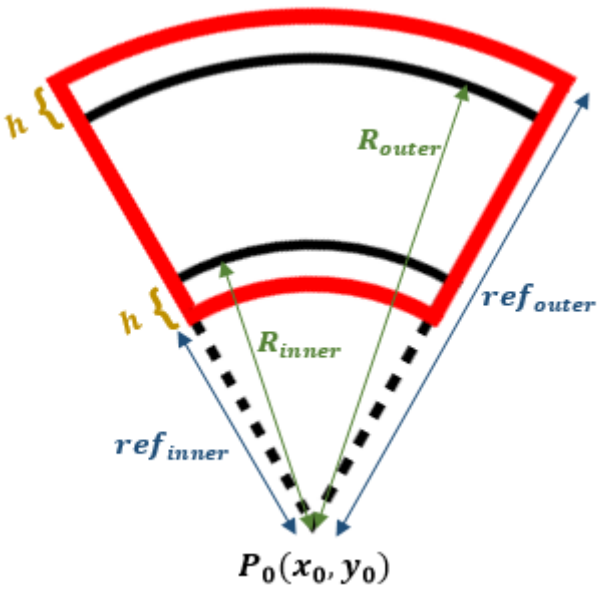


Figure 3-6-F1 Reference segment inner and outer radius.

Before finding the inner and outer radius of the genomic segments, the inter-track distance, $\boldsymbol{l}$ is found out by the $\boldsymbol{R_{outer}}$, $\boldsymbol{R_{inner}}$ and $\boldsymbol{N}$. Inter-track distance, $\boldsymbol{l}$ is the distance between the tracks. $\boldsymbol{N}$ is the number of tracks.

$$l = \frac{(R_{outer} - R_{inner})}{T - 1}$$

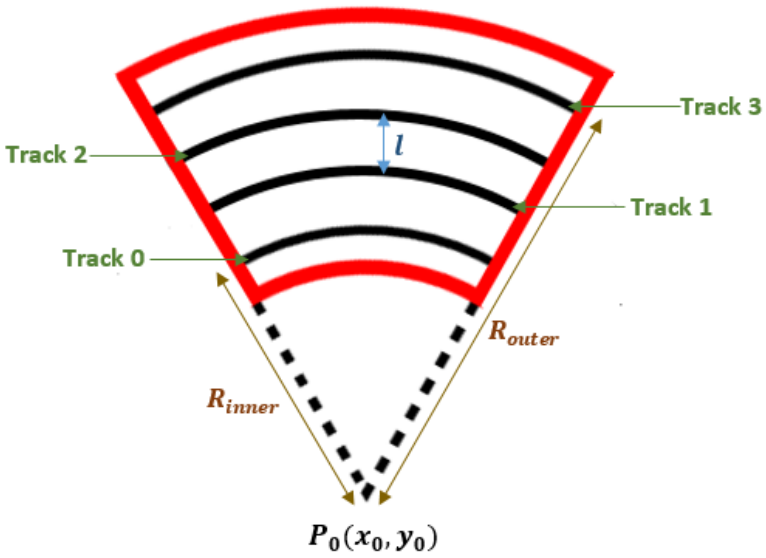


Figure 3-6-F2 Inter-track distance of 4 tracks.

In addition, the genomic segment inner radius, $\boldsymbol{seg_{inner}}$ and outer radius, $\boldsymbol{seg_{outer}}$ are calculated. The placement of the genomic segment, $\boldsymbol{assign}$ is used to draw the arc of the genomic segments. The assignment, $\boldsymbol{assign}$ indicate the track of the genomic segment assign to. It is $[\boldsymbol{0},\ \boldsymbol{N-1}]$, where $\boldsymbol{N}$ is the number of tracks. Example, if the placement of the genomic segment is first track, then $\boldsymbol{assign} = 0$.

$$R_{assign} = R_{inner} + assign * l$$

$$seg_{inner} = R_{assign} - \frac{h}{2}$$

$$seg_{outer} = R_{assign} + \frac{h}{2}$$

Where, $h = track\ thickness\ /\ thickness\ of\ the\ genomic\ segments\ arc$

Figure 3-6-F3 Genomic segment arc thickness, h.

Given that the genomic segment, "seg" is assigned to first track, $\boldsymbol{assign = 0}$. Then, the inner radius, $\boldsymbol{seg_{inner}}$ and outer radius, $\boldsymbol{seg_{outer}}$ are:

$$R_{assign} = R_{inner} + assign * l$$

$$R_{assign} = R_{inner}, \quad assign = 0$$

$$seg_{inner} = R_{inner} - \frac{h}{2}$$

$$seg_{outer} = R_{inner} + \frac{h}{2}$$

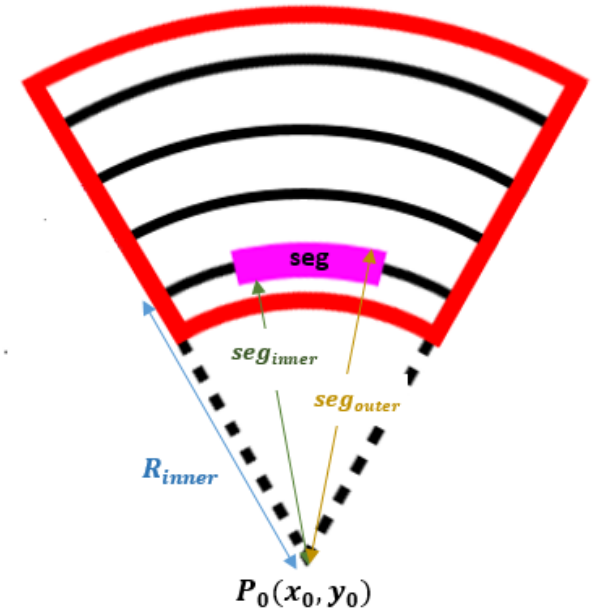


Figure 3-6-F4 Example genomic segment arc.

The start angle and end angle of the d3.arc of the genomic segments are assigned to their $\boldsymbol{a_{start}}$ and $\boldsymbol{a_{end}}$ respectively which can get on the data pre-processing. While, the start angle and end angle of the d3.arc of the reference segments are assigned to their own $\boldsymbol{A_{start}}$ and $\boldsymbol{A_{end}}$ respectively which can get on the data pre-processing.

With all the informations, the d3.arc function is written. The findings are put into the d3.arc function.

For the reference segments:

```
var referenceArc = d3.arc()
      .innerRadius(ref_inner)
      .outerRadius(ref_outer)
      .startAngle(A_start)
      .endAngle(A_end)
```

Each genomic segments:

```
var genomicArc = d3.arc()
      .innerRadius(seg_inner)
      .outerRadius(seg_outer)
      .startAngle(a_start)
      .endAngle(a_end)
```

3-7 Interaction

In order to make the circos-type visualization more attractive and user-friendly, the circos-type visualization is programmed to have the interaction. The users are able to drag the segments from one track to the others track. It means that the segment is dragged from one position to a new position. The new position of the dragged segment is then calculated and the segment is translated to the new position. Then, all the other segments are refreshed to the new position. To find out which track does the user drag to, we compute the radius of the new position and find the track that is nearest to the new position which user dragged to.

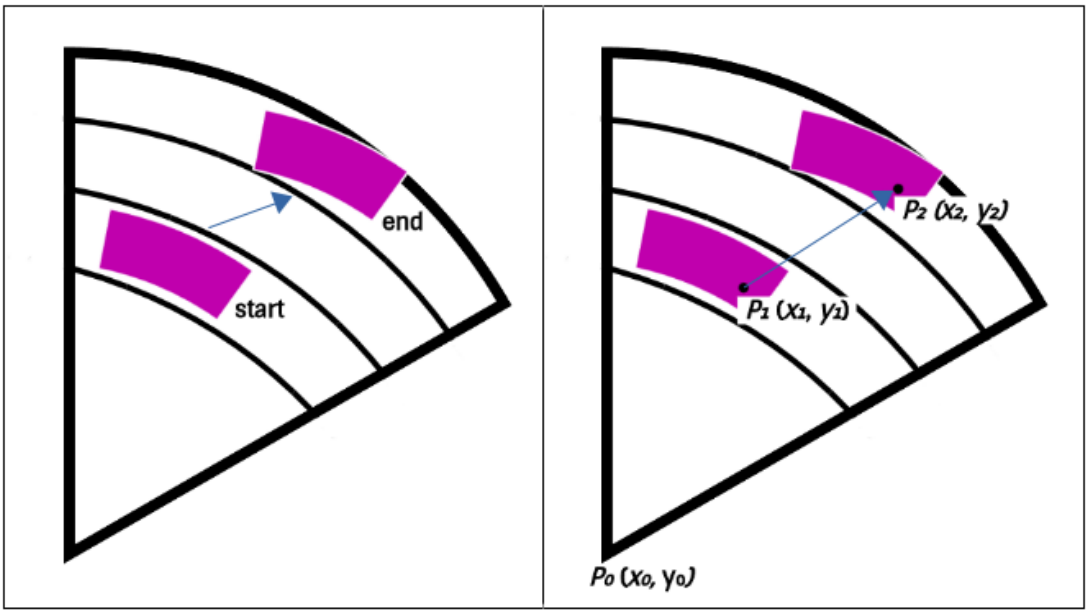


Figure 3-7-F1 Movement of a segment.

Figure 3-7-F1 shows the segment move from one track to other track. The dot shape represents the point that cursor pointed at. As user drags on the segment, the segment moves along with the cursor. The segment will be placed when the mouse up. So, the problem is which track does the segment dragged to. In Figure 3-7-F1, $\boldsymbol{P_0}$ is the origin point of the arc, $\boldsymbol{P_1}$ is the cursor point on the segment which user going to drag, and $\boldsymbol{P_2}$ is the point that the segment dragged to. By using these three point - $\boldsymbol{P_0}$, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, we can calculate where the segment dragged to.

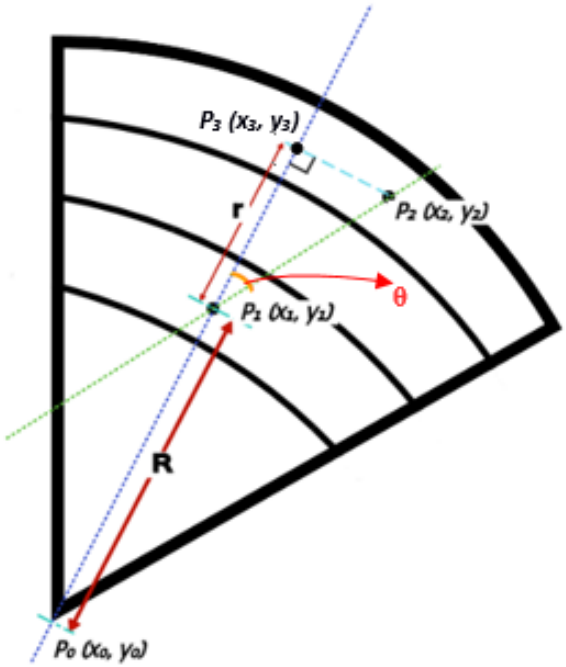


Figure 3-7-F2 Track calculation.

To determine where the segment locates after dragging, the radius is used as a parameter to calculate the final track location. $\boldsymbol{R}$ is the radius of between the segment cursor start point, $\boldsymbol{P_1}$ and the origin point, $\boldsymbol{P_0}$. $\boldsymbol{r}$ is the distance between the cursor start point $\boldsymbol{P_1}$ and the point, $\boldsymbol{P_3}$ that is the point that perpendicular to the $\boldsymbol{P_2}$. $\boldsymbol{\theta}$ is the acute angle between two lines that form by line1($\boldsymbol{P_0}$ and $\boldsymbol{P_1}$) and line2($\boldsymbol{P_1}$ and $\boldsymbol{P_2}$).

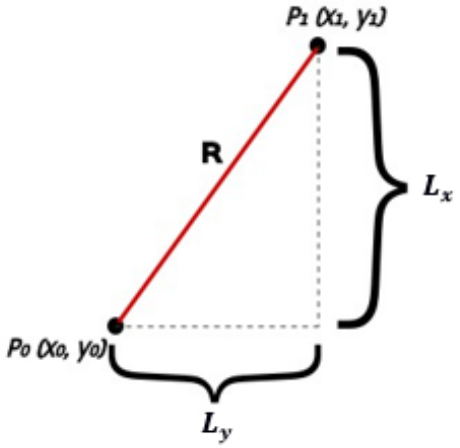


Figure 3-7-F3 Radius calculation.

The length, $\boldsymbol{R}$ between the cursor start point $\boldsymbol{P_1}$ and the origin point $\boldsymbol{P_0}$ is computed as shown:

$$L_y = |y_1 - y_0|$$
$$L_x = |x_1 - x_0|$$
$$R = \sqrt{{L_y}^2 + {L_x}^2}$$

The distance, $\boldsymbol{r}$ between cursor start point $\boldsymbol{P_1}$ and point $\boldsymbol{P_3}$. Before finding the distance, $\boldsymbol{r}$, the acute angle between two lines – line1($\boldsymbol{P_0}$ and $\boldsymbol{P_1}$) and line2($\boldsymbol{P_1}$ and $\boldsymbol{P_2}$) is computed.

Figure 3-7-F4 shows the concept on finding the angle between two lines.

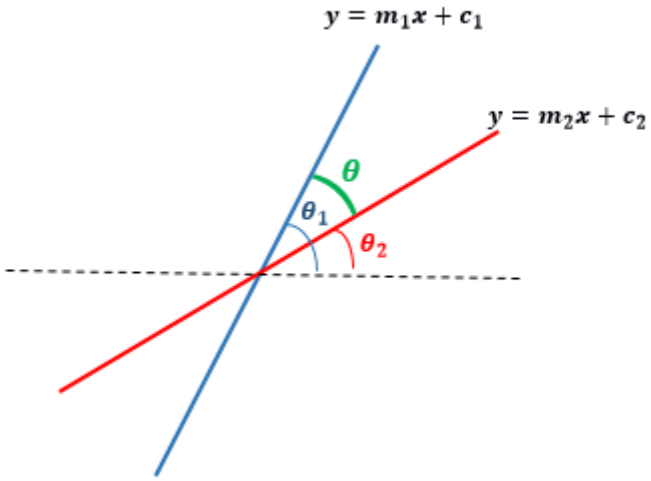


Figure 3-7-F4 Angle concept.

From Figure 3-7-F4, we know that:

$$\theta = \theta_1 - \theta_2$$
$$\tan\theta = \tan(\theta_1 - \theta_2)$$
$$\tan\theta = \tan[(\theta_1 + (-\theta_2)]$$
$$\tan\theta = \frac{\tan\theta_1 + \tan(-\theta_2)}{1 - \tan\theta_1 \tan(-\theta_2)}$$
$$\tan\theta = \frac{\tan\theta_1 - \tan(\theta_2)}{1 + \tan\theta_1 \tan(\theta_2)}$$

Since we don't know the $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$, we substitute it with $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$. Where $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ are the slope of the respective lines.

$$m_1 = \tan\theta_1$$

$$m_2 = \tan\theta_2$$

$$\tan\theta = \frac{\tan\theta_1 - \tan(\theta_2)}{1 + \tan\theta_1 \tan(\theta_2)}$$

$$\tan\theta = \frac{m_1 - m_2}{1 + m_1 m_2}$$

$$\boldsymbol{\theta = \tan^{-1}\frac{m_1 - m_2}{1 + m_1 m_2}}$$

Figure below shows the zoom in image of the Figure 3-7-F2. By using the concept shows in above, we can find the angle, $\boldsymbol{\theta}$. Then, the length, $\boldsymbol{d}$ is calculated by two points, $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$. With the angle, $\boldsymbol{\theta}$ and length, $\boldsymbol{d}$, the computation of distance, $\boldsymbol{r}$ can be carried out.

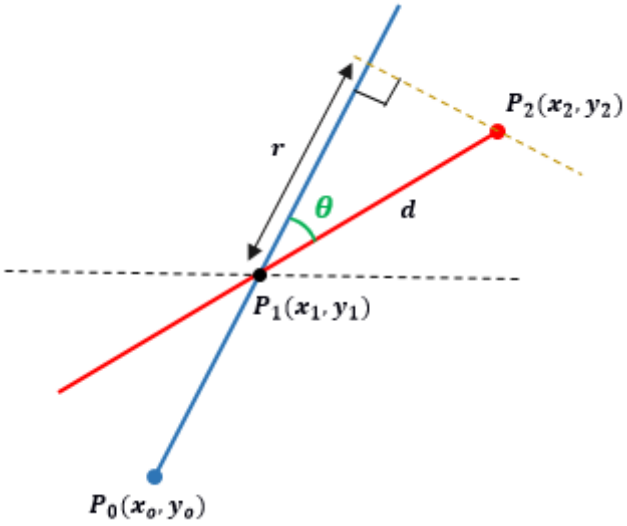


Figure 3-7-F5 Distance, r computation.

The two slope value, $\boldsymbol{m_1}$ and $\boldsymbol{m_2}$ are calculated as shown:

$$m_1 = \frac{(y_1 - y_0)}{(x_1 - x_0)}$$

$$m_2 = \frac{(y_2 - y_1)}{(x_2 - x_1)}$$

Hence, the angle, $\boldsymbol{\theta}$ is computed as shown:

$$\theta = \tan^{-1}\frac{m_1 - m_2}{1 + m_1 m_2}$$

The length, $\boldsymbol{d}$ is the distance between point $\boldsymbol{P_1}$ and point $\boldsymbol{P_2}$. The computation is same as the distance, $\boldsymbol{R}$ that shown in Figure 3-7-F3 but the only difference is the points are different. $\boldsymbol{P_1(x_1, y_1)}$ and $\boldsymbol{P_2(x_2, y_2)}$ are used to compute the length, $\boldsymbol{d}$.

$$L_y = |y_2 - y_1|$$
$$L_x = |x_2 - x_1|$$
$$d = \sqrt{{L_y}^2 + {L_x}^2}$$

By using the angle, $\boldsymbol{\theta}$ and length, $\boldsymbol{d}$, we can find the distance, $\boldsymbol{r}$. The computation is shows in below:

$$r = d\cos\theta$$

Finally, the final radius, $\boldsymbol{R'}$ can be found out and the nearest track is computed. Before we compute the final radius, $\boldsymbol{R'}$, we need to know the movement of the cursor, whether it will increase by the distance, $\boldsymbol{r}$ or decrease by the distance, $\boldsymbol{r}$. Hence, the condition test is carried out. Since $\boldsymbol{P_1}$ is the start cursor point, and $\boldsymbol{P_2}$ is the end cursor point, the distance between each point to the origin point, $\boldsymbol{P_0}$ can be discovered and used as a condition to find the final radius, $\boldsymbol{R'}$.

The distance between $\boldsymbol{P_2}$ and $\boldsymbol{P_0}$, $\boldsymbol{D_{02}}$ is computed as shown:

$$L_y = |y_2 - y_0|$$
$$L_x = |x_2 - x_0|$$
$$D_{02} = \sqrt{{L_y}^2 + {L_x}^2}$$

The Figure 3-7-F6 show the distance between $\boldsymbol{P_2}$ and $\boldsymbol{P_0}$, $\boldsymbol{D_{02}}$ greater than the distance between $\boldsymbol{P_1}$ and $\boldsymbol{P_0}$, $\boldsymbol{R}$.

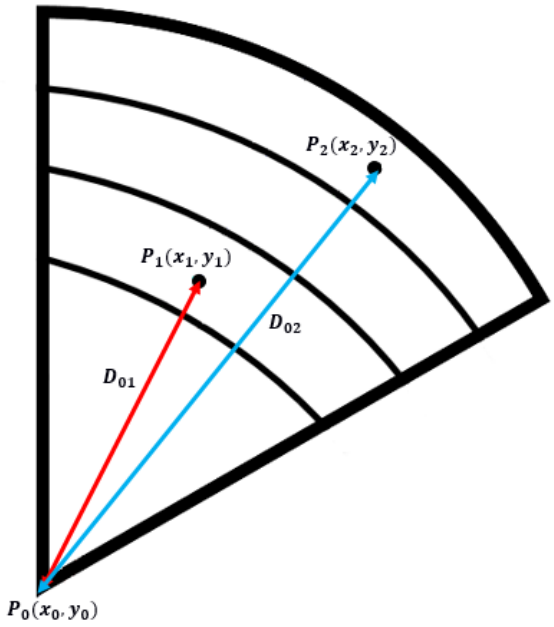


Figure 3-7-F6 Distance, $\boldsymbol{D_{02}}$ greater than distance, $\boldsymbol{R}$.

Figure 3-7-F6 shows the distance between $\boldsymbol{P_2}$ and $\boldsymbol{P_0}$, $\boldsymbol{D_{02}}$ smaller than the distance between $\boldsymbol{P_1}$ and $\boldsymbol{P_0}$, $\boldsymbol{R}$.

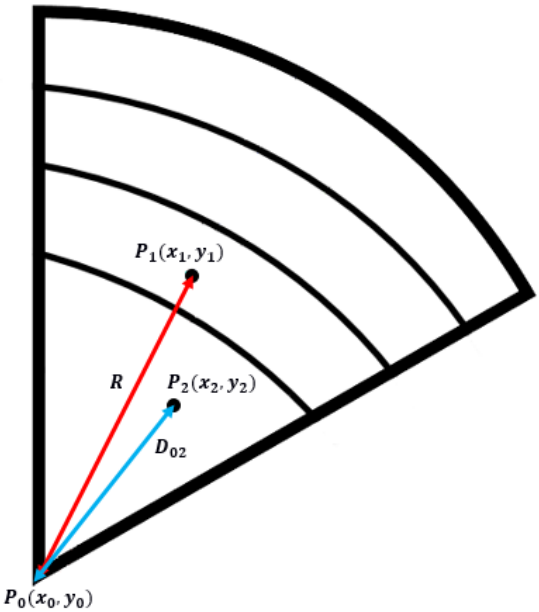


Figure 3-7-F7 Distance, $\boldsymbol{D_{02}}$ smaller than distance, $\boldsymbol{R}$.

In this case, if the distance of the $\boldsymbol{P_2}$ and $\boldsymbol{P_0}$, $\boldsymbol{D_{02}}$ is greater than the distance of the $\boldsymbol{P_1}$ and $\boldsymbol{P_0}$ which also known as $\boldsymbol{R}$, then the final radius, $\boldsymbol{R'}$ is the addition of distance, $\boldsymbol{R}$ and distance, $\boldsymbol{r}$. On the other way round, if the distance of the $\boldsymbol{P_2}$ and $\boldsymbol{P_0}$, $\boldsymbol{D_{02}}$ is smaller than the distance, $\boldsymbol{R}$, then the final radius, $\boldsymbol{R'}$ is the subtraction of distance, $\boldsymbol{R}$ and distance, $\boldsymbol{r}$.

Procedure 1 Calculate the final radius, $\boldsymbol{R'}$

Input: The distance of $\boldsymbol{P_0}$ and $\boldsymbol{P_1}$, R, the distance of $\boldsymbol{P_0}$ and $\boldsymbol{P_2}$, D_{02} and the distance, r
Output: The final radius, R'

I. $R' \leftarrow \mathbf{0}$
II. **IF** $D_{02} \geq R$ **THEN**
$R' = R + r$
ELSE
$R' = R - r$
END IF

Next, the nearest track, $\boldsymbol{N'}$ is found out. We compare the final radius, $\boldsymbol{R'}$ with the middle radius of each track, $\boldsymbol{R_{mid}[n]}$. The difference between the final radius, $\boldsymbol{R'}$ and $\boldsymbol{R_{mid}[n]}$ is the lowest then the nearest track, $\boldsymbol{N'}$ is $\boldsymbol{n}$.

The middle radius is calculated as shown:

$$l = \frac{(R_{outer} - R_{inner})}{T - 1}$$

$$R_{mid}[n] = R_{inner} + (n * l)$$

Where $T = number\ of\ tracks$

$l = intertrack\ distance$

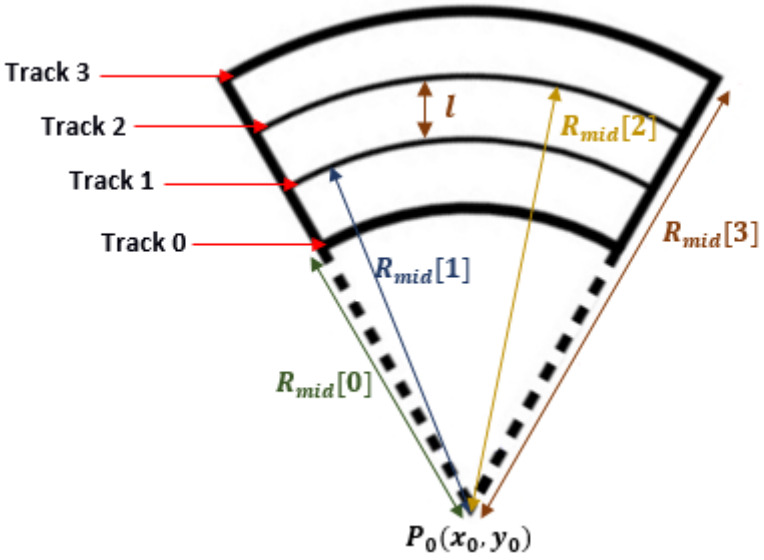


Figure 3-7-F8 Radius of the tracks.

Procedure 2 Find the nearest track, N'

Input: The radius of each track, $R_{mid}[n]$, the final radius, R' and the number of tracks, N
Output: The nearest track, N'

I. $min \leftarrow$ **1000**
II. **FOR** $i = 0$ **TO** N **STEP** 1 **DO**
    $difference \leftarrow |R' - R_{mid}[i]|$
    IF $difference \leq min$ **THEN**
        $min = difference$
        $N' = i$
    END IF
END FOR

After that, the genomic segment that had been dragged is moved to the new track, N'. And also the others genomic segments are repositioned to the new track which uses the solution of the arranging genomic segments.

Besides, the genomic segments are programmed to be able to double click for the lockin purpose. If the genomic segment is double clicked, then the position of the genomic

segment is fixed and it cannot be dragged unless double click the genomic segment again.

When the genomic segment A is locked, and the genomic segment B is dragged to the new position, the repositioning of the genomic segments is done except of the genomic segment A and genomic segment B. Reposition is the process of finding the possible genomic segments placement with the condition. So, if the genomic segment A is fixed on the first track and genomic segment B is dragged to the second track, then the reposition will find the solution of the placement of all the genomic segments where B=1 and A=0.

If the segment A is locked on track $\boldsymbol{n}$, then the segment B cannot drag to the track $\boldsymbol{n}$, when:

1. $\boldsymbol{A} \in \boldsymbol{B}$ or $\boldsymbol{B} \in \boldsymbol{A}$
2. $\boldsymbol{a_{end}[A] > a_{start}[B]}$ or $\boldsymbol{a_{start}[A] < a_{end}[B]}$

CHAPTER 4 METHODOLOGIES AND TOOLS

4-1 Design Specifications

4-1-1 Methodologies and General Work Procedures

The realization of this project will show the drawing and arranging the cross-intersecting genomic segments for circos-type visualization. Past research about the problem is studied and used as a basis for the drawing. A new method will be implemented to allow the drawing of the genomic segments. The correctness of the idea will be verified by using the software prototype and empirical analysis.

This project will be divided into several phases. First, study the previous research on arranging problem for circos-type visualization. Second, formulate an algorithm to draw the genomic segments. Lastly, validate the drawing with the program written. The different type of information will be used to test the algorithm, in order to check the correctness of the idea. The steps are looped until the final best approach to draw the circos-type visualization.

4-1-2 Tool to use

Ubuntu

A Linux-based operating system used to run the program.

Git

A version control system for updating changes in the files and managing the files among multiple people. It is used to clone the files.

Django

A web framework used in this project.

Postgres SQL

An open-source database is used to handle the database.

Visual Studio Code

A software used to write the algorithm of the project.

Python2.7/pip

Used in running the server,

D3.js

A JavaScript library for visualizing data for the circos-type visualization.

SVG

Scalable Vector Graphics is an XML-based markup language for describing two-dimensional vector graphics. It is used to form a circos-type visualization.

TypeScript

A type of JavaScript language used to determine the correctness of the idea of the project.

CSS (Cascading Style Sheets)

A style sheet language used in presenting the information.

SASS (Syntactically awesome stylesheets)

A style sheet language that compiled into CSS.

4-1-3 User requirements

The interaction on the circos-type visualization is required. The number of tracks used to allocate the cross-intersecting genomic segments in the circos-type visualization must as little as possible.

4-1-4 System Performance Definition

The targeted improvements of arranging the cross-intersecting genomic segments for circos-type visualization can be concluded as:

- The number of tracks used to arranging genomic segments as little as possible. This will relieve the limitation of the size of the circos.

- The interaction on the circos-type visualization must be accurate and interactable.

4-1-5 Verification Plan

To test this project, the information of different patient with same disease is used. The genomic segments used as an input to develop circos-type visualization. The different disease information is used to test the algorithm. The interaction is test for multiple trial.

4-2 Implementation Issues and Challenges

Various challenges are faced throughout this project. This project is an interdisciplinary research that involves computer science, biology and mathematics – bioinformatics. Therefore, a lot of knowledge need to cover for a better understanding of this problem. Moreover, d3 is new visualization library to learn. Besides, there are a substantial amount of researches on allocation problem, but most of them are allocation in rectangle shape not circular shape, hence it is lack of resources of this project. Moreover, the cursor handling is difficult for d3 and the interaction hard to implement when there are a lot of the segments.

CHAPTER 5 RESULT

5-1 Results

The circos-type visualizartion will generate once the system runs.

Figure 5-1-F1 Result of the circos-type visualization.

When the user clicks on the segment, the other segments of the same reference segment (big arc) will have the opacity of 0.3 and the selected segment will have the opacity of 1. Besides that, the boundary of the track is drawn, which to give the guide to the user boundary position to drag. Figure 5-1-F2 shows the changes when click event is triggered.

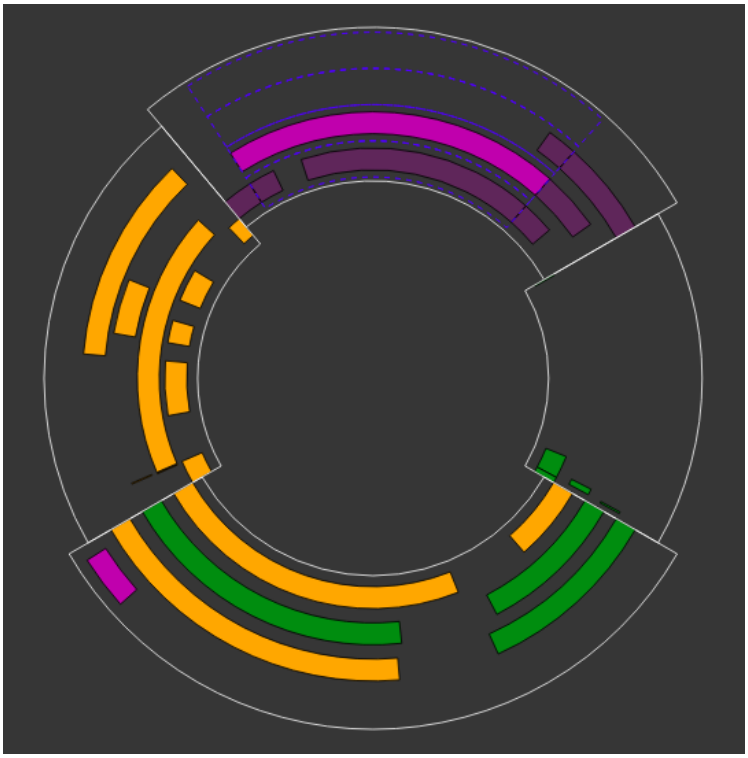


Figure 5-1-F2 Changes when click event is triggered.

Figure below shows the changes when double click event is done. Double click on the segment will restrict the segment from drag. Once the segment is double clicked, the arc (segment) will has the stroke with color red and width 4. This shows the segment is locked to the position.

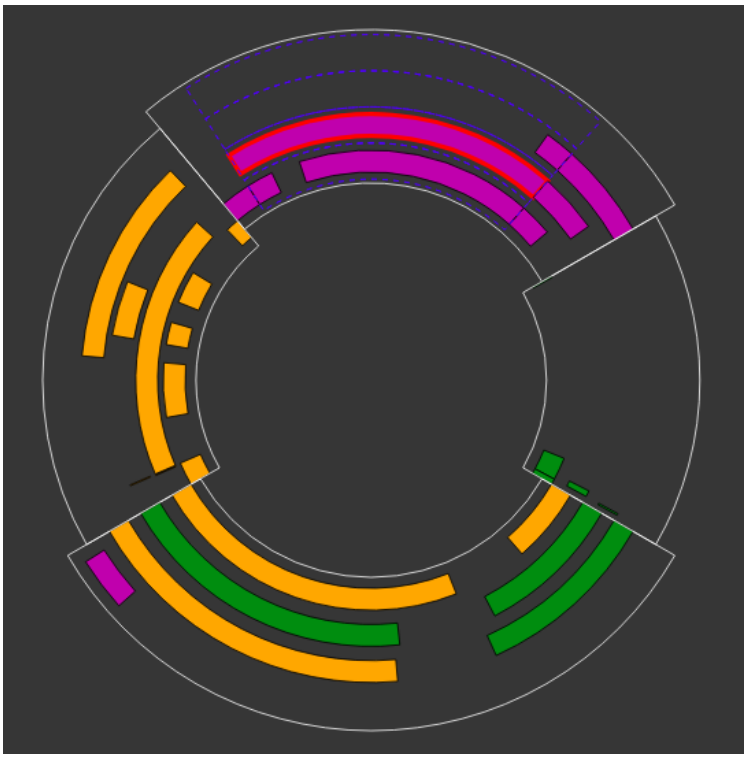


Figure 5-1-F3 Changes when double click event is triggered.

To unlock the segment on the position, we need to double click again the segment. The arc will change to the normal arc without the stroke red.

The alert message will show to the user when the segment A is dragged to the track of the segment B that is locked, if:

1. $\boldsymbol{A} \in \boldsymbol{B}$ or $\boldsymbol{B} \in \boldsymbol{A}$
2. $\boldsymbol{a_{end}[A] > a_{start}[B]}$ or $\boldsymbol{a_{start}[A] < a_{end}[B]}$

It will show the segment cannot put in that track, since it is occupied by the other parallel segment. Then the dragged segment will reposition to the origin track.

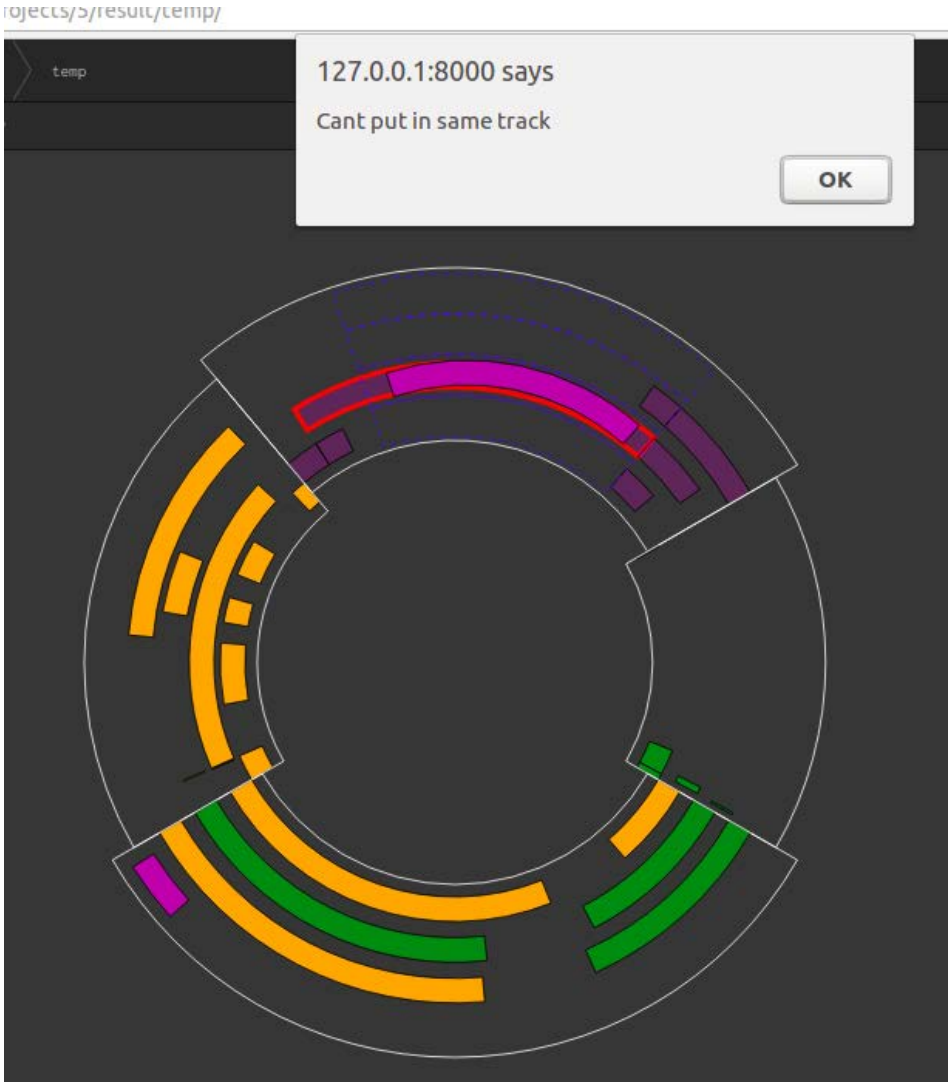


Figure 5-1-F4 Segment is dragged to the track that occupied.

All the segments are repositioned once drag event success except for the segments that are locked on the track. The segments in same big arc (reference segment) will also change to the new color.

Figure 5-1-F5 Reposition of the segments.

CHAPTER 6 CONCLUSION

6-1 Project Review

The research on human disease issues become more concerned by the public nowadays. Circos-type visualization gives the better knowledge on the effect of the virus on the human genomic segments. The patient genomic segments will be restructured by the virus segments. This is because of the mutation occurred. Circos-type visualization would make the users more understand the arrangement of genomic segments after infected by the virus.

In this project, we generate the circos-type visualization with interaction to better display the graph to the user. With the more readable and understandable graph, the relationship of the genomic segments can be explored easily. Therefore, the medical scientist can get this information to develop the methods that can treat cancer-related disease.

Besides, this project will solve the problem of how to arrange the cross-intersecting genomic segments for circos-type visualization. Hence, the new algorithm is going to develop to show the genomic segments in circle. This will help the genetic researchers to show their genetic information in circular form. The researchers can present their studies in a more attracting way.

6-2 Discussion

There are some problems faced when developing this project. Due to the framework used is a large complex program, the framework must be loaded correctly and completely. If there are any error in the loading process, it would cause the problems on the later stage. Therefore, the framework must be loaded attentively.

The typescript documentation available in the web are limited while the javascript documentations are plentiful. The solution provided all are in javascript. There are also not much examples online that could serve as references.

In addition, there is the issue of lack of d3 version 4 documentation. The most available d3 documentation on the web are in version 3. The syntax of d3 version 4 and version

3 are different. Hence, it is hard to find the solution to develop this project. A lot of time is spent to fix the minor bugs due to the version difference.

In handling the interaction on the circos-type visualization, the entire project progress is slowed down. This is because the lack of documentation provided. The drag event is hard to handle due to the cursor point is hard to handle.

6-3 Future Work

The circos-type visualization only arrange the genomic segments on the circos-type visualization, but no have the relationship between the genomic segments. So, the future improvements on showing the relationship between the genomic segments are needed.

Furthermore, the drag event allows the user to drag out the undesirable position example - outside the allowed position, hence the restriction could be added to the drag event in future. Besides, more interactivity can be added to the circos-type visualization such as change the color, reposition the reference segments (big arc) and so on.

BIBLIOGRAPHY

Robson, J.M., 1974. Bounds for Some Functions Concerning Dynamic Storage Allocation. Journal of the ACM, vol. 21, no. 3, pp. 491-499.

Knuth, D.E., 1973. Fundamental Algorithms: The Art of Computer Programming. Addison-Wesley Pub, vol. 1.

Garey, M.R. & Johnson, D.S., 1979. Computers and Intractability; A Guide to the Theory of NP-Completeness. Freeman.

Kierstead, H.A., 1988. The Linearity of First-Fit Coloring of Interval Graphs. SIAM J. Disc. Math., no. 1, pp. 526-530.

Kierstead, H.A., 1991. A Polynomial Time Approximation Algorithm for Dynamic Storage Allocation. Discrete Mathematics, no. 88, pp. 231-237.

Gergov, H., 1996. Approximation Algorithms for Dynamic Storage Allocation. European Symp. on Algorithms (ESA ’96), Springer, LNCS 1136, pp. 52-61.

Gergov, H., 1999. Algorithms for Compile-Time Memory Optimization. ACM-SIAM Symposium on Discrete Algorithms, pp. 907-908.

Li, S.C., Leong, H.W. & Steven, K.Q., 2004. New Approximation Algorithms for Some Dynamic Storage Allocation Problems.

Martin, I.K., Jacqueline, E.S., Joseph, C., Randy, G., Doug, H., Steven, J.J. & Marco, A.M., 2009. Circos: An information aesthetic for comparative genomics.

Circos, 2009. Dynamic Formatting With Run-Time Rules. [digital image] Available at: http://circos.ca/ [Accessed 12 Nov. 2017]

Zhou, Y.Y., 2014. Plain Text Input File. [digital image] Available at: https://www.slideshare.net/mkim8/circos-33052984 [Accessed 12 Nov. 2017]

Circos, 2009. Various Sample Image. [digital image] Available at: http://circos.ca/images/samples/ [Accessed 12 Nov. 2017]

Circos, 2009. Circos Course. [digital image] Available at: http://circos.ca/documentation/course/ [Accessed 12 Nov. 2017]

Zhang, H., Meltzer, P., & Davis, S., 2013. BMC Bioinformatics.

ARRANGING CROSS-INTERSECTING GENOMIC SEGMENTS FOR CIRCOS-TYPE VISUALIZATION

Introduction

Circos-type visualization is a popular way to visualize the genomic data.Traditionally,there are some tools to create circos-type visualization, such as CIRCOS, RCircos, but the relationships (lines) within the circle that developed by the previous circos-type visualization are dazzling. Therefore, this project is carried out to create a new circos-type visualization. This project is focus on the find the allocation problem of cross-intersecting genomic segments and the. The circos-type visualization is generated with the interaction.

Problem Statement

- Existing circos-type visualization more hard to visualize the relationship between the cancer virus and human chromosome.
- Lack of researches on the allocation problem for circle shape.

Motivation

- Allow the segments to allocate the the circle
- Allow the genetic researcher to present their data in more understandable and interested way.
  - due to genomic data too large and hard to visualize in 2D graph.
- Allow the user to know the relationship between the human gene and virus.
  - help medical scientist to investigate the methods to treat cancer

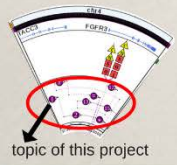
Objective

- To formulate the algorithm to allocate the segments in circle.
- To generate the circos-type visualization with interaction
- To optimize the tracks used to arrange all the segments.
- To increase the clarity of the circos-type visualization.

Methods

- Past researches about the problem are studied and used as the basis to create a new method.
- The real genomic data are used to test the algorithm.
- A software prototype will be built and the system is run to verify the idea.
- The circos-type visualization is developed
- The interaction on the visualization graph are added.

Overview

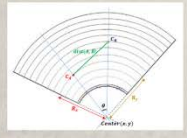


topic of this project

Due to the size of the circos-type visualization is limited.

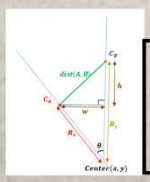
Hence the tracks used to arrange the segments must be optimized. Changes made on one part of the pie will affect the other parts.

The user input is used to generate the visualization graph, such as the inner radius and outer radius of the arc, number of tracks to place the segment, the angle span of each arc, track thickness and also the distance threshold



For the placement of the segments, the distance between two segments must be larger than the distance threshold that key in by the user. If the distance is lesser than the threshold then the placement is impossible

$$|\{min_B dist(A,B) > T\}||\{min_A dist(A,B) > T\}|$$



$$w = R_x \sin\theta$$

$$h = R_y - R_x \cos\theta$$

$$dist(A,B) = \sqrt{w^2 + h^2}$$

The distance between 2 segments is calculated as shown in the left.

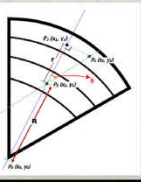
The nearest track is computed when the segment is dragged to the new position

$$r = d\cos\theta$$

$$L_y = |y_2 - y_1|$$

$$L_x = |x_2 - x_1|$$

$$d = \sqrt{{L_y}^2 + {L_x}^2}$$



#SampleID	contig_NO	repeat_time	regid_string	numbering
T003	1	1	A,	1[A],
T003	2	9	B,C,D,E,r_b,G,H,	2[B,C,D,E], 3[r_b], 4[G,H],
T003	3	3	B,E,r_b,G,H,	5[B], 6[E], 7[r_b], 8[G,H],
T003	4	1	B,E,r_b,G,D,E,r_b,G ,H,I,	9[B], 10[E], 11[r_b], 12[G], 13[D,E], 14[r_b], 15[G,H,I] ,

#SampleID	Seg_type	Seg_ID	Ref_seg	left_pos	right_pos	size	res	copy_number
T003	host	A	chr4	1733000	1739090	6091	80	1.03
T003	host	B	chr4	1739090	1744503	5414	70	12.65
T003	host	C	chr4	1744503	1750435	5933	100	8.94
T003	host	D	chr4	1750435	1793573	43139	500	10.39
T003	host	E	chr4	1793573	1798238	4666	60	12.32
T003	host	F	chr4	1798238	1798257	20	1	0.71
T003	host	G	chr4	1798257	1804955	6699	100	13.68
T003	host	H	chr4	1804955	1808762	3808	100	12.97
T003	host	I	chr4	1808762	1813000	4239	100	0.97
T003	virus	a	HPV16	1	7128	7128	100	N/A
T003	virus	b	HPV16	7128	7743	616	100	N/A
T003	virus	c	HPV16	7743	7905	163	100	N/A

GIven that input file, the data is processing and it is used to generated the circos-type visualization

Results

Circos-type visualization

When segment is clicked the other segments will have 0.3 opacity

When segment is double clicked, the position is fixed and it cannot be dragged

When the segment is dragged to the position that had been occupied

All the segments from same reference segment (big arc) is repositioned except the locked segment

Discussion

- The interactive graph are the product of the project, it allow the user to drag the segment to other position (track).This will result the graph display in what the user needs
- The graph will display the segments in the optimized position that used up as less track as possible.

Bachelor of Computer Science (Hons)
TAN SEE YEE FYP2
Universiti Tunku Abdul Rahman