InteGene: An Integer Linear Programming Tool for Discovering Approximate Gene Clusters

Princess Danielle V. Florendo[1]
[1]*Department of Physical Sciences and Mathematics*
University of the Philippines Manila
Manila, Philippines
pvflorendo@up.edu.ph

Geoffrey A. Solano[1,2]
[2]*Algorithms and Complexity Laboratory*
University of the Philippines Diliman
Manila, Philippines
gasolano@up.edu.ph

***Abstract*—The opportunity of finding conserved segments in different species has increased with the increasing availability of completely sequenced genomes. Similarity of different parts or regions of different genomes suggests evolutionary relationships among different species and might foretell functional roles which prevented genes to separate. These similar segments are called conserved gene clusters or gene clusters. Approximate Gene Cluster Discovery Problem (AGCDP) is the problem of finding genes that are kept together in different species. Presented in this study is an Integer Linear Programming (ILP) formulation of the AGCDP. Since ILP is proven to be NP-complete, the study also made use of LP Relaxation since linear programming problems can be solved in polynomial time. The software used Java for the interface and other functionalities and R for solving ILP. InteGene, the tool produced by the study, can provide the user the best clusters given a set of data and input constraints which can be further tested for biological significance.**

***Index Terms*—Genome, Gene, Gene Clusters, Approximate Gene Cluster Discovery Problem, Integer Linear Programming, LP Relaxation**

I. Introduction

In the field of comparative genomics, a field in biology that focuses in the comparison of genome sequences of different species, it is imperative to find how different species are interconnected at the genetic level. With the increasing availability of completely sequenced genomes, the opportunity of finding conserved segments in different species has increased as well. Different biological processes such as gene rearrangements, gene transfer, gene duplication, and gene loss result to changes in gene order. And over time, the gene order and gene complement of the genomes that initially have the same gene order and gene content will diverge [6], [7]. However, gene order is not random [2]. Similarity of different parts or regions of different genomes suggests evolutionary relationships among different species and might foretell functional roles which prevented genes to separate [6]. These similar segments are called conserved gene clusters or gene clusters.

Gene clusters, defined as ”genomic regions that share a common ancestor [3]”, are set of genes in two or more genomes [8]. Comparative genomics suggests that genes that stay conserved in different species imply a biological function [9]. Biological reasons are due to being part of a biochemical network, co-expression, functional pressure, and evolutionary proximity [10] [11]. Through identification of common gene clusters among different organisms, it can be identified how these organisms are related at the genetic level. Different models for gene clusters have been defined such as r-window clusters, gene teams or max-gap clusters, and common intervals [2]. Different approaches of finding gene clusters depending on the model have materialized in the past years.

An integer linear programming approach to the Approximate Gene Cluster Discovery Problem (AGCDP) was presented in a study by Rahmann and Klau in 2008 [12]. The approximate gene cluster discovery problem aims to find genes that are kept most likely together. Although ILP generally do not produce efficient algorithms, it provides a general framework to identify gene clusters of different models. The advantage of such formulation versatility of the objective function defined. The formulation is general in the sense that the constraints can be easily tweaked to find max-gap clusters, r-window clusters, and common intervals in permutations and sequences. In succeeding years, further studies were done on AGCDP It was proven NP-hard in a study by Cabunducan et. al. [16] while its graph theoretic aspects were explored in [1].

The goal of the study is to develop a system that implements the integer linear programming formulation for AGCDP in [12] and test it using available data on genomes and gene clusters. The software tool can help the researchers identify approximate gene clusters.

A. This Study

InteGene is a tool for discovering approximate gene clusters will be produced. The tool uses Integer Linear Programming (ILP) and also allows relaxing the integral constraints in finding an optimal gene cluster provided a set of genomes and formulation constraints as input. The tool allows the researcher to:

1) Input genomic data of the species in csv format.
2) Preview the transformed data from genes to integers.
3) Choose whether to use Integer Linear Progamming or LP relaxation.
4) Choose whether to use the basic integer linear programming formulation or the modified formulation for

finding either common intervals, max-gap clusters, or r-windows.

5) Input the constraints that will be used in ILP minimization such as:
   a) size range $[D^-, D^+]$ for the reference genome
   b) integer weights $w^- \geq 0$ and $w^+ \geq 0$ for the cost of missing and additional genes
   c) gap size g if user is interested in finding max-gap clusters
   d) size k if user is interested in finding r-windows
6) View the generated gene clusters.
7) View user guide on how to use the tool.
8) Export the results in a PDF file.
9) Export the results in csv format.

Identification of gene clusters can provide information about the relationship of different species. Gene clusters may also be associated to a function, a trait, or a disease. However, manual identification of gene clusters among genomes can be tedious since the number of genes in a chromosome range from hundreds to thousands. By the help of a software tool, the process of identifying these gene clusters can be performed faster.

Proprietary tools for solving integer linear programming problems have been developed such as CPLEX of International Business Machines (IBM). The ILP formulation derived by Rahmann in [12] was demonstrated using CPLEX. However, no tool especially designed for finding gene clusters using integer linear programming has been made.

Upon discovery of optimal gene clusters using the software, new researches can be conducted to verify if the approximate gene clusters have a biological significance.

II. Preliminary Concepts

A. Gene

A gene is a DNA segment that encodes for proteins and RNAs [17] [18]. In this study, genes are represented as positive integers. Genes that are homologs of each other are represented with the same integer. Homologs are genes that occur in the same genome or across different genomes that account for the same function. Orthologs are homologs which are conserved sequences of genes found in different genomes predicted to have come from a single ancestor. Paralogs are homologs which are conserved sequences found in the same genome due to gene duplication [19].

Genes are stored in molecules called chromosomes which consist of two DNA strands in opposite directions. In these strands are where genes are located. A gene's orientation is dependent on which strand it is located. The orientation tells how the information is read, either positive(+) or negative(-). Gene order is the sequential location of a gene in a chromosome. Changes in gene order and gene orientation are due to gene rearrangements [20]. In this study, gene orientation is ignored since orientation is sometimes not conserved.

A gene is represented as an integer $g \in \mathbb{Z}_+$. The gene universe $U = \{1, 2, ...N\}$ is the set of all genes with existing homologs where N is the total number of genes in consideration. Genes that do not have a homolog known is represented as the integer 0.

B. Genome

A genome constitutes to the complete DNA of an organism [21]. A genome $g = (g_1, g_2, ...g_n)$ is a sequence of n genes. A set of genomes is represented by $G = \{g^1, g^2, ...g^m\}$ where m is the total number of genomes in the set. Given a genome g^i where g^i is the ith genome in the set of genomes G, the number of genes in g^i or its length is denoted by $|g^i| = n_i$.

C. Linear Interval and Gene Content

A linear interval is a set of genes that occur consecutively in a genome. A linear interval J in genome g^i is represented as an index set which can be empty $J = \emptyset$ or non-empty $J = \{j, j+1, ...k\}$. It can also be denoted as $J = [j : k]$ where $1 \leq j \leq \ k \leq n$. The length of a linear interval $J = [j : k]$ is $|J| = k - j + 1$. The gene content of a linear interval is the set of all genes that are included in the interval. The corresponding gene content of linear interval $J = [j : k]$ is the set $G_J = \{g_j, ...g_k\}$.

D. Models of Gene Clusters

Gene clusters, defined as "genomic regions that share a common ancestor [3]", are set of genes in two or more genomes [8]. In the context of the study, a gene cluster is a conserved region that occurs in a set of genomes where gene order and content is not strictly conserved [3]. A genome, or a subset of it, is represented as a string of integers. Models of gene clusters were thoroughly defined in [2], [3], and [8]. r-windows and max-gap clusters were introduced in [3]. Combinatorial models were discussed in [2]: common interval versus max-gap in permutations versus sequences. Conserved segments was also introduced. In [8], the notion of nested intervals was discussed.

E. lp_solve

lp_solve, a free library written in ANSI C, is an open source (mixed-integer) linear programming (MILP) system. There are four versions available for use; the most recent was version 5.5 that was released in 2005. It solves linear programming problems using two algorithms: revised simplex method and the branch-and-bound algorithm. lp_solve can be called as a library in different languages such as C, VB, .NET, Java, etc. Similarly, the library can be called from AMPL, MATLAB, Octave, R, etc. The library can handle integer variables, semi-continuous variables, and special ordered sets by using the branch-and-bound algorithm.

The Revised Simplex Method expresses linear programs as matrices. The revised simplex method is preferable than its original, the simplex method, since it works faster on large problems [22]. On the other hand, the Branch-and-Bound algorithm is for solving NP-hard combinatorial optimization problems. It looks for the best solution by looking at all potential solutions.

F. *Approximate Gene Cluster Discovery Problem (AGCDP)*

The approximate gene cluster discovery problem is the problem of finding genes that are kept together in different species. The advantage of using the formulation presented in this paper is its independence of hard constraints. Given

- gene universe $U = \{1, 2, ..., N\}$
- m genomes $(g^i)_{i=1,...m}$ where $g^i = \{g^i_1, ..., g^i_{n_i}\}$
- integer weights w^+ and w^- which correspond to the cost of additional gene and missing gene, respectively
- a size range $[D^-, D^+]$ for the reference gene set

find $X \subset U$ such that $0 \notin X$ and $D^- \leq |X| \leq D^+$ and a linear interval J^i for each genome to minimize the objective function

$$c := c(X, (J_i)) = \sum_{i=1}^{m} [w^- \cdot |X \setminus G^i_{J_i}| + w^+ \cdot |G^i_{J_i} \setminus X|] \quad (1)$$

where c is the cost function, $|X \setminus G^i_{J_i}|$ is the number of X-genes not found in the interval or "missing genes", and $|G^i_{J_i} \setminus X|$ is the number of genes found in the interval that are not in X or "additional genes".

The modeling approach does not give a closed definition on what is contained in the set of all approximate gene clusters. It is not interested in all approximate gene clusters but of the best clusters in terms of the objective function. It is, thus, sure to provide a solution unless the objective function is constrained [12].

G. *Integer Linear Programming Formulation*

An overview of the variables used in the ILP formulation [12] is provided in below. All variables are binary.

Main Objects	ILP Variables (Binary)
Reference gene set X	$x = (x_q)_{q=0,...,N}$
Interval J_i in ith genome	$z^i = (z^i_j)_{j=1,...,n_i}, i = 1, ..., m$
Gene content $G^i_{J_i} of J_i$ in g^i	$X^i = (X^i_q)_{q=0,...,N}, i = 1, ..., m$

Auxiliary Objects	ILP Variables (Binary)
Increments is z^i	$^+z^i = (^+z^i_j)_{j=1,...,n_i}, i = 1, ..., m$
Decrements in z^i	$^-z^i = (^-z^i_j)_{j=1,...,n_i}, i = 1, ..., m$
Intersection $X \cap G^i_{J_i}$	$\iota^i = (\iota^i_q)_{q=0,...,N}, i = 1, ..., m$

Target Quantities	ILP Expression
#{Missing genes in g^i}: $\lvert X \setminus G^i_{J_i}\rvert$	$\sum_{q=0}^{N} x_q - \iota^i_q$
#{Additional genes in g^i}: $\lvert G^i_{J_i} \setminus X\rvert$	$\sum_{q=0}^{N} X_q - \iota^i_q$

TABLE I
OVERVIEW OF VARIABLES AND EXPRESSIONS REPRESENTING OBJECTS AND QUANTITIES IN THE BASIC ILP FORMULATION

1) Modeling the Reference Gene Set X: The reference gene set X is represented as a binary vector $x = (x_0, ..., x_N) \in \{0, 1\}^{N+1}$, where we set $x_q = 1$ if and only if $q \in X$. Additional conditions include

$$x_o = 0 \text{ and } D^- \leq \sum_q x_q \leq D^+.$$

2) Modeling the Intervals J_i: Binary indicator vectors $z^i = (z^i_j)_j = 1, ..., n_i$ are used to model the selected interval J_i in the genome i. A linear interval in genome i is characterized by z^i occurring consecutively. This property is enforced by introducing auxiliary binary vectors $^+z^i = (^+z^i_1, ..., ^+z^i_{n_i})$ and $^-z^i = (^-z^i_1, ..., ^-z^i_{n_i})$ which increments and decrements, respectively in z^i.

Therefore, $z^i_1 =^+ z^i_1 -^- z^i_1$, and for $2 \leq j \leq n_i : z^i_j = z^i_{j-1} +^+ z^i_j -^- z^i_j$. Simultaneous increment and decrement at each position is not allowed: $^+z^i_j +^- z^i_j \leq 1$ for all $j = 1, ..., n_i$. At most one increment and decrement is allowed: $\sum_{j=1}^{n_i} {}^+z^i_j \leq 1$ and $\sum_{j=1}^{n_i} {}^-z^i_j \leq 1$.

All three vectors z^i, $^+z^i$ and $^-z^i$ are elements of $\{0, 1\}^{n_i}$. An interval $[j : k]$ with $1 \leq j \leq n_i$ can be represented in a unique way by setting $^+z^i_j = 1$ and $^-z^i_{k+1} = 1$. If $k = n_i$, then ^-z is the zero vector.

3) Modeling the Intervals' Gene Contents $G^i_{J_i}$: The gene content $G^i_{J_i}$ in genome i, is modeled by another indicator vector $X^i = (X^i_q)_{q=0,...,N}$: if some position j is covered by the chosen interval J_i, the corresponding gene must be included in the gene content; thus $X^i_{g^i_j}$ for all $j = 1, ..., n_i$. On the other hand, if some gene $q \in 1, ..., N$ is not covered by J_i, it must not be included: $X^i_q \leq \sum_{j:g^i_j=q} z^i_j$ for all $q \in 0, ..., N$. If gene q is not in genome i, the sum inequality is $X^i_q = 0$.

4) Modeling the Target Function: The intersection between the reference gene set X and the selected gene content $G^i_{J_i}$ in the ith genome. A family of indicator vectors for $i = 1, ..., m$: $\iota^i = (\iota^i_q)_{q=0,...,N}$ is introduced to model the set intersection $X \cap G^i_{J_i}$ via the inequalities $\iota^i_q \leq x_q$, $\iota^i_q \leq X^i_q$, and $\iota^i_q \geq x_q + X^i_q - 1$. Therefore, the target function is composed of

$$|X \setminus G^i_{J_i}| = \sum_{q=0}^{N} (x_q - \iota^i_q); \qquad |G^i_{J_i} \setminus X| = \sum_{q=0}^{N} (X^i_q - \iota^i_q)$$

where $|X \setminus G^i_{J_i}|$ is the number of "missing genes" and $|G^i_{J_i} \setminus X|$ is the number of "additional genes".

The whole ILP formulation for (AGCDP) is provided in [12].

III. Design and Implementation

A. *Data Specifications*

To find approximate gene clusters, the researcher shall input a file. The format of the file should strictly comply with the following:

- The input file is in csv format.
- Each row in the data input shall correspond to one genome.
- The first column of each row is the scientific name of the organism.
- Subsequent columns after the first column must contain the genes of the species represented by its symbol.

- Genes that are unidentified or missing at a certain position must be represented as the number 0.
- The homologous genes in the data file are already represented with the same names.
- The ordering of genes in the input file is the same to the ordering of genes in the genome.

B. System Design

The succeeding illustrations show how the system is implemented. The input and output requirements of the application is defined in Figure 1. The researcher inputs genomic data alongside the parameters needed for solving the ILP problem. In return, the tool produces approximate gene clusters as results.

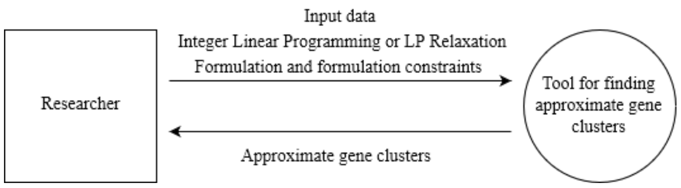

Input data
Integer Linear Programming or LP Relaxation
Formulation and formulation constraints
Researcher
Tool for finding approximate gene clusters
Approximate gene clusters

Fig. 1. Context Diagram

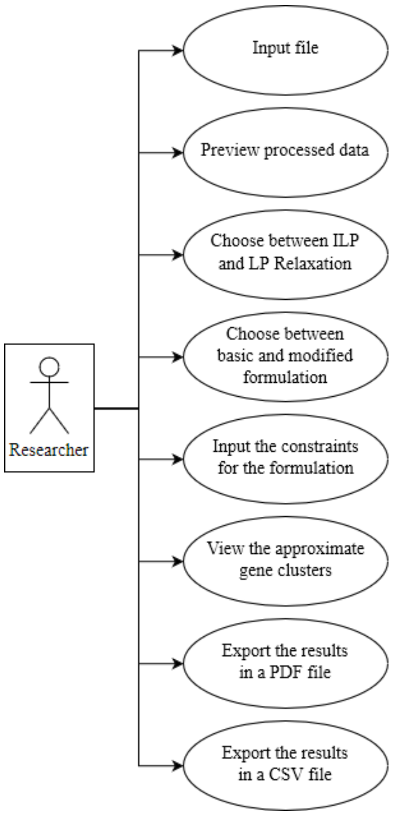

Researcher
Input file
Preview processed data
Choose between ILP and LP Relaxation
Choose between basic and modified formulation
Input the constraints for the formulation
View the approximate gene clusters
Export the results in a PDF file
Export the results in a CSV file

Fig. 2. Flowchart Diagram - Researcher

The tool has only one user, the researcher. The use case diagram for the researcher is provided in Figure 2.

The researcher can input a genome file in CSV format. The researcher may browse through the transformed data i.e. the data after each gene is represented as an integer. The researcher can either use Integer Linear Programming or LP Relaxation. The researcher can also enter the constraints for solving the problem. After running the program, the researcher can view the approximate gene clusters discovered. The researcher can also check out the tool manual provided.

The flowchart diagram of the system is provided in Figure 3. The first step in using the tool is to input a file in csv format which contains the genomic data. The system transforms the genomic data by representing genes as integers and allows the user to view the processed data. The user is then asked to choose between using Integer Linear Programming and LP Relaxation. The user is then asked if the formulation to be used is basic or modified. After selecting which formulation to use, the researcher is prompted to input the variables needed for the formulation. The system then computes for approximate gene clusters and displays the results. The user may also export the results in a file in csv format.

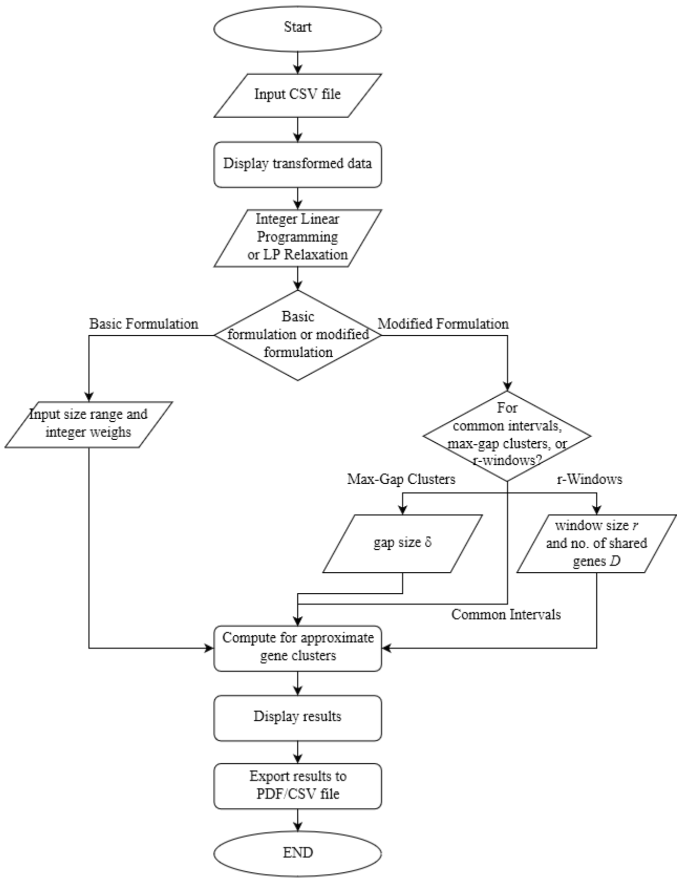

Start
Input CSV file
Display transformed data
Integer Linear Programming or LP Relaxation
Basic formulation or modified formulation
Basic Formulation
Modified Formulation
Input size range and integer weighs
For common intervals, max-gap clusters, or r-windows?
Max-Gap Clusters
r-Windows
gap size δ
window size r and no. of shared genes D
Common Intervals
Compute for approximate gene clusters
Display results
Export results to PDF/CSV file
END

Fig. 3. Flowchart Diagram - Researcher

C. System Architecture

The software uses R and the library lp_solve to minimize the integer linear programming (ILP) function and integrates backend processes to Java which handles other functionalities and the user interface. The user can run the program without an installed Java Runtime Environment (JRE) and can run on any operating system.

D. Technical Architecture

Specifications required for the software to run smoothly include: 4 GB RAM and processor speed of 2.40 GHz.

IV. System Interface Navigation

The main window is shown once the application is opened. The user can either start finding gene clusters, read the user manuals, or learn more what the application is about.

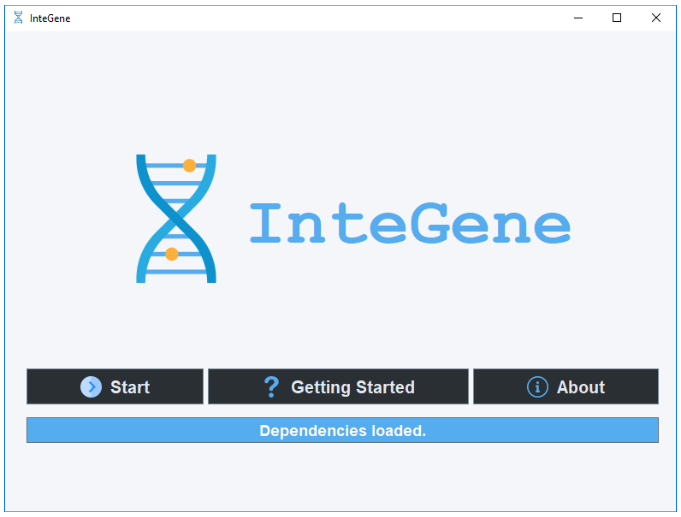
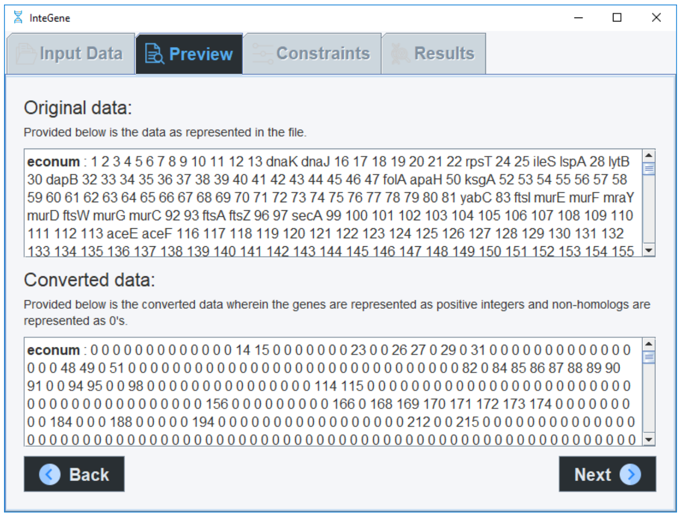


Fig. 4. Home Screen, Approximate Gene Cluster Discovery Tool

First, the user must be able to provide input data. Once the user has provided the data, the user will be able to see the contents of the data. After providing the input data, the application will now convert the genomes in the data to integers. Once this process is finished, the user will now be able to proceed in the Preview tab seen in Figure 5. There the user will be able to see the converted data where the non-homologs in the original data as seen in the first text box are converted to zero as seen on the second text box.

After viewing the transformed data, the user can proceed to the Constraints tab. The default formulation is general. In here, the user is given the power to change the formulation, size range, and additional and missing gene weights. The gap size can only be used if the formulation chosen is for finding max-gap clusters. Similarly, the k-size can only be used if the formulation chosen is for finding r-Window clusters. After changing the constraints as desired by the user, the user can now proceed to finding the best clusters given the constraints. Solving for approximate gene clusters may take a few minutes especially if the data provided is big. The view while this process is happening can be seen in Figure 6.

After the application has computed for the best clusters, the user will be able to see the results in the Results tab as seen in Figure 7. From here, the user can export the results in CSV format or in PDF format.



Original data:

Provided below is the data as represented in the file.

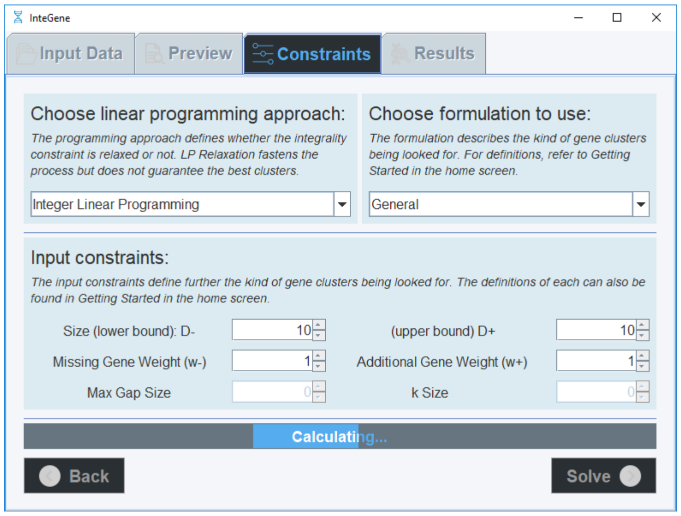
```
econum : 1 2 3 4 5 6 7 8 9 10 11 12 13 dnaK dnaJ 16 17 18 19 20 21 22 rpsT 24 25 ileS lspA 28 lytB
30 dapB 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 folA apaH 50 ksgA 52 53 54 55 56 57 58
59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 yabC 83 ftsI murE murF mraY
murD ftsW murG murC 92 93 ftsA ftsZ 96 97 secA 99 100 101 102 103 104 105 106 107 108 109 110
111 112 113 aceE aceF 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132
133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155
```

Converted data:

Provided below is the converted data wherein the genes are represented as positive integers and non-homologs are represented as 0's.

```
econum : 0 0 0 0 0 0 0 0 0 0 0 0 0 14 15 0 0 0 0 0 0 0 23 0 0 26 27 0 29 0 31 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 48 49 0 51 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 82 0 84 85 86 87 88 89 90
91 0 0 94 95 0 0 98 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 114 115 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 156 0 0 0 0 0 0 0 0 0 166 0 168 169 170 171 172 173 174 0 0 0 0 0 0 0
0 0 184 0 0 0 188 0 0 0 0 0 194 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 212 0 0 215 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Fig. 5. Preview, InteGene



Choose linear programming approach:

The programming approach defines whether the integrality constraint is relaxed or not. LP Relaxation fastens the process but does not guarantee the best clusters.

Integer Linear Programming

Choose formulation to use:

The formulation describes the kind of gene clusters being looked for. For definitions, refer to Getting Started in the home screen.

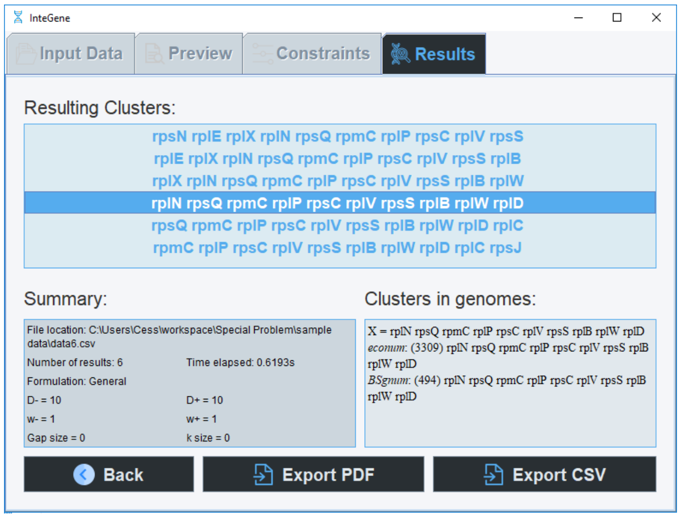
General

Input constraints:

The input constraints define further the kind of gene clusters being looked for. The definitions of each can also be found in Getting Started in the home screen.

Size (lower bound): D-	10	(upper bound) D+	10
Missing Gene Weight (w-)	1	Additional Gene Weight (w+)	1
Max Gap Size	0	k Size	0

Fig. 6. Constraints, InteGene



Resulting Clusters:

rpsN rplE rplX rplN rpsQ rpmC rplP rpsC rplV rpsS
rplE rplX rplN rpsQ rpmC rplP rpsC rplV rpsS rplB
rplX rplN rpsQ rpmC rplP rpsC rplV rpsS rplB rplW
rplN rpsQ rpmC rplP rpsC rplV rpsS rplB rplW rplD
rpsQ rpmC rplP rpsC rplV rpsS rplB rplW rplD rplC
rpmC rplP rpsC rplV rpsS rplB rplW rplD rplC rpsJ

Summary:

File location: C:\Users\Cess\workspace\Special Problem\sample data\data6.csv

Number of results: 6	Time elapsed: 0.6193s
Formulation: General	
D- = 10	D+ = 10
w- = 1	w+ = 1
Gap size = 0	k size = 0

Clusters in genomes:

X = rplN rpsQ rpmC rplP rpsC rplV rpsS rplB rplW rplD
econum: (3309) rplN rpsQ rpmC rplP rpsC rplV rpsS rplB rplW rplD
BSgnum: (494) rplN rpsQ rpmC rplP rpsC rplV rpsS rplB rplW rplD

Fig. 7. Results, InteGene

V. Results and Discussion

InteGene is a tool for finding approximate gene clusters which uses an integer linear programming approach. The application provides biologists and researchers in the field of genomics a platform in which they can use for discovering gene clusters that may have significance. Using available data, the user or the researcher can define the constraints which define the cluster the user is interested to find. However, since the results returned by the system are those groups of genes that are merely relatively near each other across genomes, it is not certain that these resulting clusters really have biological significance. This is outside the scope of the tool.

InteGene uses ILP that is proven to be NP-complete. Other algorithms developed such as [14] for finding common intervals in k permutations that run in $O(kn + K)$. An algorithm of time complexity $O(n^2log(n))$ is also presented in [13] for finding common intervals given 2 sequences where n is the length of the longer sequence. Two algorithms for finding max-gap clusters are also presented in [15]. Blin et. al presented three efficient algorithms for finding nested intervals, if present, that run in cubic time, quadratic time, and linear time [8]. The advantage of using ILP, despite being NP-complete is that it does not restrict the results to one model of gene cluster such as presented in the other studies. Furthermore, the ILP approach can be modified to cater other gene models. However, other algorithms for finding specific models that are more efficient are more preferable.

There is limited available data online that represents genes as numbers since homologous genes in different species are represented differently. However, there are already different studies such as [12] that used genome data represented as integers which means there are already available data in this form. Two sets of of data were used to test the validity and correctness of the application.

The data is adapted from [4] and [5]. The dataset provided in [4] provides a list of identified orthologous genes between the species *Escherichia coli* and 36 other species. The dataset found in [5] provided a list of orthologous genes found in every species and their corresponding gene number. The gene number is based on the chromosal position of the gene. The data used in [4] and [5] can be accessed in http://www.cs.kent.edu/ arvind/intellibio/orthos.html and https://academic.oup.com/mbe/article/22/6/1456/1111802, respectively.

The data provided in [4] is structured as seen in Table V but only columns 2 and 3 will be necessary and other columns are disregarded. The table contains all the orthologous genes found between genome$_1$ and genome$_2$. Note that genome$_1$ will always correspond to *Escherichia coli*. The values in columns 2 and 3 are gene symbols with their corresponding gene position i.e. in row 1, column 2, the gene symbol is thrA and the gene position is 2.

To fit the required data format for the application, the data had to be modified by inserting $0's$ to positions with no available homologs. It must also be converted into the required format for the application.

TABLE II
Structure of data from [4]

Escherichia coli vs. genome$_2$					
No.	Gene$_1$	Gene$_2$	Strands	Location$_1$	Location$_2$
1.	thrA.2	hom.3221	+:-	337..2799	3313886..3315187

Escherichia coli vs. genome$_2$

Length$_1$	Length$_2$	Align$_1$	Align$_2$	Enzyme$_1$:Enzyme$_2$
820	433	550:704	75:228	"2.7.2.4":"1.1.1.3"

Escherichia coli vs. genome$_2$

Aligned Length	Identity	Similarity	Maximum continuous
58	29	97	3

The dataset used in [5] is structured as seen in Table V where the first column *name* contains the common names of the genes in *E. coli* that have orthologs in the subsequent columns, the second column contains the gene numbers of E. coli, and the subsequent columns contains the gene numbers of each genome. There are a total of 30 genomes.

Similarly, the only relevant data is the name of the orthologous gene and its gene number or gene position. Each unique ortholog is represented as x where $x \in \mathbb{Z}_+$. It follows that orthologous genes are represented by the same positive integer x. After representing each gene into a positive integer, sequences of genomes can be derived. Provided the gene number of each gene in a genome, the genome sequence can be derived by using the integer representation of each gene and all genes that are not represented will be assigned the integer 0.

Also, the data was processed in order to fit the data specifications of the application. Zero's were inserted to positions with no homologous gene in the data. This must be done since non-homologous genes contribute to the validity of a gene cluster.

In a study by Bansal and Woolverton [23], the authors used the same data to predict metabolic pathways from identified gene clusters in the data. However, they have refined the data so there were newly introduced clusters that were not in the original data. The identified pathways from gene clusters in the study is provided in Table V. InteGene was able to produce the same gene clusters provided the correct constraints. The results and the corresponding constraints supplied is provided in Table V. The time for computation is provided but this may vary depending on the machine specifications.

It was observed that using very small sizes for finding gene

TABLE III
Structure of data from [5]

name	econum	BSgnum	...	vpanum
aceE	b0114	BUsg199	...	VP2519

TABLE IV
PATHWAY SEEDS USING *E. coli* VERSUS *B. subtilis*

Pathway	Genes
Citrate Cycle	(sdhA, sdhB), (sucA, sucB), (sucC, sucD)
Oxidative phosphorylation	(sdhA, sdhB)
Butanoate metabolism	(sdhA, sdhB)
Propanoate metabolism	(sucC, sucD)
Lysine degradation	(sucA, sucB)
Urea cycle and metabolism of amino groups	(proB, proA), (argC, argB)
Reductive carboxylate cycle	(sdhA, sdhB, sucC, sucD)
Purine metabolism	(guaA, guaB, purM, purN), (prsA, tchB, purD, purH)
Sulfur metabolism	(cysC, cysH)
Peptideglycan biosynthesis	(yabB, yabC, ftsl, murE, murD, murF, mraY, ftsW, murG, murC)
Flagellar assembly	(fliE, fliF, fliG, fliI, flil, fliM, fliN, flip, fliQ, fliR)
Pentose and glucuronate Interconversions	(araD, araA, araB), (lyxk, yiaS), (orf, ygcE)
Histidine metabolism	(hisG, hisD, hisB, hisH, hisA, hisF, hisl)
Valine, leucine, and isoleucine biosynthesis	(leuC, leuB, leuA)
Fatty acid biosynthesis	(folP, hflB)
Terpenoid biosynthesis	(plsX, fabD, fabG, acpP)
Alanine and aspartate metabolism	(orf, ispA, xseB)
Cs-branched dibasic acid metabolism	(argH, oxrR)
Phospholipid degradation	(glpQ, glpT)

clusters takes much longer time since it produces more number of intervals for finding the best gene cluster. In contrast, using larger sizes for finding gene clusters takes much less time since there are fewer intervals to check. Thus, using larger ranges takes much more time since intervals with sizes in the interval are generated, the number of which is much larger than when considering one value for the range.

The data provided in [5] was also used to test the tool. This contains 30 genomes where only genes shared among the genomes are present in the data. Experimental results are provided in Table V. It can be seen that the difference of choosing between LP Relaxation and Integer Linear Programming is not that significant as it only takes only a couple of seconds.

VI. CONCLUSION

The software produced in this study uses an Integer Linear Programming formulation adapted from [12] and datasets used for testing from [4] and [5]. The application's performance in terms of time depends on the size of the input data and the constraints specified by the user.

To prove the validity of the application, the results of the study in [23] where the results were generated using the tool. Some of the clusters identified in the study were generated using the application, however, some were not found due to modification of the data in the said study.

The performance of using Integer Linear Programming versus LP Relaxation was also demonstrated using the dataset

TABLE V
GENE CLUSTERS GENERATED BY INTEGENE AND CORRESPONDING INPUT CONSTRAINTS

Genes	Formulation	D-	D+	w-	w+	g	k	Time Elapsed
(sucA, sucB)	General	2	2	1	1	-	-	1973.1019s
(sucC, sucD)	General	2	2	1	1	-	-	1973.1019s
(proB, proA	General	2	2	1	1	-	-	1973.1019s
(yabB, yabC, ftsl, murE, murD, murF, mraY, ftsW, murG, murC)	Max-gap	10	11	0	0	3	-	
(fliE, fliF, fliG, fliI, flil, fliM, fliN, flip, fliQ, fliR)	Max-gap	10	18	0	0	8	-	$> 7200s$
(araD, araA, araB/lyxK)	Max-gap	3	3	0	0	1	-	$> 3600s$
(orf, ygcE)	General	2	2	1	1	-	-	1973.1019s
(hisG, hisD, hisB, hisH, hisA, hisF, hisl)	Max-gap	7	8	0	0	1	-	98.2902s
(leuC, leuB, leuA)	General	3	3	1	1	-	-	504.2793s
(plsX, fabD, fabG, acpP)*	General	3	3	1	1	-	-	504.2793s
(orf, ispA, xseB)	General	3	3	1	1	-	-	504.2793s
(glpQ, glpT)	General	2	2	1	1	-	-	1973.1019s

*The gene plsX does not exist in the dataset, however, {fabD, fabG, acpP} were discovered by the tool.

TABLE VI
SUMMARY OF RESULTS AND THE CORRESPONDING INPUT CONSTRAINTS USING THE DATASET FROM [5]

Formulation	D-/D+	w-/w+	Cost	Number of Results	Time Elapsed (ILP)	Time Elapsed (LP Relaxation)
General	5	1	8	4	581.8555s	599.7276s
General	6	1	8	2	325.5048s	380.4599s
General	7	1	16	10	226.1462s	221.3463s
General	8	1	18	6	144.7781s	142.3204s

from [5]. The standalone software produced in the study provides biologists and researchers of the same field a tool that finds gene clusters from different genomes given a set of constraints.

VII. Recommendations

One improvement that can be done is to allow homologous genes from different species to be accepted by the system even if they are not represented in the same manner. This will lessen the user's effort in preprocessing data since there are online repositories where whole sequences of genomes can be downloaded. Instead, representation of the same symbol shall be done internally.

Acknowledgment

G. Solano is supported by the Engineering Research and Development for Technology (ERDT) Scholarship Program of the Department of Science and Technology(DOST) and the Doctoral Sandwich Scholarship Program of the Commission on Higher Education (CHEd) Philippines.

References

[1] J. A. Aborot, H. Adorna, J. B. Clemente, B. K. de Jesus and G. Solano. Search for a Star: Approximate Gene Cluster Discovery Problem (AGCDP) as a Graph Problem. Philippine Computing Journal, vol.7 no.2 (2012)

[2] A. Bergeron, Y. Gingras, and C. Chauve, Formal models of gene clusters, in Bioinformatics Algorithms: Techniques and Applications (I. I. Mndoiu and A. Zelikovsky, eds.), ch. 8, pp. 177-202, Hoboken, New Jersey, United States: John Wiley Sons, Inc., 2008.

[3] R. Hoberman and D. Durand, The incompatible desiderata of gene cluster properties, Lecture Notes in Bioinformatics, vol. 3678, pp. 77-87, 2005.

[4] A. K. Bansal, An automated comparative analysis of 17 compelete microbial genomes, Bioinformatics, vol. 15, pp. 900-908, 1999.

[5] E. Belda, A. Moya, and F. J. Silva, Genome rearrangement distances and gene order phylogeny in gamma-proteobacteria, Molecular biology and evolution, vol. 22, 2005.

[6] D. Durand and D. Sankoff, Tests for gene clustering, Journal of Computational Biology: A Journal of Computational Molecular Cell Biology, vol. 10, 2003.

[7] S. Bocker, K. Jahn, J. Mixtacki, and J. Stoye, Computation of median gene clusters, Journal of Computational Biology, vol. 16, no. 8, pp. 10851099, 2009.

[8] G. Blin, D. Faye, and J. Stoye, Finding nested common intervals efficiently, Journal of Computational Biology, vol. 17, no. 9, pp. 1183-1194, 2010.

[9] J. Alfoldi and K. Lindblad-Toh, Comparative genomics as a tool to understand evolution and disease, Genome Res, vol. 23, pp. 1063-1068, Jul 2013. 23817047[pmid].

[10] M. Semon and L. Duret, Evolutionary origin and maintenance of coexpressed gene clusters in mammals, Molecular Biology and Evolution, vol. 23, no. 9, pp. 1715-1723, 2006.

[11] B. Snel, P. Bork, and M. A. Huynen, The identification of functional modules from the genomic association of genes, Proceedings of the National Academy of Sciences, vol. 99, no. 9, pp. 5890-5895, 2002.

[12] S. Rahmann and G. W. Klau, Integer linear programming techniques for discovering approximate gene clusters, in Bioinformatics Algorithms: Techniques and Applications (I. I. Mndoiu and A. Zelikovsky, eds.), ch. 9, pp. 203-222, Hoboken, New Jersey, United States: John Wiley Sons, Inc., 2008.

[13] G. Didier, Common intervals of two sequences, in WABI, pp. 1724, Springer, 2003.

[14] T. Uno and M. Yagiura, Fast algorithms to enumerate all common intervals of two permutations, Algorithmica, vol. 26, no. 2, pp. 290-309, 2000.

[15] A. Bergeron, S. Corteel, and M. Raffinot, The algorithmic of gene teams, in International Workshop on Algorithms in Bioinformatics, pp. 464-476, Springer, 2002.

[16] G. S. Cabunducan, J. B. Clemente, R. T. Relator, and H. N. Adorna, Approximate gene cluster discovery problem (agcdp) is np-hard, 2011.

[17] E. Fox Keller and D. Harel, Beyond the gene, PLoS One, vol. 2, p. e1231, Nov 2007.

[18] H. M. Wain, E. A. Bruford, R. C. Lovering, M. J. Lush, M. W. Wright, and S. Povey, Guidelines for human gene nomenclature.

[19] D. W. Mount, Bioinformatics Sequence and Genome Analysis. Cold Spring Harbor Laboratory Press, 2004.

[20] A. Griffiths, W. Gelbart, and J. Miller, Modern Genetic Analysis. New York: W. H. Freeman, 1999.

[21] R. J. Reece, Analysis of Genes and Genomes. John Wiley Sons, Ltd, 2004.

[22] S. S. Morgan, A comparison of simplex method algorithms, 1997.

[23] A. K. Bansal and C. J. Woolverton, Applying automatically derived gene-groups to automatically predict and refine metabolic pathways, IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 4, pp. 883-894, 2003.