Biblioteca Digital
F C E N - U B A

BIBLIOTECA CENTRAL LUIS F LELOIR
BIBLIOTECA CENTRAL LELOIR
FACULTAD DE CIENCIAS EXACTAS Y NATURALES UBA

Tesis de Maestría

Búsqueda de mecanismos regulatorios en redes biológicas usando multigrados en base de datos no relacionadas

Cardona Velásquez, Crhistian

2016-08-18

Este documento forma parte de la colección de tesis doctorales y de maestría de la Biblioteca Central Dr. Luis Federico Leloir, disponible en digital.bl.fcen.uba.ar. Su utilización debe ser acompañada por la cita bibliográfica con reconocimiento de la fuente.

This document is part of the doctoral theses collection of the Central Library Dr. Luis Federico Leloir, available in digital.bl.fcen.uba.ar. It should be used accompanied by the corresponding citation acknowledging the source.

Cita tipo APA:
Cardona Velásquez, Crhistian. (2016-08-18). Búsqueda de mecanismos regulatorios en redes biológicas usando multigrados en base de datos no relacionadas. Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires.

Cita tipo Chicago:
Cardona Velásquez, Crhistian. "Búsqueda de mecanismos regulatorios en redes biológicas usando multigrados en base de datos no relacionadas". Facultad de Ciencias Exactas y Naturales. Universidad de Buenos Aires. 2016-08-18.

EXACTAS UBA
Facultad de Ciencias Exactas y Naturales

UBA
Universidad de Buenos Aires

Dirección: Biblioteca Central Dr. Luis F. Leloir, Facultad de Ciencias Exactas y Naturales, Universidad de Buenos Aires. Intendente Güiraldes 2160 - C1428EGA - Tel. (++54 +11) 4789-9293

Contacto: digital@bl.fcen.uba.ar

UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

BÚSQUEDA DE MECANISMOS REGULATORIOS EN REDES BIOLOGICAS USANDO MULTIGRADOS EN BASE DE DATOS NO RELACIONADAS

Tesis presentada para obtener el título de
Magíster en exploración de datos y descubrimiento de conocimiento

Estudiante:
Crhistian Cardona Velásquez

Director de tesis
Dr. Marcelo Soria

Buenos Aires, Argentina, 2016

TABLA DE CONTENIDO

Pág.

RESUMEN 10
ABSTRACT 11
1. INTRODUCCION 12
1.1 MOTIVACIÓN 12
1.2 GRAFOS Y MULTIGRAFOS 13
1.2.1 Grafos 14
1.2.2 Grafos dirigidos, no dirigidos, mixtos y multigrafos. 14
1.2.3 Social Network Analisys (SNA) 16
1.2.3.1 Propiedades globales de grafos complejos 17
1.2.3.2 Medidas de Centralidad 18
1.2.3.3 Cliques 19
1.2.3.4 Cliques débiles o quasi-cliques 21
1.2.4 El concepto de NoSQL o No relacional. 22
1.2.5 Grafos en bases de datos 27
1.2.6 Bases de datos orientadas a grafos 30
1.2.6.1 NEO4J 34
1.2.6.2 Titán DB 37
1.2.7 RDBMS Versus bases de datos de grafos 40
1.2.8 Lenguajes de consultas orientados a grafos 40
1.2.8.1 SPARQL y SPASQL 40
1.2.8.2 CYPHER 41
1.2.8.3 Gremlin 42
1.3 BIOINFORMÁTICA Y *Mycobacterium tuberculosis* 42
1.3.1 Mycobacterium tuberculosis 42
1.3.2 ANÁLISIS BIOLÓGICOS 44
1.3.2.1 Interacción proteína-proteína 44
1.3.2.2 Análisis de las interacciones entre regiones promotoras y proteínas 44
1.3.2.3 Co-expresión génica 45
1.3.2.3.1 Matrices de adyacencia 45
1.3.2.3.2 Método C3NET 46
1.3.2.3.3 Coeficiente de variación 46
1.4 BASES DE DATOS BIOLÓGICAS PARA *M. tuberculosis* 46
1.4.1 Tuberculist 46
1.4.2 TBDB 47
1.5 PLATAFORMA DE INTEGRACIÓN SEMÁNTICA DE DATOS BIOLÓGICOS:

BIO4J 47
2. OBJETIVOS 50
3. HIPÓTESIS GENERAL 50
4. ORGANIZACIÓN DE LA TESIS 50
5. MATERIALES Y MÉTODOS 51
5.1 HERRAMIENTAS 51
5.2 LIBRERÍAS 51
5.3 SERVICIOS 52
5.4 DESCRIPCIÓN DE LOS DATOS BASE 52
5.5 CONJUNTOS DE DATOS PARA GRAFOS INDIVIDUALES 58
5.5.1 Composición de datos de los nodos 58
5.5.2 Composición de datos de las relaciones 59
5.5.2.1 Grafo no dirigido 1 63
5.5.2.2 Grafo dirigido 2 69
5.5.2.3 Grafo no dirigido 3 73
5.5.2.4 Grafo no dirigido 4 77
5.5.2.5 Grafo no dirigido 5 82
5.6 MULTIGRAFO: CONSTRUCCIÓN Y ANÁLISIS 87
5.6.1 Construcción de multigrafo 87
5.6.2 Composición de datos de los nodos 88
5.6.3 Composición de datos de las relaciones 88
5.6.4 Conjunto de datos 89
5.6.5 SNA (*Social Network Analysis)* 92
5.7 IMPORTACIÓN DE DATOS A BASES DE DATOS DE GRAFOS Y EXPORTACIÓN DE GRAFOS A FORMATO GRAPHML 96
5.7.1 Importación a NEO4J 96
5.7.1.1 Consideraciones adicionales en NEO4J 97
5.7.2 Exportación de grafos a formato graphML en NEO4J 98
5.7.3 Importación en TITAN DB 98
5.7.3.1 Consideraciones adicionales en TITAN 98
5.7.4 Exportación de grafos a formato graphML en TITAN DB 99
6. ANÁLISIS DE LOS RESULTADOS 101
6.1 QUASI-CLIQUES EN MULTIGRAFOS 101
6.2 PATRONES BIOLÓGICOS EN QUASI-CLIQUES DE MULTIGRAFOS 103
6.2.1 Prototipo de consulta 1 – Relaciones consecutivas específicas. 104
6.2.2 Prototipo de consulta 2: Relaciones consecutivas y maximización de métricas conjuntas 107
6.2.3 Prototipo de consulta 3 - Relaciones consecutivas y proteínas o genes en identificadores. 113

6.2.4 Prototipo de consulta 4 - Relaciones consecutivas y proteínas o genes en nombres y anotaciones. 116
6.2.5 Prototipo de consulta 5 - Número de relaciones de tipo distinto. 125
6.2.6 Prototipo de consulta 6 - Relaciones de conjuntos de datos distintos. 128
6.2.7 Prototipo de consulta 7 - Relaciones distintas consecutivas. 130
6.3 ANÁLISIS BIOLÓGICO DE RESULTADOS 133
6.4 RESUMEN GENERAL DE RESULTADOS EN LOS PROTOTIPOS DE CONSULTAS 135
CONCLUSIONES 138
TRABAJO FUTURO 139
BIBLIOGRAFIA 140
ANEXOS 144
ANEXO 1: Scripts de importanción de multigrafo a Titan DB 144

INDICE DE TABLAS

Pág.

Tabla 1: Listado de principales productos relacionados con bases de datos de grafos en db-engine ranking [33]. 29
Tabla 2: Listado de las bases de datos de grafos listadas en db-engine [33]. 30
Tabla 3: Datos resumidos de conjuntos de datos seleccionados para experimentación... 53
Tabla 4: Detalles de conjunto de datos 1 de tipo PPI. 54
Tabla 5: Detalles de conjunto de datos 2 de tipo regulación de la expresión de genes a nivel transcripcional. 54
Tabla 6: Detalles de conjunto de datos 3 de tipo PPI. 55
Tabla 7: Detalles de conjunto de datos 4 de tipo co-expresión génica. 56
Tabla 8: Detalles de conjunto de datos 5 de tipo co-expresión génica. 58
Tabla 9: Correspondencia de atributos en las relaciones según el tipo de datos. 61
Tabla 10: Correspondencia entre conjunto de datos base y conjuntos de datos generados para construir los grafos individuales. 62
Tabla 11: Listado de las 10 proteínas con máximo grado en multigrafo. 92
Tabla 12: Resumen de datos generales y medidas SNA de grafos 1-5 y multigrafo 100
Tabla 13: Factores variables de los 7 prototipos de consultas planteadas. 103
Tabla 14: Información de nodos y relaciones de los 3 quasi-cliques tomados del prototipo de consultas 3. 116
Tabla 15: Listado de genes y proteínas relevantes por cada trabajo relacionado con los datos que componen el multigrafo. 117

INDICE DE ILUSTRACIONES

Pág.

Ilustración 1: A la izquierda representación del rio Pregel. A la derecha el grafo correspondiente a la solución al problema de los puentes de Königsberg [1]. 13
Ilustración 2: Grafo G, donde el conjunto de vértices $VG = u, v, w, z$ y donde el conjunto de aristas $EG = uv, vw, wu, wz$ [16]. 14
Ilustración 3: Grafo no dirigido, dirigido y mixto (de izquierda a derecha) [17]. 15
Ilustración 4: Multigrafo y/o pseudografo (de izquierda a derecha) según [18]. 15
Ilustración 5: Contracción de la arista e del multigrafo G en un multígrafo G/e [15]. 16
Ilustración 6: Ejemplo de un multigrafo mixto. (Original tomado de www.wikimedia.com) 16
Ilustración 7: Evolución del análisis de redes sociales [19]. 17
Ilustración 8: k-clique con k=3, k=4 y k=5 (de izquierda a derecha respectivamente) [19]. 20
Ilustración 9: n-clique con n=1, 2, 3 de tamaño k=4 [19]. 20
Ilustración 10: Teorema de **CAP** o teorema de Brewer's con algunos ejemplos de bases de datos que usan dos de sus tres principios en cada intersección (Tomado de http://changeaas.com). 23
Ilustración 11: Ejemplo de estructura de datos de bases de datos de tipo key-value store [26]. 25
Ilustración 12: Ejemplo de estructura de datos de bases de datos de tipo Column Family Store [26]. 26
Ilustración 13: Ejemplo de estructura de datos de bases de datos de tipo Document Store [26]. 27
Ilustración 14: Cuadrante mágico de Gartner para sistemas de gestión de bases de datos 2015 [32]. 28
Ilustración 15: Funcionamiento de un motor de cómputo de grafos [26]. 31
Ilustración 16: Clasificación de bases de datos de grafos basados en su almacenamiento y modelo de procesamiento [38]. 32
Ilustración 17: Estructura de grafos en una base de datos NoSQL orientada a grafos [26]. 33
Ilustración 18: Arquitectura de alta disponibilidad (HA) de Neo4j [26]. 35
Ilustración 19: Diseño de integración con librerías externas para visualización de grafos en NEO4J [26]. 36
Ilustración 20: Vista lógica de la API en Neo4j [38]. 37
Ilustración 21: Tecnologías alrededor del proyecto Thinkaurelius [44]. 37
Ilustración 22: Titan DB y sus posibilidades de consistencia eventual aplicando el teorema CAP gracias al uso de distintas bases de almacenamiento de datos [44]. 38
Ilustración 23: Ecosistema Apache Tinkerpop 3.0 [46]. 39
Ilustración 24: Arquitectura de la base de datos TITAN [44] . 39
Ilustración 25: Conjuntos de datos usados en BIO4J. (Imagen Tomada de http://bio4j.com). 48
Ilustración 26: Modelo de datos de BIO4J. (Imagen Tomada de http://bio4j.com). 49
Ilustración 27: Logaritmo de la frecuencia de repeticiones de relaciones de proteínas en el grafo 1. 63
Ilustración 28: Logaritmo de la frecuencia de repeticiones de las relaciones de proteínas consigo mismo en el grafo 1. 64
Ilustración 29: Grafo 1 con tamaño de nodos y color según el grado. 65
Ilustración 30: Zona ampliada del grafo 1 correspondiente a región demarcada en

Ilustración 29. 65
Ilustración 31: Distribución de grado de grafo 1. 66
Ilustración 32: Distribución de centralidad de intermediación (betweenness centrality) para el grafo 1, usando escala logarítmica en la frecuencia. 66
Ilustración 33: Distribución de centralidad de cercanía (closeness centrality) para el grafo 1, usando escala logarítmica en la frecuencia. 67
Ilustración 34: Distribución de comunidades según su tamaño para el grafo 1, usando escala logarítmica de la frecuencia. 68
Ilustración 35: Número de cliques del Grafo 1 por cada k-clique con k desde 1 hasta 9. 68
Ilustración 36: Logaritmo de tiempo relativo de procesamiento en base al menor tiempo para cálculo de k-cliques. 69
Ilustración 37: Grafo 2 con tamaño de nodos y color según el grado. 70
Ilustración 38: Distribución de grado del grafo 2, usando escala logarítmica para la frecuencia. 71
Ilustración 39: Distribución de centralidad de intermediación (betweenness centrality) para el grafo 2, usando escala logarítmica para la frecuencia. 71
Ilustración 40: Distribución de centralidad de cercanía (closeness centrality) para el grafo 2, usando escala logarítmica para la frecuencia. 72
Ilustración 41: Tamaño de comunidades para el grafo 2. 72
Ilustración 42: Número de cliques del Grafo 2 por cada k-clique con k desde 1 hasta 6 que es el máximo clique. 73
Ilustración 43: Grafo 3 con tamaño de nodos y color según el grado. 74
Ilustración 44: Distribución de grado del grafo 3, usando escala logarítmica para la frecuencia. 75
Ilustración 45: Distribución de centralidad de intermediación (betweenness centrality) para el grafo 3, usando escala logarítmica para la frecuencia. 75
Ilustración 46: Distribución de centralidad de cercanía (closeness centrality) para el grafo 3, usando escala logarítmica para la frecuencia. 76
Ilustración 47: Tamaño de comunidades para el grafo 3. 76
Ilustración 48: Número de cliques del grafo 3 por cada k-clique con k desde 1 hasta 3 que es el máximo clique. 77
Ilustración 49: Distribución de coeficiente de variación, grafo 4. 78
Ilustración 50: Distribución de percentiles de los coeficientes de variación para el grafo 4. 78
Ilustración 51: Grafo 4 con tamaño de nodos y color según el grado. 79
Ilustración 52: Distribución de grado de grafo 4. 80
Ilustración 53: Distribución de centralidad de intermediación (betweenness centrality) para el grafo 4, usando escala logarítmica para la frecuencia. 80
Ilustración 54: Distribución de centralidad de cercanía (closeness centrality) para el grafo 4, usando escala logarítmica para la frecuencia. 81
Ilustración 55: Distribución de tamaño de comunidades para el grafo 4. 81
Ilustración 56: Distribución del logaritmo del coeficiente de variación, grafo 5. 82
Ilustración 57: Distribución de percentiles del logaritmo de los coeficientes de variación para el grafo 5. 83
Ilustración 58: Grafo 5 con tamaño de nodos y color según el grado. 84
Ilustración 59: Distribución de grado de grafo 5. 85
Ilustración 60: Distribución de centralidad de intermediación (betweenness centrality) para el grafo 5, usando escala logarítmica para la frecuencia. 85
Ilustración 61: Distribución de centralidad de cercanía (closeness centrality) para el grafo 5, usando escala logarítmica para la frecuencia. 86
Ilustración 62: Distribución de tamaño de comunidades para el grafo 5. 86
Ilustración 63: Proceso de enriquecimiento de datos para construcción de multigrafo. ... 88

Ilustración 64: Multigrafo con tamaño de nodos y color según el grado en escala de rojos y grafos 1-5 que lo componen con colores diferenciados. 89
Ilustración 65: Multigrafo con componentes de **datos de tipo 1: Proteína-Proteína** con tamaño de nodos y color según el grado en escala de rojos y grafos 1 y 3 que lo componen con colores diferenciados. 90
Ilustración 66: Multigrafo con componentes de **datos de tipo 2: regulación de la expresión de genes** con tamaño de nodos y color según el grado en escala de rojos y grafo 2 que lo compone de color azul. 91
Ilustración 67: Multigrafo con componentes de **datos de tipo 3: Expresión génica** con tamaño de nodos y color según el grado en escala de rojos y grafos 4 y 5 que lo componen con colores diferenciados. 92
Ilustración 68: Distribución de grado del multigrafo, usando escala logarítmica para la frecuencia. 93
Ilustración 69: Distribución de centralidad de intermediación (betweenness centrality) para el multigrafo, usando escala logarítmica para la frecuencia. 94
Ilustración 70: Distribución de centralidad de cercanía (closeness centrality) para el multigrafo, usando escala logarítmica en la frecuencia. 94
Ilustración 71: Distribución del tamaño de comunidades para el multigrafo. 95
Ilustración 72: Número de cliques del multigrafo por cada k-clique con k desde 1 hasta 9. 95
Ilustración 73: Ejemplo de un 0.5- quasi-clique o un 6-clique con k=5 en donde se marcan con líneas de puntos las 5 relaciones faltantes para completar un strong clique o clique completo. 102
Ilustración 74: Cuatro primeros quasi-cliques con k=3 correspondientes al prototipo de consulta 1. 106
Ilustración 75: Prototipo de consulta 2 con 2 quasi-cliques basados en el grado de centralidad conjunto mayor. 109
Ilustración 76: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de centralidad conjunta mayor con quasi-cliques no buscados. 110
Ilustración 77: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de intermediación mayor. 111
Ilustración 78: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de cercanía mayor. 112
Ilustración 79: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de repetición conjunta mayor. 113
Ilustración 80: Prototipo de consulta 3 que muestra 3 de los 11 quasi-cliques encontrados filtrando el top 5 de las proteínas con más grado de multigrafo. 115
Ilustración 81: Prototipo de consulta 4 para el artículo 1, que muestra un quasi-clique encontrado filtrando las proteínas y genes relevantes en el artículo mencionado. 119
Ilustración 82: Prototipo de consulta 4 para el artículo 2, que muestra 4 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo. 121
Ilustración 83: Prototipo de consulta 4 para el artículo 3, que muestra dos quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho el artículo. 122
Ilustración 84: Prototipo de consulta 4 para el artículo 4, que muestra 8 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo. 124
Ilustración 85: Prototipo de consulta 4 para el artículo 4, que muestra 3 de los 8 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo. 125
Ilustración 86: Prototipo de consulta 5 para el artículo 5, que muestra 3 de los 126 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho el artículo. 128
Ilustración 87: Prototipo de consulta 6, que muestra 4 quasi-cliques encontrados. 130
Ilustración 88: Prototipo de consulta 7, que muestra dos quasi-cliques con k=5. 133
Ilustración 89: Quasi-clique enriquecido con componentes adicionales de operones. 134

Ilustración 90: Modelo hipotético planteado para regulador transcripcional hallado en los quasi-cliques del prototipo 6. 135

RESUMEN

El análisis de redes de interacciones se aplica en forma creciente en investigación biológica. Se han reportado análisis derivados de los métodos desarrollados para análisis de redes sociales en diferentes organismos y con distintos objetivos. Para el análisis de redes biológicas se manipulan estructuras de datos en memoria que en su mayoría no están optimizadas para grandes cantidades de datos y en las cuales no es posible realizar exploración de datos conectados en red de manera ágil y flexible. Una alternativa de uso creciente para este tipo de situaciones, no solo en biología, es el uso de bases de datos no relacionales orientadas a grafos.

En este trabajo investigamos el uso de bases de datos no relacionales orientadas a grafos para el análisis de datos moleculares de la bacteria *Mycobacterium tuberculosis,* causante de la tuberculosis humana. Específicamente utilizamos datos de experimentos que cubren diversos procesos biológicos: expresión génica, interacciones proteína - proteína e interacciones proteína - ADN. Este tipo de enfoque integrado se conoce como biología de sistemas y es un área donde el uso de bases de datos de grafos todavía está en su infancia, pero tiene enorme potencial. Para los análisis se construye un multigrafo mixto y se definen preguntas a nivel biológico que posteriormente se traducen en patrones de consultas con quasi-cliques; las consultas se implementan en dos bases de datos no relacionales orientadas a grafos: Neo4J (http://neo4j.com/) y Titan (http://thinkaurelius.github.io/titan/) haciendo uso de lenguajes de consulta tales como Cypher y Gremlin.

Palabras Clave: quasi-clique, base de datos de grafos, multigrafo, mecanismo regulatorio, *Mycobacterium tuberculosis*.

ABSTRACT

The Interaction Network Analysis applies increasingly in biological research. Derivatives methods have been reported developed for analysis of social networks in different organisms and different objectives. For the analysis of biological networks in memory data structures that mostly are not optimized for large amounts of data and where not possible exploration network data online swiftly and flexibly handled. An alternative scenario growing use such situations, not only in biology, is the use of non-relational graph databases.

We investigated the use of non-relational graph databases oriented to analysis of molecular data of the bacterium *Mycobacterium tuberculosis*, which causes human tuberculosis. Specifically, we use data from experiments covering various biological processes: gene expression, protein-protein interactions and protein – DNA interactions. Such an approach known as Integrated Systems Biology and is an area where the use of bases graph databases is still in its infancy but has huge potential. For analyzes a mixed multigraph is constructed and biologically questions later translate into patterns of queries with quasi-cliques are defined; the queries are implemented in two non-relational graph databases: Neo4J (http://neo4j.com/) and Titan (http://thinkaurelius.github.io/titan/) by use query languages like Cypher and Gremlin.

Keyword: quasi-clique, graph database, multigraph, regulatory mechanisms, *Mycobacterium tuberculosis*.

1. INTRODUCCION

1.1 MOTIVACIÓN

El análisis de redes de interacciones se aplica en forma creciente en investigación biológica [1]. Se han reportado análisis de cliques y otros derivados de los métodos desarrollados para análisis de redes sociales (SNA) en diferentes organismos y con distintos objetivos [2] [3] [4]. En general para el análisis de redes biológicas se emplean datos de un organismo en particular y mediante librerías en algún lenguaje de tipo *scripting*, como R o *python*, se manipulan estructuras de datos en memoria que no están optimizadas para grandes cantidades de datos y en las cuales no es posible realizar exploración de datos de manera ágil y flexible.

Mycobacterium tuberculosis (*M. tuberculosis*) es la bacteria causante de la tuberculosis humana. En 2013 se estima que nueve millones de personas desarrollaron tuberculosis y un millón y medio murieron por esta enfermedad [5]. *M. tuberculosis* debe sobrevivir dentro del ambiente hostil del hospedador humano, para ello debe sensar y responder a una gran variedad de microambientes cambiantes incluyendo el interior de diversas células del sistema inmune como macrófagos, granulomas, entre otras. Para regular estas respuestas la bacteria debe controlar el funcionamiento de diversos genes que conforman redes regulatorias que actúan en forma colectiva para coordinar la adaptación a los diferentes estreses generados por el hospedador [6].

En los últimos años ha aumentado considerablemente el conocimiento sobre diferentes procesos celulares de adaptación al estrés, de virulencia y de resistencia a antibióticos en *M. tuberculosis*. Este conocimiento se deriva de diversos trabajos de investigación que utilizan variadas técnicas experimentales que han permitido acumular una cantidad considerable de datos (ver por ejemplo, [7], [8], [9], [10], [11]). A pesar de esto, no ha ocurrido una integración sistemática del conocimiento, a pesar de los esfuerzos realizados por iniciativas como Tuberculist (http://tuberculist.epfl.ch/ [12]) y TB Database (http://www.tbdb.org/ [13]).

En el presente proyecto se plantea el uso de datos experimentales de biología molecular obtenidos de la cepa *M. tuberculosis H37Rv*, la variante más utilizada en investigación, para diseñar y construir un multigrafo mixto del cual se obtienen patrones - a partir del planteo de preguntas biológicas iniciales - que permiten revelar nuevas redes regulatorias o caracterizar mejor aquellas conocidas mediante la aplicación de técnicas SNA y análisis de cliques y quasi-cliques.

Los trabajos en los que se aplican técnicas de análisis de redes en biología emplean grafos simples, dirigidos o no, cada uno desarrollado para un tipo único de datos. Por ejemplo, datos de transcripción, de interacción proteína-proteína, interacción proteína-ADN, etc. El aporte significativo del presente proyecto es la integración de datos de diferentes tipos en un único multigrafo mixto usando herramientas orientadas a manipulación de grandes bases de datos no relacionales orientadas a grafos. Adicionalmente se presenta una serie de herramientas y procesos que usan tecnologías de Big Data y bases de datos no relacionales orientadas a grafos para realizar tareas de

exploración y descubrimiento de una manera más ágil, permitiendo responder una serie de preguntas a nivel biológico en forma sencilla y flexible, más cercana a la experiencia de los profesionales del área biológica.

1.2 GRAFOS Y MULTIGRAFOS

Una definición simple y exacta de grafo es la dada en [1] que dice: “Un grafo es un objeto matemático consistente en vértices y aristas que representan elementos y conexiones.”.

La teoría de grafos se remonta a Leonard Euler y el llamado “problema de los puentes de *Königsberg*" propuesto en 1736 el cual plantea lo siguiente: el río Pregel atraviesa la ciudad, como se muestra en la **Ilustración 1** en la parte izquierda. Siete puentes fueron construidos sobre el río. La pregunta es si es posible caminar alrededor de la ciudad de una manera que implicaría cruzar cada puente exactamente una vez; Euler demostró que no es posible representado dicho escenario en forma de grafo como se puede ver en la **Ilustración 1** en la parte derecha.

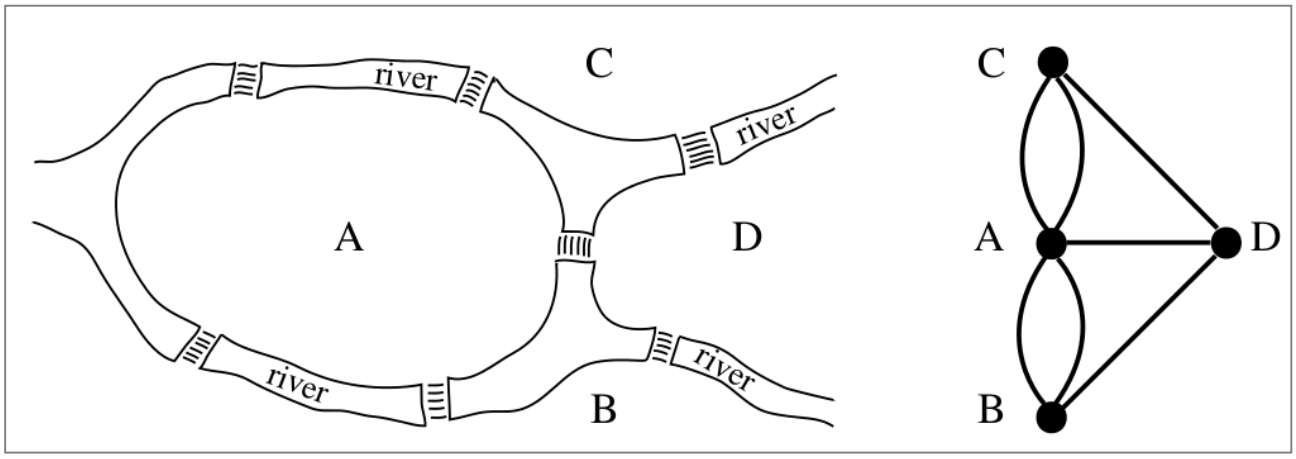

Ilustración 1: *A la izquierda representación del rio Pregel. A la derecha el grafo correspondiente a la solución al problema de los puentes de Königsberg [1].*

Desde este primer uso documentado de los grafos de Euler, éstos han sido utilizados en muchas áreas del conocimiento, permitiendo hoy en día representar tanto relaciones sociales de lo que se conocen como redes sociales, como también interacciones entre proteínas [14], redes eléctricas, entre otros.

El análisis de grafos presenta diversas consideraciones de tipo estructural que permiten definir patrones con determinados significados; a este estudio mediante teoría de grafos se le conoce como *Social Network Analysis (*SNA) o análisis de redes sociales y propone diversos algoritmos y análisis de patrones que son fácilmente trasladables de una disciplina a otra haciendo algunos ajustes de interpretación según el caso.

Muchos conceptos de teoría de grafos no serán expuestos debido a que esta es una disciplina muy extensa por lo que se plantearán solo los conceptos necesarios para el entendimiento del presente trabajo, tales como definiciones formales de grafos y multígrafos y las métricas para caracterizar SNAs, entre otros. Conceptos que en el presente trabajo se usarán para en análisis de datos biológicos.

1.2.1 Grafos

Tomando la definición formal planteada en [15], un grafo es un par $G = (V, E)$ donde los conjuntos satisfacen $E \subseteq [V]^2$; asi, los elementos de E son un subconjunto de 2-elementos de V. También asumimos que $V \cap E = \emptyset$. Los elementos de V son los vértices (nodos o puntos) del grafo G y los elementos de E son sus aristas (o líneas). Una forma usual para representar un grafo es dibujar un punto por cada vértice y unir dos de ellos con una línea correspondiente a una arista. Algo importante es que la forma en que estén dibujados tanto los puntos como las líneas es irrelevante, lo importante es la información de qué pares de vértices forman una arista o no. Ver **Ilustración 2**.

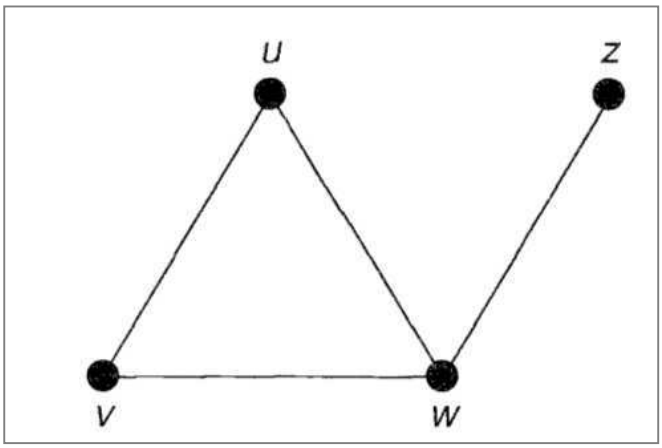

Ilustración 2: *Grafo G, donde el conjunto de vértices $V(G) = \{u, v, w, z\}$ y donde el conjunto de aristas $E(G) = \{uv, vw, wu, wz\}$ [16].*

1.2.2 Grafos dirigidos, no dirigidos, mixtos y multigrafos.

A continuación, las definiciones y algunos ejemplos de algunos tipos de grafos con sus respectivos ejemplos a nivel biológico [1] [17]. Ver **Ilustración 3**.

Grafos no dirigidos: En un grafo no dirigido, una arista entre un vértice u y v es representada por un par de vértices *no ordenados* $\{u, v\}$. Ejemplos en biología de estos grafos son redes de interacción de proteínas, redes filogenéticas y redes de correlación.

Grafo dirigido: En un grafo dirigido, una arista entre un vértice u y v es representada por un par de vértices *ordenados* $\{u, v\}$ en donde la línea que lo representa es comunmente finalizada con una flecha. En biología, las redes metabólicas, las redes de regulación de genes entre otras son ejemplos de estos grafos.

Grafo mixto: En un grafo mixto, ambos tipos de aristas, dirigidas y no dirigidas, están presentes. En biología un ejemplo es una red especial de interacción entre proteínas donde hay interacciones no dirigidas (obtenidas por experimentos del tipo *two-hybrid screening*) y otras dirigidas que representan, por ejemplo, activación fosforilación y otras interacciones que requieren orientación entre sus vértices.

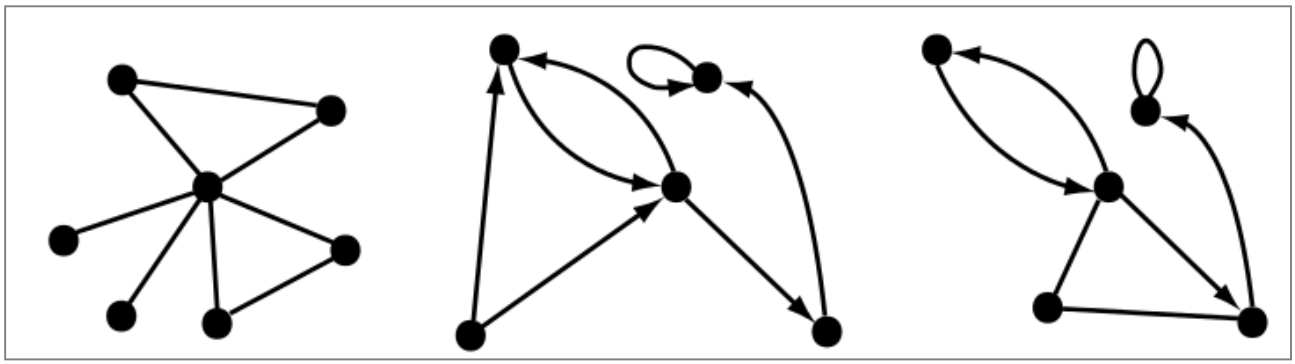

***Ilustración 3**: Grafo no dirigido, dirigido y mixto (de izquierda a derecha) [17].*

Multigrafos o pseudografo: Los multigrafos o pseudografos son grafos que contienen múltiples aristas, es decir, dos o más aristas que inciden en los mismos dos vértices y en el caso de grafos dirigidos tiene la misma dirección. Tales aristas son también llamadas aristas paralelas o multiaristas. En contraposición, los grafos no dirigidos, libres de ciclos y sin múltiples aristas son llamados *grafos simples*. Algunos autores como [18] plantean que un pseudografo es un multígrafo que admite ciclos, es decir, aristas de un vértice consigo mismo (Ver **Ilustración 4**), sin embargo, para el presente trabajo tomaremos la definición en la cual uno u otro es lo mismo [15].

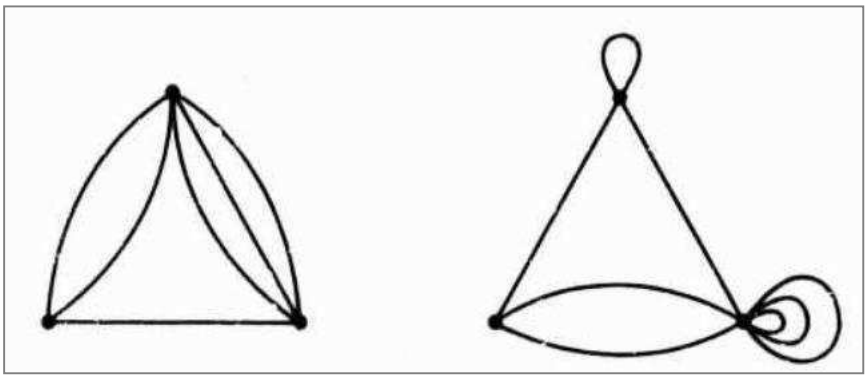

***Ilustración 4:** Multigrafo y/o pseudografo (de izquierda a derecha) según [18].*

Así, un multigrafo es un par (V, E) de conjuntos disjuntos (de vértices y aristas) junto con un mapa $E \to V \cup [V]^2$ que asigna para toda arista uno o dos vértices a sus extremos. Así, los multigrafos también pueden tener ciclos y múltiples aristas, sin embargo, todas sus aristas son dirigidas. Para expresar que x e y son los extremos de una arista podemos escribir $e = xy$, aunque esto ya no determina a e como única. De ésta forma, y mediante el concepto de contracción, podemos convertir una multiarista en un ciclo como se muestra en **Ilustración 5**.

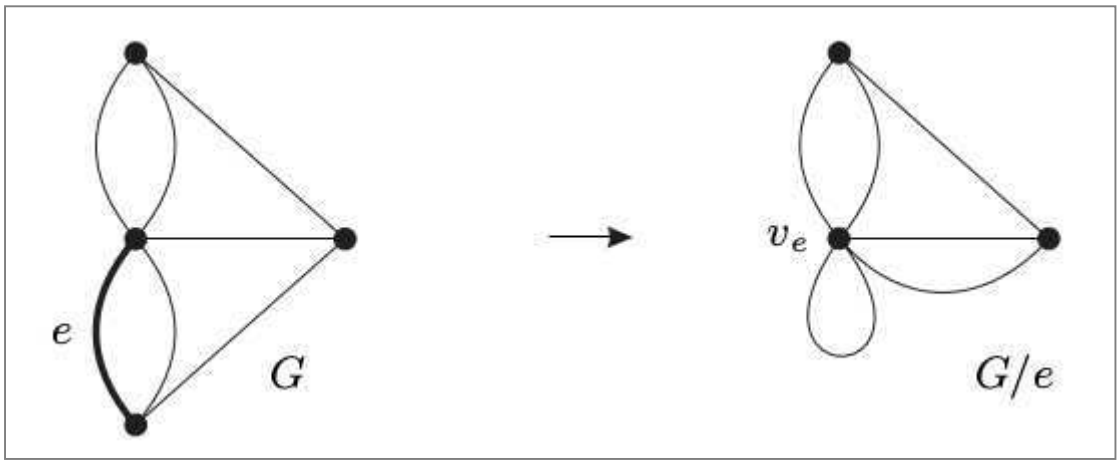

Ilustración 5: *Contracción de la arista e del multigrafo G en un multígrafo G/e [15].*

Multigrafo mixto: Un multígrafo mixto es aquel que además de permitir multi-aristas y ciclos, tiene aristas dirigidas (Ver **Ilustración 6**). Este grafo es especialmente útil para generar una sola representación de diferentes grafos de distinto tipo que planteen algo en común. Para los efectos del presente trabajo, se presentará un multígrafo mixto, que para efectos de facilidad se nombrará desde ahora simplemente multigrafo.

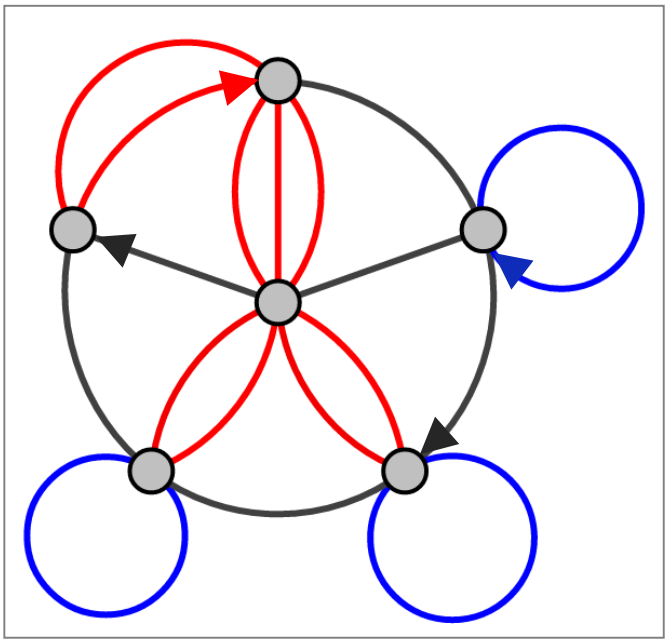

Ilustración 6: *Ejemplo de un multigrafo mixto. (Original tomado de www.wikimedia.com)*

1.2.3 Social Network Analisys (SNA)

El análisis de redes sociales o SNA es una de las herramientas más importantes y usadas en la actualidad para la interpretación de patrones en grafos tanto en áreas que modelan comportamiento o interacciones humanas como áreas que lo hacen con eventos biológicos, químicos o de otros tipos.

Desde el punto de vista histórico, el área se forma con el análisis de sociometría, en donde se logran avances técnicos con los métodos de teoría de grafos, posteriormente

investigadores de Harvard en la década del 1930 exploraron patrones de relaciones interpersonales y la formación de cliques a nivel social - grupos humanos donde se verifican todas la conexiones posibles entre miembros -; luego antropólogos de Manchester construyeron el concepto de comunidades en relación a sociedades tribales y pueblos. Finalmente, en las décadas de 1960 y 1970 en Harvard volvieron a revivir los conceptos antes mencionados y se construye gran parte de lo que en la actualidad se conocen como análisis de redes sociales [19] (Ver **Ilustración 7**).

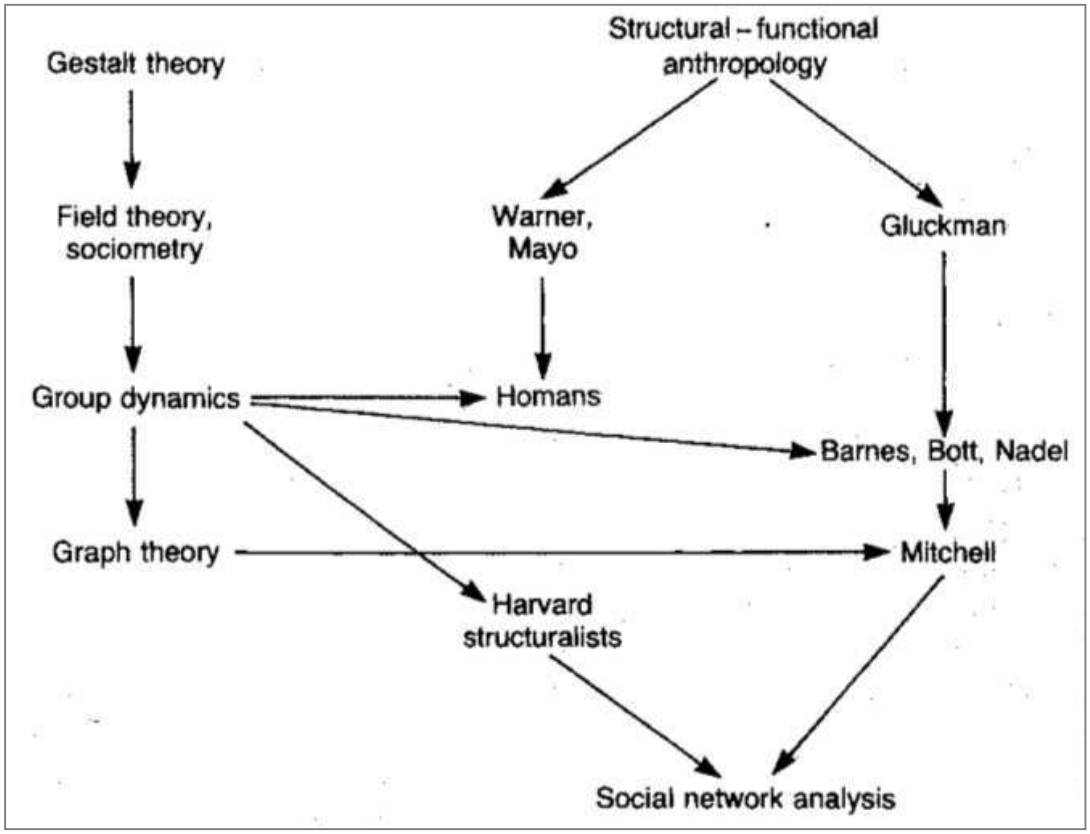

Ilustración 7: *Evolución del análisis de redes sociales [19].*

A continuación, se mencionarán algunas de las múltiples medidas y conceptos de teoría de grafos y SNA muy comúnmente usados en análisis de grafos y que para el desarrollo del presente trabajo es necesario su conceptualización.

1.2.3.1 Propiedades globales de grafos complejos

El análisis de ciertas propiedades o medidas de los grafos a nivel global, permite darse una idea de su composición a nivel estructural, por lo cual en estados iniciales de análisis es de suma utilidad y un paso natural para cualquier proceso que implique manipulación de grafos. A continuación, se mencionarán algunas de las medidas globales más básicas e importantes.

1. **Tamaño de camino promedio (*shortest path*):** es el mínimo número de aristas que se necesitan para atravesar el camino entre el vértice n_i al n_j. El camino más corto entre dos vértices no tiene que ser único, es más, comúnmente existen diversos caminos alternativos entre ellos [1]**.**

2. **Distancia (*distance*):** En un grafo consistente de N_V vértices, la distancia d_{ij} entre dos vértices n_j es dada por el tamaño del camino más corto (*shortest path*) entre los vértices. Así, para grafos dirigidos, la distancia entre dos vértices n_i a n_j es usualmente asimétrica, es decir $d_{ij} \neq d_{ji}$. También es un caso posible en donde en un grafo no conectado (*disconnected network*) existen componentes separados en el que no existe un camino que conecte los vértices n_i al n_j, en cuyo caso la distancia respectiva entre los mismos será infinito: $d_{ij} = \infty$ [1].

3. **Diámetro (*diameter*):** el diámetro $d_m = \max(d_{ij})$ de un grafo es definido como la máxima distancia entre cualquier par de vértices. El promedio o distancia característica $d = \langle d_{ij} \rangle$ de un grafo es definida como el promedio de distancias entre todos los pares de vértices. En el caso de distancias infinitas, el promedio inverso de tamaño de camino $d_{eff} = \langle \frac{1}{d_{ij}} \rangle$ también conocido como eficiencia *(efficiency)* puede ser usado para especificar el promedio del tamaño de camino dentro de una red. En éste caso, un grafo completamente conectado $d_{ij} = 1\ \forall i, j$ tiene una eficiencia $d_{eff} = 1$, mientras que para distancias grandes y componentes no conexos (usando el límite de $1/d_{eff} \rightarrow 0\ para\ d_{ij} \rightarrow \infty$) se reduce la eficiencia del grafo [1].

4. **Distribución de grado (*degree*):** el grado k_i de un vértice n_i se define como el número de aristas adyacentes a un vértice. En un grafo sin ciclos (*self-loops*), es decir, aristas que conectan un vértice consigo mismo ni múltiples conexiones (dos vértices están conectados por más de una arista), el grado es igual al número de vecinos del vértice. En el caso de grafos dirigidos, se distinguen grado de entrada (*input degree*) k_i^{in} y grado de salida k_i^{out} *(output degree)*. Ahora bien, tomando en cuenta todos los vértices del grafo, un dato de interés es la probabilidad $p(k)$ del grado de un vértice aleatoriamente seleccionado igual a k [1].

1.2.3.2 Medidas de Centralidad

Una función que asigna un valor numérico a cada vértice de un grafo es llamada centralidad. Así dado un grafo dirigido o no dirigido $G = (V, E)$ tenemos una función $\mathrm{C}: V \rightarrow \mathbb{R}$ que llamaremos centralidad; de ésta forma la centralidad asigna a cada vértice un número real. Esto permite una comparación a pares entre los vértices de un grafo, siendo v_1 un vértice con mayor centralidad que un v_2 si $\mathrm{C}(v_1) > \mathrm{C}(v_2)$. Finalmente existen dos consideraciones a la hora de analizar la centralidad en un grafo:

1. Los valores de centralidad son solo comparables a nivel de una red específica.
2. Algunas medidas de centralidad solo pueden aplicarse a redes que sean conectadas.

Existen diversos conceptos para calcular la centralidad y distintos tipos de ellas, a continuación, veremos las más usadas [1] [19].

a) **Centralidad de grado (*degree centrality*)**: Consiste en el conteo del número de aristas conectadas a un vértice. Así, dado un grafo no dirigido $G = (V, E)$ el grado de centralidad es definido como $C_{deg}(v) \coloneqq |\{e | e \in E \wedge v \in e\}|$ [1].

b) **Centralidad de Cercanía *(closeness centrality)*:** se define como el recíproco de

la suma de todas las distancias a pares que componen el grafo. Así, dado un grafo no dirigido y conectado $G = (V, E)$ el grado de centralidad de cercanía es definido como $C_{clo}(s) \coloneqq \frac{1}{t \in V^{dist(s,t)}}$, donde $dist(v, t)$ denota la distancia entre el vértice s y t, que es el tamaño del camino más corto entre s y t [1].

c) **Centralidad de Intermediación (*shortest path betweenness centratily*):** es una métrica que permite desde un vértice monitorear su comunicación con otros vértices. Así, σ_{st} denota el número de caminos más cortos entre dos vértices s y t; además denotamos $\sigma_{st}(v)$ como el número de caminos más cortos entre s y t que usa v como un vértice interior. La tasa de comunicación entre s y t que puede ser monitoreada por un vértice interior v es denotada por $\delta_{st}(v) \coloneqq {}^{\sigma_{st}(v)}/_{\sigma_{st}}$. Si no existen caminos más cortos entre s y t ($\sigma_{st} = 0$), entonces se dice que $\delta_{st}(v) \coloneqq 0$. Así, dado un grafo no dirigido $G = (V, E)$ el grado de centralidad de intermediación es definido como $C_{spb}(v) \coloneqq {}_{s \epsilon V \wedge s \neq V \; t \in V \wedge t \neq v} \delta_{st}(v)$ [1].

Para poder comparar los valores de centralidad de intermediación - y cualquier otro – a través de distintos grafos, es necesario realizar una normalización, para lo cual es necesario conocer el máximo valor que dicha métrica toma. Así, la centralidad de intermediación toma su máximo valor cuando el nodo v_i es entre los caminos más cortos de s a t para cualquier par (s, t); esto es, $\forall (s, t), s \neq t \neq v_i, \frac{\delta_{st}(v_i)}{\delta_{st}} = 1$, en donde, el máximo valor es:

$$C_b(v_i) = \sum_{s \neq t \neq v_i} \frac{\sigma_{st}(v)}{\sigma_{st}} = \sum_{s \neq t \neq v_i} 1 = 2\binom{n-1}{2} = (n-1)(n-2)$$

De ésta manera, la intermediación puede ser dividida por su máximo valor para obtener la centralidad de intermediación normalizada [19, 17]:

$$C_b^{norm}(v_i) = \frac{C_b(v_i)}{2\binom{n-1}{2}}$$

1.2.3.3 Cliques

Como se mencionó anteriormente, el estudio de cliques en sociometría fue planteado en la década de 1930 por investigadores de Harvard [19]. Específicamente, los primeros investigadores en formalizar la definición de clique fueron Luce and Perry (1949) and Luce (1950) [20].

De ésta forma, el subgrafo más común buscado en un grafo es un clique. El clique es un subgrafo completo en el cual todos sus pares de nodos adentro del subgrafo están conectados entre sí. En términos de características de nodos, un clique de tamaño k es un subgrafo de k nodos donde todos los nodos en el subgrafo inducido son de tipo $k - 1$ [19] [21]. Un ejemplo de k-cliques puede ser visto en la **Ilustración 8**.

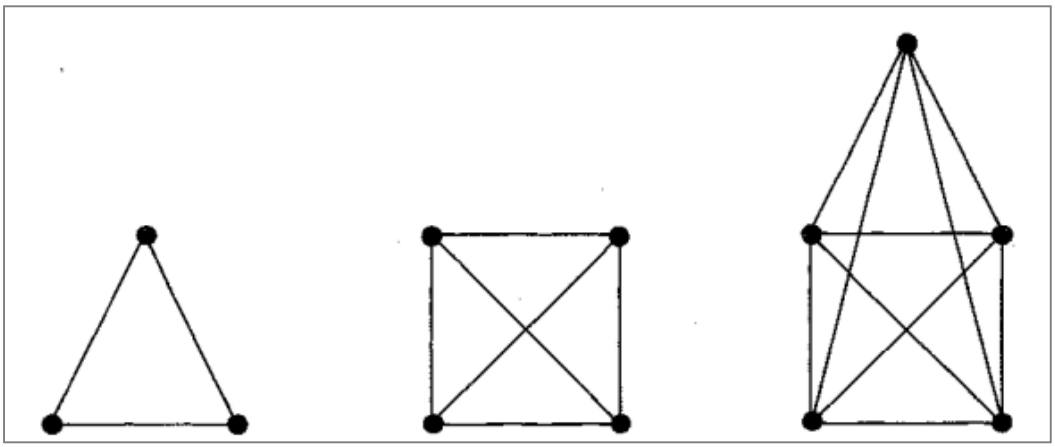

Ilustración 8: *k-clique con k=3, k=4 y k=5 (de izquierda a derecha respectivamente) [19].*

Por otro lado, algunos otros investigadores plantearon el concepto de cliques fuertes (*strong cliques*) y cliques débiles (*weak cliques*) definiendo que el primero es el que tiene todas sus relaciones completas, es decir, es el máximo subgrafo completo (*maximal complete subgraph*), en tanto que el segundo tiene ausencia de algunas de ellas. Un concepto que se deriva de esto es que dadas las restricciones de las redes sociales reales, aparecen conceptos de cliques "parciales" en donde un n-clique es la expresión de completitud en cuanto a relaciones se refiere de un k-clique determinado. Así, un 1-clique es un máximo subgrafo completo, es decir, un conjunto en el cual todos los pares de puntos son directamente conectados por una distancia 1; un 2-clique, por otra parte, es uno en el cual los miembros son conectados directamente por distancias 1 o indirectamente a vecinos comunes (distancia 2), un ejemplo de esto puede ser visto en la **Ilustración 9**.

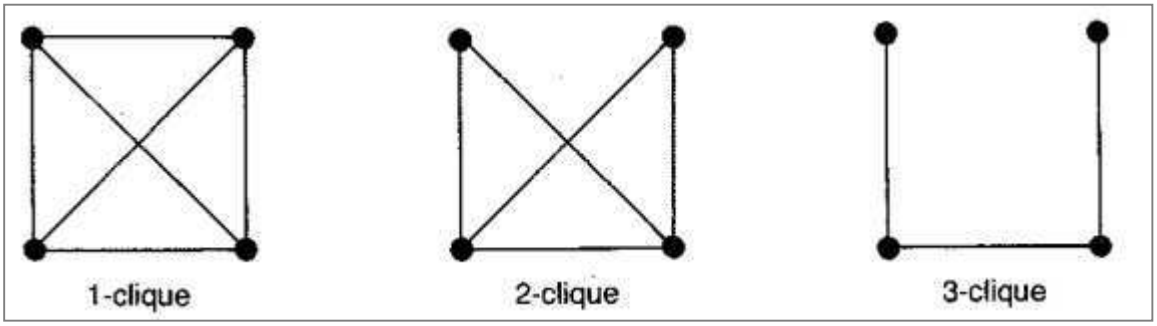

Ilustración 9: *n-clique con n=1, 2, 3 de tamaño k=4 [19].*

De ésta manera, un progresivo incremento en el valor de n, resulta en una gradual relajación del criterio de membrecía del clique. El máximo valor de n es $k-1$, donde k es el total de nodos del grafo. En sociometría, este análisis trae consigo limitaciones de tipo analítico e interpretativo, sin embargo, dicho concepto ayudó a la definición de posteriores ideas como *n-clan*, *k-plex*, componentes y círculos sociales con intersecciones, entre otros [19] [20].

Finalmente, un máximo clique de un grafo (*maximum clique*) es el máximo subgrafo k-clique posible perteneciente a un grafo determinado.

Existen múltiples conceptos derivados de lo anteriormente planteado entre los cuales se encuentran búsqueda de comunidades, optimizaciones de algoritmos para hallar cliques,

máximo cliques y *quasi-cliques*, entre otros. Estos conceptos no son mencionados debido a que no hacen parte del objetivo principal del presente trabajo.

1.2.3.4 Cliques débiles o quasi-cliques

En la literatura, existen diversas aplicaciones de cliques o "relajaciones" de éstos, no solo en el campo de la sociometría – como ya se ha mencionado – sino también en otros como la biología, la química y la electrónica, entre otros.

El concepto de clique no completo o cliques débiles (*weak cliques*) ha sido referenciado y usado en muchos trabajos debido a que el cliqué definido de manera estricta (strong clique) en general no se encuentra de manera frecuente en grafos reales. A continuación, se presentan algunos trabajos en donde se exponen dichos tipos de cliques débiles enfocados a la biología y la sociometría.

Por ejemplo, el concepto de casi-clique (*near-clique*) es usado para mostrar en una red de interacción proteína-proteína que un 77% de los *near-clique* encontrados por el algoritmo propuesto, comparten al menos una función dentro del mismo, y que un 68% de los *near-clique* son correspondientes a procesos experimentales ya catalogados en la literatura y en bases de datos [2]. En el trabajo mencionado se presentan 3 tipos de casi-cliques:

1. Cuando una proteína fuera de un clique interactúa con dos o más proteínas dentro del clique, la proteína y el clique forman un *near-clique*.
2. Cuando un clique comparte una proteína con otro clique, los cliques fusionados forma un *near-clique*.
3. Cuando dos o más cliques interactúan con una proteína común afuera de ellos y la proteína tiene al menos dos interacciones con cada clique, el clique y la proteína forma un *near-clique*.

En otros casos, no se usan cliques débiles sino los máximos cliques, por ejemplo, en [3], donde la idea principal del método propuesto es predecir la interacción a través de proteínas basado en el grado de correlación entre la proteína y el clique de proteínas usando máximos cliques. Sus resultados demostraron que mediante estructuras clique se pudieron predecir entre un 80% y un 90% de las interacciones en los datos usados de prueba.

En algunos otros casos [4] [22] [23] [24], se usan las siguientes definiciones de relajación de cliques:

- Grafo casi-completo (*γ quasi-complete graph):* Un grafo $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ es un $\gamma - \mathrm{quasi} - \mathrm{complete}$ graph $(0 < \gamma \leq 1)$ si cada vértice en el grafo tiene al menos grado $[\gamma.(|\mathrm{V}| - 1)]$.
- *Casi-clique (γ-quasi-clique)*: En un grafo $\mathrm{G} = (\mathrm{V}, \mathrm{E})$ un subconjunto de vértices $V' \in V$ forman un *γ-quasi-clique* $(0 < \gamma \leq 1)$ si el subgrafo inducido por $\mathrm{V}', \mathrm{G}_{\mathrm{V}'}$, es un *γ quasi-complete graph.*

Para el caso de [4], los autores plantean la necesidad de "relajar" la definición de cliques para encontrar *quasi-cliques* en grandes grafos, para lo cual utilizan la definición de los conceptos antes mencionados. Los autores proponen entonces un algoritmo heurístico que localiza *quasi-cliques* de redes de interacciones proteína-proteína en humanos, obteniendo resultados promisorios para investigadores en biología computacional – como lo mencionan los autores – para la exploración de módulos de proteínas significativos.

Otro ejemplo es el trabajo [22], en donde los autores introducen el concepto de *quasi-clique correlacionado*, también haciendo uso del concepto *quasi-clique*, desarrollando un nuevo algoritmo CoClique que resuelve el problema y mejora la eficiencia de la búsqueda de *quasi-cliques correlacionados* en conjuntos de datos de interacción proteína-proteína.

De manera similar, en sociometría, específicamente en grandes conjuntos de datos de redes sociales como Facebook y Twitter se han utilizado *quasi-clique* para recomendar comunidades a un usuario en particular, seleccionando antes "relaciones de amistad fuertes" [23].

Finalmente, y para ejemplificar otro uso de los *quasi-clique,* en la literatura se encuentran trabajos en los cuales la optimización de la búsqueda de *quasi-cliques* es el objetivo principal. En el caso de [24], los autores orientan su algoritmo de optimización para usarlo en el análisis de correlación en precios de inventarios, predicción de funciones de genes y análisis de proteínas a nivel molecular entre otros.

1.2.4 El concepto de NoSQL o No relacional.

La nueva generación de bases de datos que surgió posterior a las bases de datos de tipo relacional han sido categorizadas bajo el nombre de NoSQL, acrónimo de *Not Only SQL*, sin embargo, en la actualidad dicha categorización es controversial.

Existen tres conceptos claves antes de analizar con un poco de profundidad las bases de datos NoSQL: ACID *(Atomicity, Consistency, Isolation, Durability)*, el teorema de CAP o teorema de Brewer's *(Consistency, Availability, Partition tolerance*) y el concepto de BASE (*BAsically available, Soft state, Eventual consistency*); todos ellos permiten entender las bases teóricas a nivel transaccional y funcionamiento que tanto las bases de datos relacionales clásicas como las NoSQL cumplen según su tipo y uso.

Jim Gray en 1981 planteo el concepto de **ACID** *(Atomicity, Consistency, Isolation, Durability)* [25] con lo cual invento el proceso de transaccionalidad moderno que las bases de datos actualmente usan:

- **Atomicidad (*Atomicity*):** cada una de las tareas dentro de una transacción es ejecutada o ninguna de ellas lo es, es decir, si alguno de los elementos falla, entonces la transacción falla.
- **Consistencia (*consistency*):** La transacción debe cumplir con todos los protocolos o reglas definidas por el sistema en todo momento. La transacción no viola los protocolos y la base de datos debe permanecer en un estado coherente al comienzo y al final de cada una de las transacciones.
- **Aislamiento (*Isolation*):** Ninguna transacción tiene acceso a ninguna otra transacción en un estado intermedio o no finalizado. Esto conlleva a que existan niveles de aislamiento (*levels of isolation*) y por ende que éste principio se cumpla total o parcialmente en una base de datos según el nivel de aislamiento que siga el sistema.
- **Durabilidad (*Durability*):** Una vez la transacción es completada, ella persistirá como completa y no se puede deshacer, sobreviviendo a cualquier falla del sistema.

De ésta forma y de manera complementaria existen los conceptos de:

- Modelos de **concurrencia pesimista** *(pessimistic concurrency model)* y modelo de

concurrencia optimista (*optimistic concurrency).*
- **Niveles de aislamiento** *(Isolation Levels).*

Estos conceptos no serán explicados, pero pueden ser ampliamente consultados en [25].

Por otro lado, en 2000, Eric Brewer presentó el concepto de **CAP** *(Consistency, Availability, Partition tolerance*) o teorema de Brewer's (Ver **Ilustración 10**) para sistemas de computación distribuida, el cual fue revisado y modificado por Seth Gilbert y Nancy Lynch del MIT en 2002 y posteriormente por algunos otros. Éste teorema asume una concurrencia en un entorno distribuido y no en un administrador central de concurrencia como en el modelo clásico, lo cual trae consigo algunas consideraciones [25]:

- **Consistencia *(Consistency)*:** responde a las preguntas: ¿El sistema es fiable a seguir las reglas establecidas por el contenido de sus datos?, ¿Todos los nodos de un clúster ven los mismos datos en el mismo estado?.
- **Disponibilidad (*Availability*):** significa que el servicio o sistema está disponible cuando se hace una petición.
- **Tolerancia a partición (*Partition tolerance)*:** o robustez, significa que, dado un sistema, este continúa operando incluso con datos perdidos y fallos de sistema.

Así, cualquier sistema distribuido solo puede garantizar solo dos de las tres características mencionadas [25].

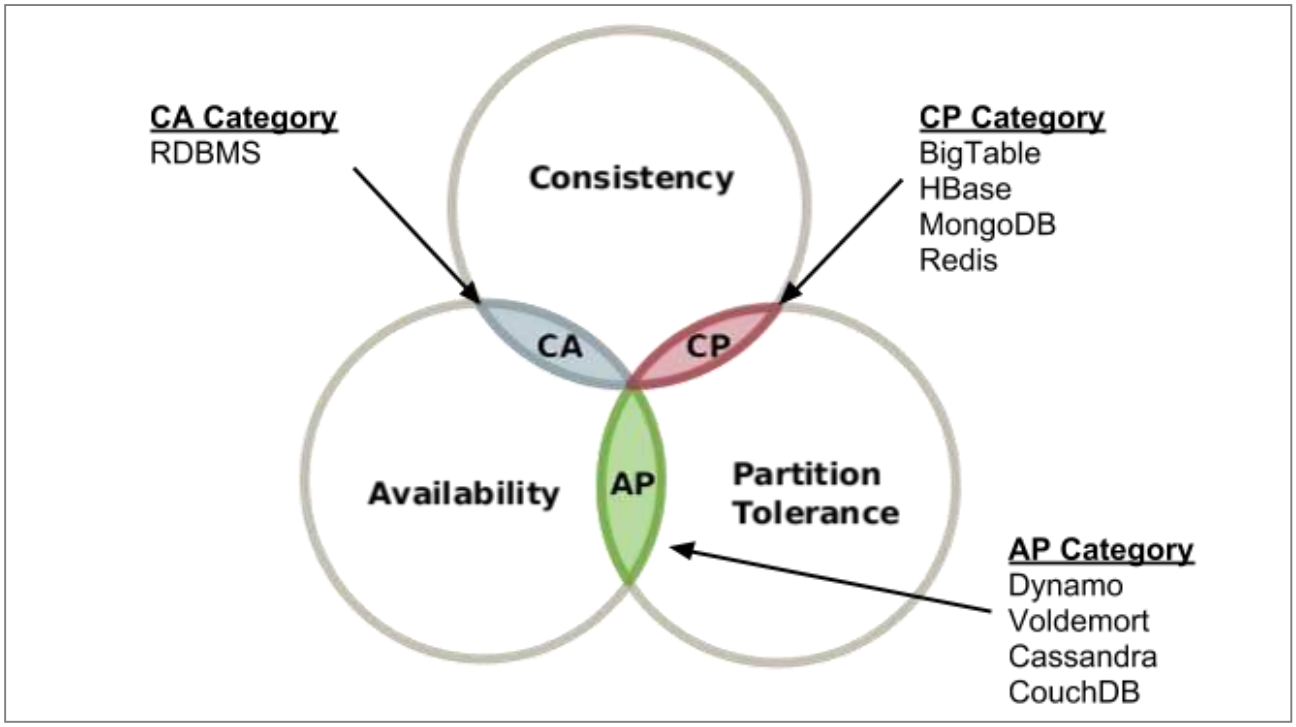

Ilustración 10: Teorema de **CAP** o teorema de *Brewer's* con algunos ejemplos de bases de datos que usan dos de sus tres principios en cada intersección (Tomado de http://changeaas.com).

Siguiendo la misma línea y tomando como una alternativa a ACID, se tiene el concepto de **BASE** (*BAsically available, Soft state, Eventual consistency*) para sistemas distribuidos:

- **Básicamente disponible *(Basically available)*:** significa que el sistema garantiza disponibilidad planteada por el teorema de CAP, sin embargo, la respuesta puede ser: no disponible (*unreliable*) a causa de que un dato pedido tiene una inconsistencia o un cambio de estado.

- **Estado "blando"** ***(Soft state):*** el estado del sistema puede cambiar con el tiempo hasta que eventualmente se logra la consistencia de los datos en un tiempo específico.
- **Consistencia Eventual (*Eventual consistency*):** eventualmente el sistema será consistente dando ventanas de tiempo en que este es inconsistente.

Así como el modelo ACID, el modelo BASE también tiene variaciones como: *Causal consistency, Read-your-writes consistency, session consistency, monotonic read consistency* y *monotonic write consistency* [25].

Ahora bien, las bases de datos NoSQL son sistemas de bases de datos *task-oriented*, es decir, orientadas a tareas específicas las cuales se pueden categorizar en cuatro tipos [26]:

- Almacenamiento clave-valor (*key-value stores*).
- Familia de almacenamiento de tipo columna *(Column-family stores).*
- Almacenamiento de documentos *(Document store).*
- Bases de datos de grafos (*Graph database).*

A continuación, se hará una breve descripción de las primeras tres, dejando las bases de datos de grafos como el tema central del siguiente apartado, ya que en el presente trabajo constituyen unas de las bases fundamentales para su desarrollo.

1. **Almacenamiento clave-valor (*key-value stores*):** Son probablemente las bases de datos NoSQL orientas a tareas más simples. El modelo de datos de la tarea original se basa en guardar datos de tipo clave:valor *(key-value);* éste modelo fue expuesto por Amazon en 2007 en el llamado *Dynamo paper* [27]. El modelo de datos discutido en el trabajo, es correspondiente al sistema de carro de compras de Amazon, en el cual se requería estar siempre disponible y soportar una carga de trabajo extrema. En las bases de datos de tipo clave-valor, los datos se guardan en un modelo de datos inherente sin esquema. Como se puede ver en la **Ilustración 11**, una característica de registros, es que la existencia de los atributos no es obligatoria, y por ende valores vacíos o de tipo nulo no debería existir, como si ocurre en bases relacionales tradicionales. La escalabilidad es extremamente alta, compuesta en general por miles de máquinas o nodos que distribuyen la carga y administran los posibles fallos del sistema. Algunos ejemplos de base de datos clave-valor son: DynamoDB, Riak, Project Valdemort, Redis, Aerospike, entre otras [26, 28, 29].

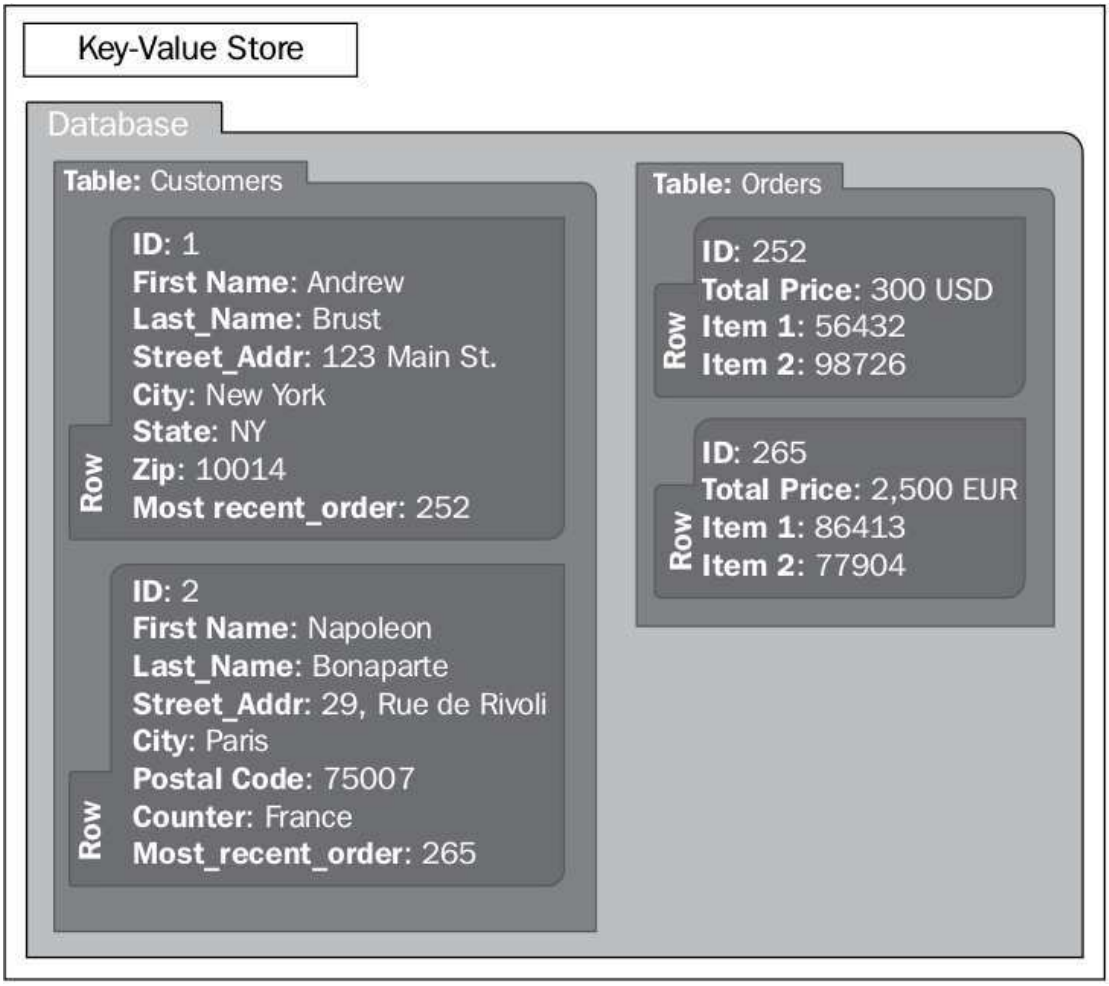

Ilustración 11: *Ejemplo de estructura de datos de bases de datos de tipo key-value store [26].*

2. **Familia de almacenamiento de tipo columna (*Column-family stores)*:** Éste modelo de datos es un poco más complejo que el de clave-valor, ya que incluye el concepto de una estructura de tabla amplia, escasamente poblada, que incluye un número de familias de columnas que especifican las claves para esta estructura de tabla particular. Un ejemplo de esto puede ser visto en la **Ilustración 12**, en donde se pueden visualizar claramente los registros de cada familia de columnas, y en su interior la asociación a otras familias de columnas con atributos específicos. Al igual que las bases de datos clave-valor, la familia de almacenamiento orientado a columnas surgió para resolver un problema, en éste caso, de Google, quien en 2006 expuso el trabajo sobre *BigTable* en *Operating Systems Design and Implementation (OSDI) symposium*, trabajo que fue publicado posteriormente en 2008 [30]. El trabajo no solo inicio grandes esfuerzos al interior de Google, sino también generó interés en implementaciones *Open Source* como Apache Cassandra y Hbase. En muchos casos, este sistema es combinado con sistemas orientados a lotes (*batch-oriented*), que usan *Map Reduce* [31] como modelo de procesamiento para consultas avanzadas o un lenguaje de alto nivel que facilita los procesos de consulta que se terminan ejecutándose mediante *Map Reduce* [26, 28, 29].

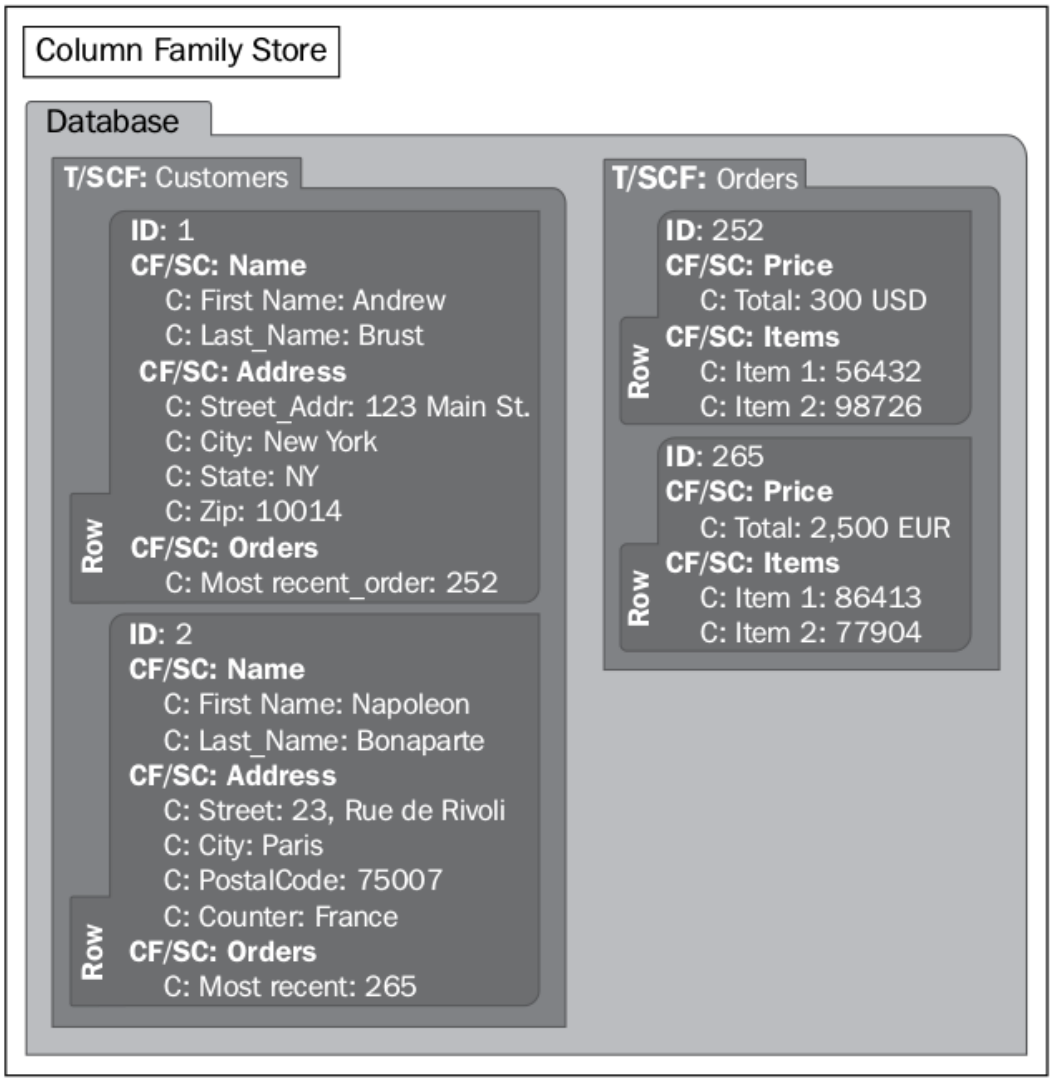

Ilustración 12: Ejemplo de estructura de datos de bases de datos de tipo *Column Family Store [26].*

3. **Almacenamiento de documentos *(Document store):*** Motivados por el crecimiento explosivo de contenido y aplicaciones web, probablemente uno de los tipos de bases de datos *NoSQL* más conocidos y más utilizados se encuentran en la categoría de documentos. Su concepto clave - como su nombre lo sugiere - es guardar unidades de información semi-estructuradas a menudo referenciadas a un documento (Ver **Ilustración 13**); esto puede ser un tipo de documento XML, JSON, YAML, OpenOffice, MS Office entre otros. Algunos ejemplos incluyen a MongoDB, Couchbase, MarkLogic y Virtuoso, entre otras [26, 28, 29].

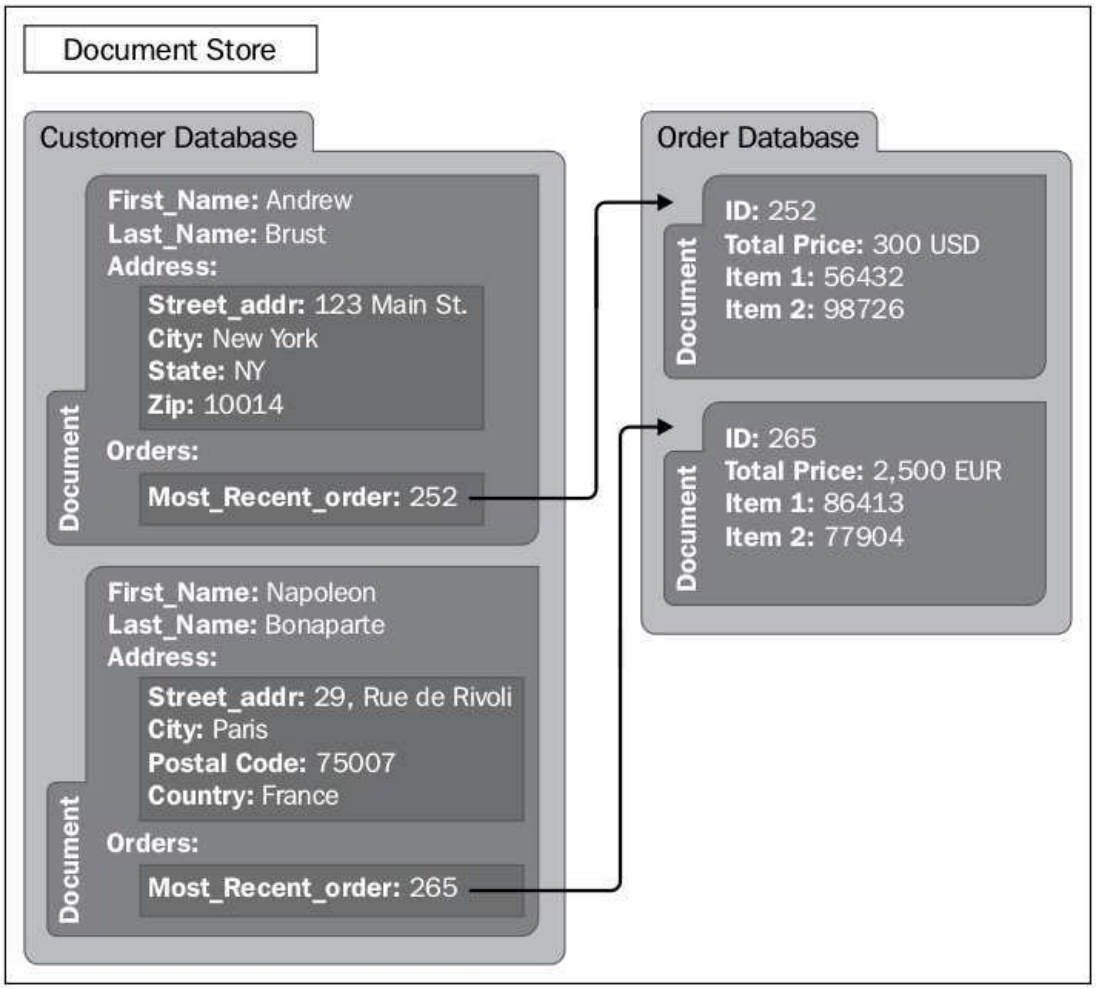

Ilustración 13: Ejemplo de estructura de datos de bases de datos de tipo *Document Store [26].*

1.2.5 Grafos en bases de datos

El segmento de las bases de datos es uno de los más grandes y en constante crecimiento y evolución en el mercado del software. Tanto a nivel comercial y educativo como en el ámbito de la investigación científica, los DBMS *(Data Base Management Systems)* hacen parte vital del engranaje tecnológico a tener en cuenta.

Actualmente, como se puede reflejar en el último informe de *Gartner* sobre DBMS [32] se puede observar que existen características de las DBMS que son tendencia actualmente, algunas de las más importantes son:

a) Soporte del modelo ACID (*atomicity, consistency, isolation and durability*).
b) En general soporte de sistemas NoSQL.
c) Modelos de datos clave:valor (key:value).
d) Almacenamiento en memoria *(in-memory*) y almacenamiento en memoria de tipo columnas *(in-memory-column-storage).*
e) Características de escalabilidad, alta disponibilidad y capacidad de operación en la nube (*cloud*).

Ilustración 14: Cuadrante mágico de *Gartner* para sistemas de gestión de bases de datos 2015 *[32].*

Como se observa, quizás la característica que llama más la atención es la habilidad de la mayoría de los productos de soportar algún tipo de tecnología NoSQL, ya sea de tipo propietario u Open Source. En la **Ilustración 14** vemos que productos de empresas como Oracle, Microsoft, IBM, Amazon Web Services y SAP se consolidan como líderes, sin embargo, productos de empresas como MongoDB, DataStax (Distribución comercial de Cassandra), MariaDB, Cloudera (distribución comercial de Hadoop) y Neo Technology, entre otros, se afianzan en el mercado gracias a sus sólidas características y enfoque de mercado especializado.

Ahora bien, centrando la mirada en el tema que concierne al presente trabajo, vemos que las bases de datos orientadas nativamente a grafos aparecen hace ya algunos años en este tipo de informes, tal es el caso de Neo Technology con NEO4J, DataStax con Titan DB y Orient Technologies con OrientDB; las dos principales razones para éste comportamiento pueden ser:

- Los datos que siguen un esquema de redes o grafos se han ido popularizando en los últimos tiempos, gracias a la aparición de enormes conjuntos de información provenientes de:

- Redes sociales (Facebook, Twitter, entre otras) y sistemas masivos de información en internet.
- Fuentes de datos científicos, como pueden ser datos genéticos o proteicos, gracias a la popularización y abaratamiento de los procesos de secuenciación de ADN y al aumento de estudios enfocados en interacciones biológicas.

- Las bases de datos relacionales pueden ser usadas para almacenar y operar con grafos, sin embargo, no resulta óptimo ni natural su uso y explotación.

Es así, como las bases de datos orientadas a grafos, toman el paradigma de los datos y plantean un conjunto de soluciones para su explotación de manera nativa, ganando de ésta forma un lugar privilegiado en el mundo de las bases de datos.

De manera similar, en *db-engine ranking* [33] se muestran bases de datos orientadas a grafos ubicadas en los primeros 60 lugares según su popularidad (Ver **Tabla 1**). En dicho listado, se ubican DBMS posicionadas teniendo en cuenta las siguientes métricas y parámetros:

a) Número de menciones de los sistemas en sitios web.
b) Interés general en el sistema.
c) Frecuencia de discusiones técnicas acerca del sistema.
d) Número de ofertas de trabajo en las que el sistema es mencionada.
e) Número de perfiles profesionales en las cuales el sistema es mencionada.
f) Relevancia del sistema en redes sociales.

Puesto	Producto	Empresa	Tipo
8	Cassandra	Open Source – DataStax	Wide column store
12	Solr	Open Source - Basado en Apache Lucene	Search engine
13	Elasticsearch	Open Source - Basado en Apache Lucene	Search engine
16	HBASE	Open source - Apache Hadoop	Wide column store
21	**Neo4j**	**Neo Technologies**	Graph DBMS
42	**TitanDB**	**DataStax**	Graph DBMS
44	OrientDB	Orient Technologies LTD	Multi-model: Document store, Graph DBMS, Key-value store
59	Berkeley DB	Open Source – Oracle	Key-value store

Tabla 1: Listado de principales productos relacionados con bases de datos de grafos en db-engine ranking *[33]*.

Vemos que, en el tipo de bases de datos orientadas a grafos, las 2 primeras posicionadas son NEO4J y Titan DB, seguida de OrientDB. En la **Tabla 1** se incluyen igualmente algunas bases de datos de tipo *Wide column store* y *Key-value store* ya que en el caso de Titan DB pueden ser usadas para el almacenamiento de los datos de grafos de forma optimizada; además aparecen dos bases de datos orientadas a búsqueda (Solv y ElasticSearch) que son ampliamente utilizadas por base de datos de grafos para indexar contenido de los grafos. Podemos ver que dichas bases de datos aparecen clasificadas en el informe de Gartner y están resaltadas en óvalos rojos en la **Ilustración 14**.

Por otro lado y siguiendo la línea de análisis de bases de datos orientadas a grafos en el último informe de *Forrester* sobre bases de datos de grafos [34] vemos que las bases de datos mencionadas anteriormente son algunas de las líderes en su segmento, siendo Neo4j la más popular en la actualidad.

A continuación, se analizarán brevemente el paradigma de bases de datos orientadas a grafos, haciendo especial énfasis en las bases de datos NEO4J y Titan DB, así como en lenguajes de consulta optimizados para grafos CYPHER y GREMLIN debido a que son los seleccionados para ser usados en el presente trabajo; además se presentan las características y razones por las cuales dichas bases de datos fueron seleccionadas.

1.2.6 Bases de datos orientadas a grafos

Como su nombre lo indica, resuelven los problemas inherentes a grafos y teoría de grafos. Están preparadas para manejar la complejidad de operar grafos densos con una estructura de datos determinada, y cuentan con un almacenamiento que puede ser propietario o un sistema de terceros. Ejemplos de bases de datos de grafos son Neo4j, Allegrograph, Dex, FlockDB, InfiniteGraph, OrientDB, entre otras. Un listado de las bases de datos de grafos más relevantes en la actualidad según *db-engine* se puede ver en la **Tabla 2** [33].

Posición	Base de datos	Tipo
1	**Neo4j**	Graph DBMS
2	**Titan**	Graph DBMS
3	OrientDB	Multi-model
4	Virtuoso	Multi-model
5	ArangoDB	Multi-model
6	**Giraph**	Graph DBMS
7	**AllegroGraph**	Multi-model
8	Stardog	Multi-model
9	Sqrrl	Multi-model
10	InfiniteGraph	Graph DBMS
11	Sparksee	Graph DBMS
12	GlobalsDB	Multi-model
13	HyperGraphDB	Graph DBMS
14	InfoGrid	Graph DBMS
15	**FlockDB**	Graph DBMS
16	VelocityGraph	Graph DBMS
17	GraphDB	Multi-model
18	GraphBase	Graph DBMS
19	Amisa Server	Multi-model
19	Blazegraph	Multi-model

Tabla 2: Listado de las bases de datos de grafos listadas en *db-engine [33].*

Existen muchas otras bases de datos, implementaciones o APIs orientadas a grafos como por ejemplo la API graphx [35] basadas en Apache Spark que, aunque parecen muy prometedoras, su madurez y aceptación por la comunidad aun no son lo suficientemente

fuertes como para rivalizar con las anteriormente expuestas.

En el segmento de los grafos, se puede realizar una división de dos grandes grupos [26]:

1. **OLTP *(OnLine Transaction Processing):*** Tecnologías usadas principalmente para realizar transacciones en línea en grafos persistentes, típicamente accedidos en tiempo real por aplicaciones.
2. **OLAP (*online analytical processing*):** Tecnologías usadas principalmente para análisis de grafos *off-line*, típicamente ejecutadas como una serie de pasos en procesamiento por lotes *(steps batch)* llamados motores de computo de grafos (*graph compute engines*) (Ver **Ilustración 15**). Algunos ejemplos de estos sistemas son Cassovary, Pegasus o Giraph. Muchos de éstos sistemas se basan en el paper de Pregel presentado por Google (parte de la empresa ahora llamada Alphabet) en 2010 [36] el cual describe el *graph compute engine* que usa para calcular el *page Rank* de su buscador.

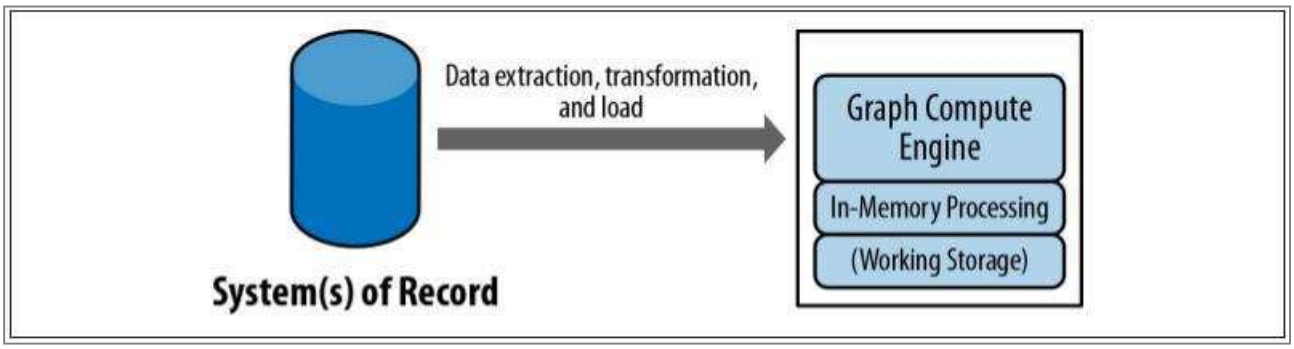

Ilustración 15: Funcionamiento de un motor de cómputo de grafos *[26]*.

Por otro lado, hay dos propiedades de las bases de datos de grafos que se deben tener en cuenta al usar dichas tecnologías (Ver **Ilustración 16**):

- **Almacenamiento subyacente:** Algunas bases de datos de grafos usan almacenamiento nativo para grafos optimizado y diseñado para guardar y manejar grafos. Otras bases serializan los datos del grafo en una base de datos relacional, en una base de datos orientada a objetos o en algún tipo de almacenamiento de datos de propósito general.
- **Motor de procesamiento:** son bases de datos de grafos que aplican lo que se conoce como índice libre de adyacencia (*index-free adjacency*), lo que significa que los nodos están conectados físicamente a otro nodo o "punto" en la base de datos [37]. Existen así, significativas ventajas de rendimiento al usar un índice libre de adyacencia, por tanto, se utiliza el término procesamiento gráfico nativo (*native graph processing*) para describir las bases de datos de grafos que lo aprovechan.

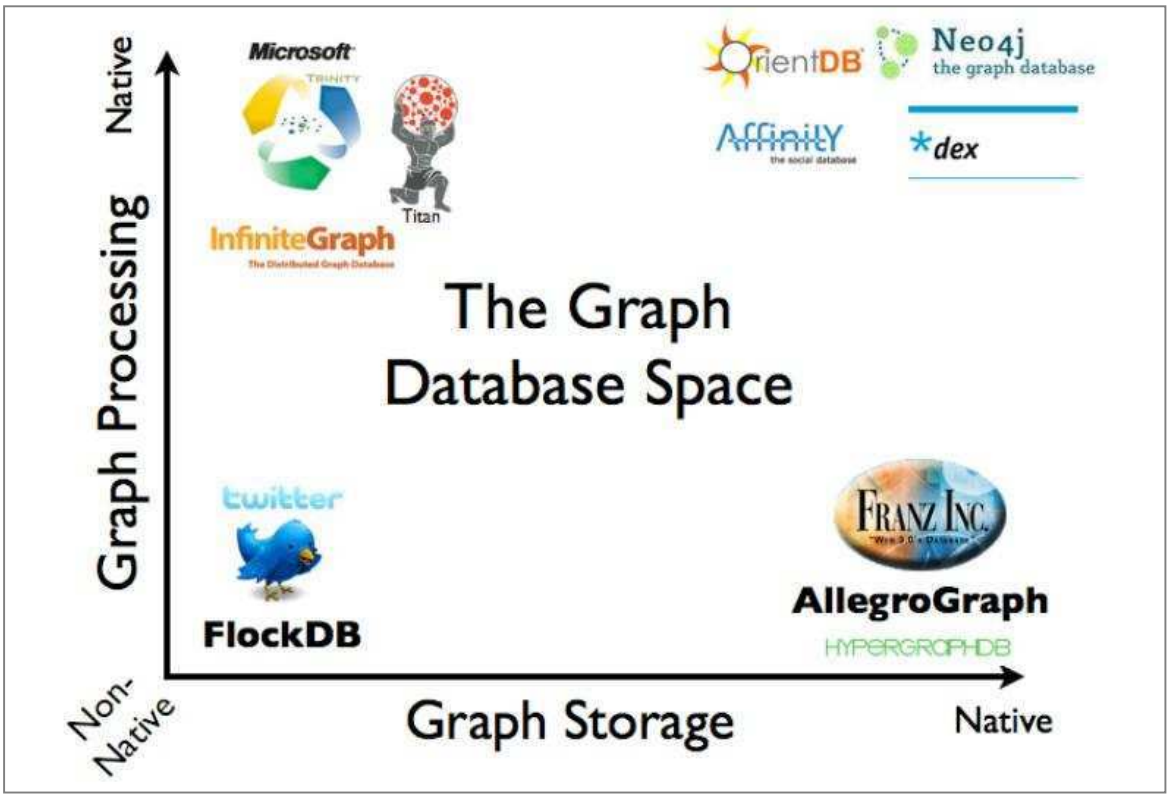

Ilustración 16: *Clasificación de bases de datos de grafos basados en su almacenamiento y modelo de procesamiento [38].*

En las bases de datos de grafos existe lo que se conoce como propiedades de un modelo de datos de grafos, lo que esencialmente plantea que para guardar datos en dichas bases de datos se deben usar vértices y aristas (nodos y relaciones) teniendo a su vez, cada una de éstas entidades atributos de la forma clave-valor (*key-value*), de esta forma, una estructura de grafos habilita a:

a) Representar datos de forma natural, sin las distorsiones de un modelo de datos relacional clásico.
b) Aplicar varios tipos de algoritmos de grafos en esas estructuras de manera optimizada.

Una de las capacidades más importantes de dichas bases de datos, es la llamada travesía del grafo (*traverse the graph)* que es la capacidad de mediante consultas "caminar sobre sus nodos y relaciones", saltando desde un nodo al siguiente siguiendo los punteros explícitos que conectan los nodos. Como ya lo hemos visto, esta capacidad a veces también se conoce como índice de adyacencia libre *(index free adjacency),* que esencialmente significa que se pueden encontrar nodos adyacentes o vecinos sin tener que hacer un índice de búsqueda, lo cual es clave para características de rendimiento necesarias para la ejecución de algoritmos sobre grafos [28].

Uno de los problemas más comunes en el mundo real en grafos con grandes cantidades de datos *(big data based graphs)* es el conocido como problema del súper nodo (*supernode problem*), el cual se da cuando un grupo de vértices tiene un alto número de aristas incidentes en él. Así, en teoría de grafos, dichos vértices se conocen como súper nodos. Estos nodos en general traen grandes problemas de rendimiento a las bases de datos de grafos, especialmente en consultas o algoritmos que busquen caminos entre vértices. Es por ésta razón que una de las características más importantes de una base

de datos de grafos orientada a *big data* es la capacidad para ejecutar algoritmos traversos a través de grafos con súper nodos [39].

Algunas bases de datos orientadas a grafos están preparadas nativamente para solucionar el problema del súper nodo mediante el uso de sistemas externos de búsqueda e indexación; en general dichos sistemas permiten realizar índices basados en Apache Lucene u otro con características similares o derivado de éste para optimizar consultas traversas a través de grafos altamente conectados con presencia de súper nodos.

De ésta forma, dichas bases de datos y sus propiedades del modelo de grafos (Ver **Ilustración 17**) están optimizados para:

a) Grafos dirigidos.
b) Grafos multirelacionados.
c) Guardado de pares clave-valor *(key-value)* como propiedades de los nodos y relaciones.

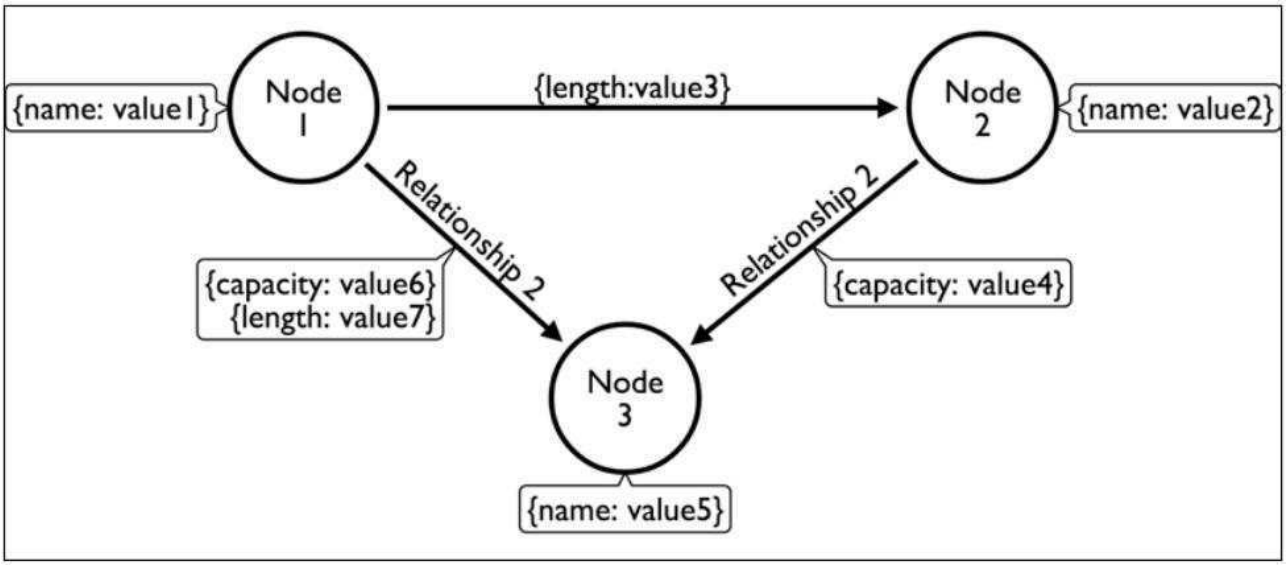

Ilustración 17: Estructura de grafos en una base de datos *NoSQL* orientada a grafos *[26].*

A continuación, se presentan las características más importantes de las bases de datos de grafos NEO4J y TITAN DB por las cuales fueron seleccionadas para las implementaciones planteadas en el presente trabajo:

1. Son líderes en el segmento de bases de datos [33, 34, 32].
2. Ambas representan diversas orientaciones de las bases de datos orientadas a grafos:
   a. **NEO4J**: Procesamiento de grafos con almacenamiento nativo, ACID.
   b. **TITAN DB**: Procesamiento de grafos con almacenamiento no nativo (Hadoop, Cassandra, Berkeley DB), ACID con posibilidades de modificaciones en el teorema CAP y propiedades BASE.
3. Permiten uso de motores de búsqueda adicionales para indexación en grandes volúmenes de datos todos ellos basados en *Apache Lucene* que permiten solucionar el *supernode problem* [39].
4. Ambas bases de datos usan el concepto de *index free adjacency.*
5. Presentan gran aceptación en el mercado y en el ámbito académico, demostrado esto mediante los múltiples casos de implementación y en la gran cantidad de documentación y soporte presente para cada una de ellas.
6. Vemos como en algunas comparativas como la presentada en [40], NEO4j obtiene

los mejores resultados en cargas de trabajo de procesos traversos (*traversal workloads*) que implican procesos de lectura, en tanto TITAN DB con *Cassandra* obtiene los mejores resultados en términos de cargas que implican lectura-escritura.

7. Ambas presentan una velocidad de evolución alta, al punto de que el presente trabajo se inicia con la versión de NEO4J 2.1.4 y para el fin del mismo existe la versión 3.0 beta disponible; de igual manera para Titan DB se inicia con la versión 0.4.4 y para la finalización del mismo existe la versión 1.0.0, convertida en un proyecto incubado por The Apache Software Foundation y luego todo el ecosistema de Titan DB fue adquirido por Datastax en 2015.

1.2.6.1 NEO4J

Es una base de datos de grafos creada en 2007 perteneciente a la empresa Neo Technologies [41]. Algunas de sus principales características son:

- Implementa CQL o *Cypher Query Languaje* que es un lenguaje fácil de usar semejante a SQL y orientado a grafos.
- Sigue el modelo de datos *Property Graph Data Model*.
- Soporta tanto índices nativos como índices usando Apache Lucene.
- Soporta restricciones de tipo UNIQUE entre otras.
- Cuenta con una interface de usuario para ejecutar comandos cypher y visualizar grafos.
- Garantiza transacciones ACID.
- Usa almacenamiento nativo de grafos con *Native GPE (Graph Processing Engine).*
- Soporta exportación de datos de consulta a formatos JSON, XLS entre otros.
- Provee una API de tipo REST para acceder desde cualquier lenguaje de programación tal como Java, Scala, etc.
- Provee una capa Javascript para acceder a cualquier UI MVC Framework como Node JS.
- Soporta dos tipos de API java: Cypher API y API de Java Nativo para desarrollo de aplicaciones embebidas en java.

Debido a las cargas de trabajo presentadas por un uso de tipo OLTP, Neo4j plantea una configuración de Alta Disponibilidad (HA, high availability) mostrada en la **Ilustración 18**, que permite soportar requerimientos de escalabilidad crítica, alta disponibilidad y tolerancia a fallos; esto lo logra creando clústeres de instancias de bases de datos que trabajan juntos para soportar la carga de trabajo [26].

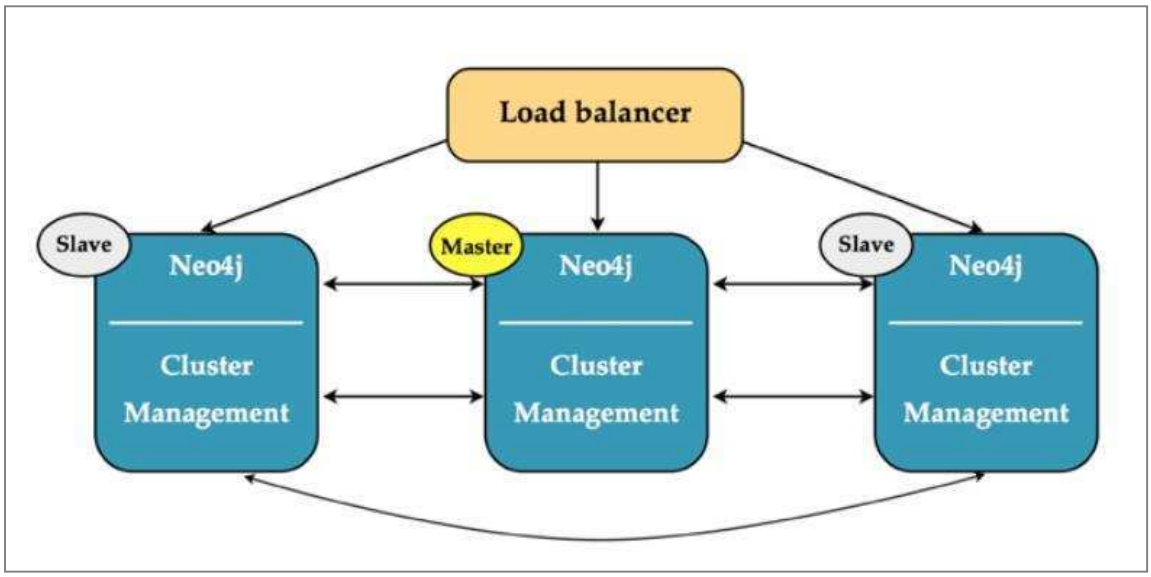

Ilustración 18: Arquitectura de alta disponibilidad (*HA*) de Neo4j *[26]*.

Neo4j ofrece dos diferentes ediciones de su base de datos de grafos:

5. **Community Edition:** es una base de datos de alto rendimiento básica, totalmente funcional.
6. **Enterprise Edition:** Adiciona un número de características a la Community Edition tales como *clustering* que permite alta disponibilidad y balanceo de carga (*load balancing*). Ver **Ilustración 18**.

Neo4j, además provee una herramienta de ejecución de consultas en Cypher y visualización de grafos llamada Neo4j Browser. Adicionalmente presenta un esquema de integración con herramientas de terceros cuyo diseño se puede ver en la **Ilustración 19**. Algunos ejemplos de integraciones con herramientas para visualización de grafos de terceros son:

- **Frameworks**: D3.js, Graphviz, Sigma.js, Vivagraph.js,
- **Herramientas**: Gephi, keylines, Linkurio.us, Tom Sawyer

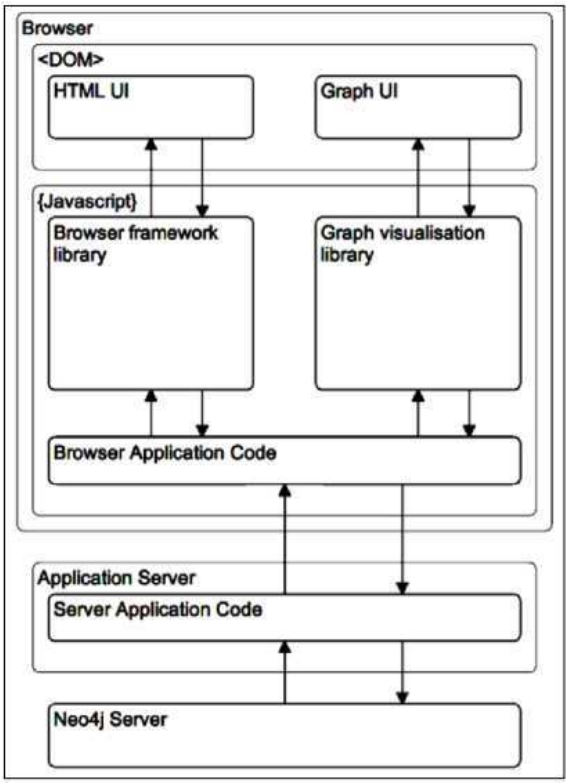

Ilustración 19: Diseño de integración con librerías externas para visualización de grafos en NEO4J *[26]*.

Como parte de la arquitectura de Neo4j, se presentan dos modos de ejecución:

1. **Base de datos embebida (*Embedded*):** en éste modo, la base de datos se ejecuta en el mismo proceso de la aplicación que la usa. Es ideal para dispositivos de hardware especializados y aplicaciones de escritorio. Entre algunas de las ventajas de éste modo se encuentra: baja latencia, posibilidad de elección de API de consulta (Ver **Ilustración 20**), transacciones explicitas entre otras. Sin embargo, al ejecutar éste modo se debe tener presente que la JVM de java es la que tiene en memoria las bases de datos, por lo que el GC (*Garbage Collector)* debe ser monitoreado constantemente, además que el ciclo de vida de la base de datos debe ser explicitado mediante código.

2. **Base de datos en modo servidor *(Server Mode*)**: permite ejecutar la base de datos en uno o varios servidores. Presenta beneficios como manejo a través de REST API, independencia de plataforma mediante el uso de formato JSON y comunicación HTTP, independencia de escalabilidad, aislamiento de la aplicación del comportamiento del GC de la JVM. De igual manera, al ejecutarse en un servidor, existen posibles extensiones o *server extensions* que brindan características adicionales programables mediante el API (Ver **Ilustración 20**).

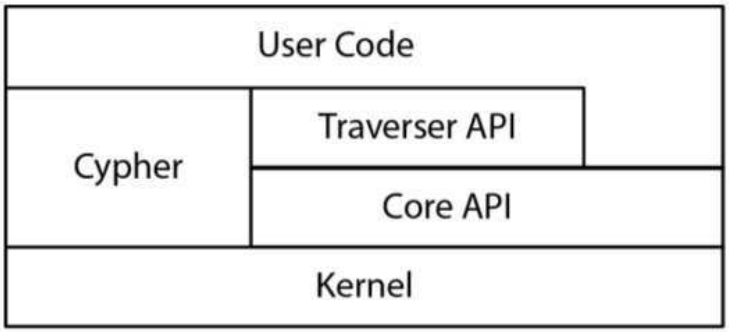

Ilustración 20: Vista lógica de la API en Neo4j *[38]*.

Neo4j interpreta de forma nativa consultas ejecutadas en Cypher, cuyo motor esta implementado internamente en el lenguaje *Scala*. Existen 4 formas para ejecutar consultas cypher en neo4j: Neo4j Shell (línea de comandos), Neo4j Web Admin Console (Interface Web), java (programáticamente) y REST (Sobre HTTP usando una interface REST). De igual forma, se puede tener acceso a la base de datos mediante el lenguaje de consultas de grafos gremlin, el cual forma parte del *tinkerpop stack* [42], nativamente implementado en Titan DB [43].

Finalmente, la última versión estable fue la 2.3.1 en ambas versiones: Community y Enterprise, lanzada en noviembre de 2015 al igual que su versión 3.0 M01 para probadores beta *(beta tester)*.

1.2.6.2 Titán DB

Es una base de datos de grafos distribuida creada en 2009, que hace parte del ecosistema Thinkaurelius [44], adquirida por la empresa Datastax (anuncio hecho en febrero 3 de 2015) quien también distribuye y soporta Cassandra Enterprise. En dicho ecosistema, están presentes el sistema Faunus (*Graph Analytics Engine*) y Fulgora *(Fast Graph Processor*) (Ver **Ilustración 21**).

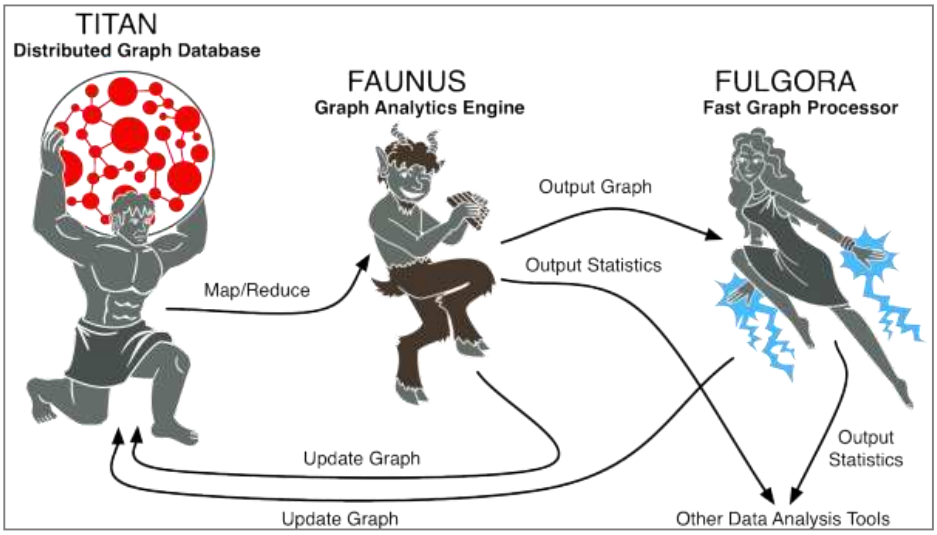

Ilustración 21: Tecnologías alrededor del proyecto *Thinkaurelius [44].*

Titan es escalable y optimizada para guardar y ejecutar consultas de grafos que contiene cientos de billones de vértices y aristas distribuidos en clústeres multi-máquina. Titan es

una base de datos transaccional que soporta miles de usuarios concurrentes y permite ejecutar complejos grafos traversos en tiempo real. Provee las siguientes características [45]:

- Escalabilidad lineal y elástica para crecimiento de datos y usuarios base.
- Distribución y replicación de datos para manejo de performance y la tolerancia a fallos.
- Alta disponibilidad mediante multi-centro de datos y copias de seguridad en tiempo real *(hot-backup).*
- Soporta ACID y consistencia eventual (CAP) (Ver **Ilustración 22**)
- Soporta varios sistemas de almacenamiento de terceros (Ver **Ilustración 22**):
    - Apache Cassandra.
    - Apache HBase.
    - Oracle BerkeleyDB.
- Brinda soporte para análisis de datos de grafos globales, reportes y ETL a través de la integración con plataformas de *big data* tales como:
    - Apache Spark.
    - Apache Giraph.
    - Apache Hadoop.
- Soporta búsquedas de tipo *full-text*, *geolocalización* y rango numérico vía herramientas de terceros como:
    - ElasticSearch.
    - Solr.
    - Lucene.
- Integración nativa con el conjunto de productos de grafos de TinkerPop [46] (Ver **Ilustración 23**):
    - Gremlin graph query language.
    - Gremlin graph server.
    - Gremlin applications.
- Licencia Open source de tipo Apache 2.

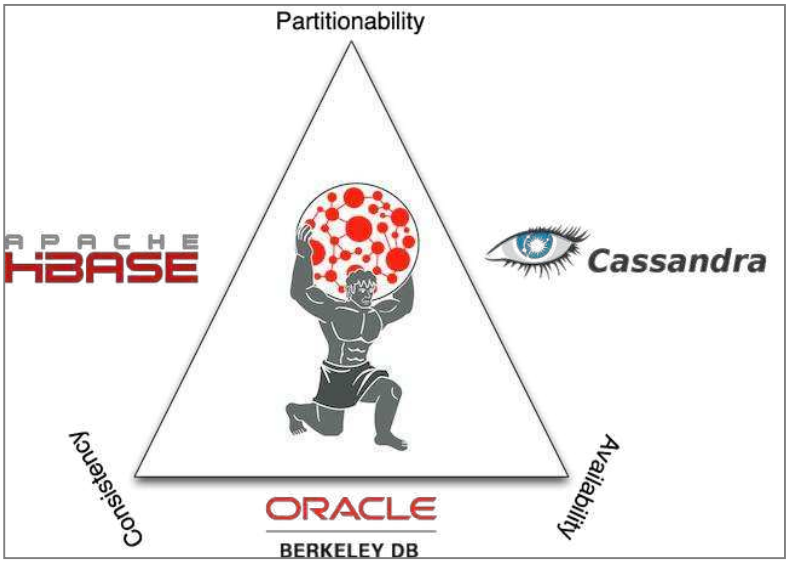

Ilustración 22: Titan DB y sus posibilidades de consistencia eventual aplicando el teorema CAP gracias al uso de distintas bases de almacenamiento de datos *[44]*.

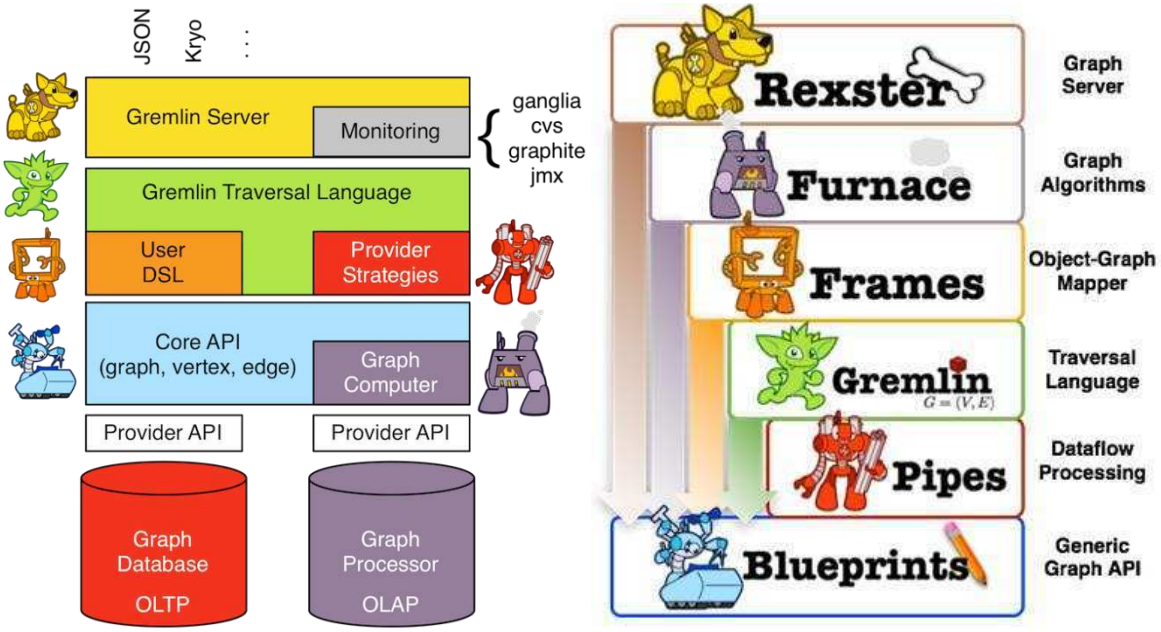

Ilustración 23: Ecosistema *Apache Tinkerpop 3.0 [46].*

La arquitectura conceptual de Titan DB es la mostrada en la **Ilustración 24**; en ella se pueden ver las características antes mencionadas, además de sus capas a nivel lógico, e indirectamente a nivel físico.

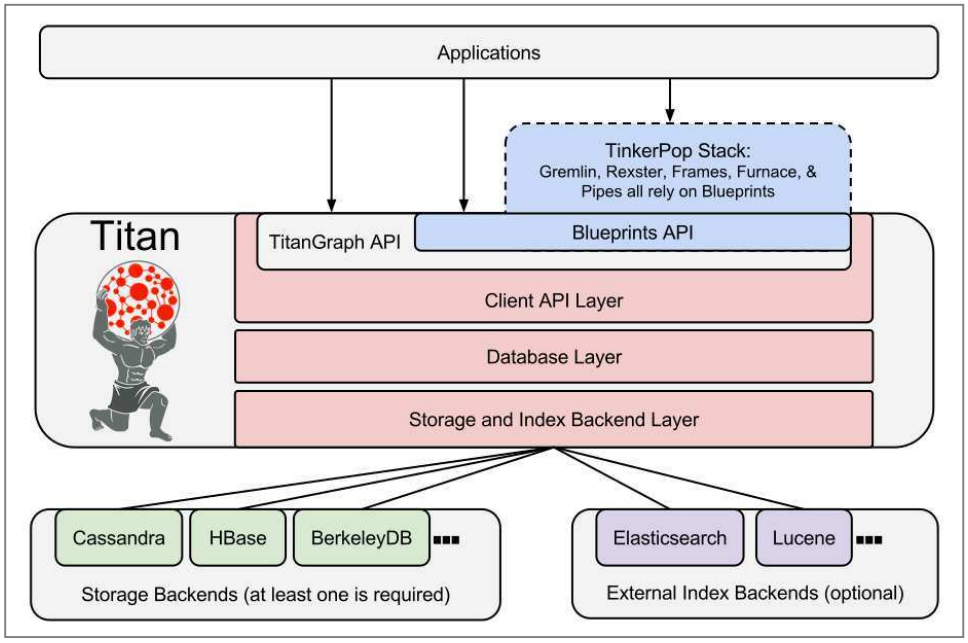

Ilustración 24: *Arquitectura de la base de datos TITAN [44] .*

La última versión registrada de Titan DB es la 1.0.0 que fue liberada en septiembre de 2015.

1.2.7 RDBMS Versus bases de datos de grafos

Como una generalización, las bases de datos de grafos se ocupan de las relaciones, mientras que los RDBMS se preocupan por los datos, teniendo así dificultades con el complejo análisis de la teoría de grafos. Es fácil manejar un grafo donde cada camino posible tiene una longitud de uno, simplemente una tabla de tres columnas (nodo, relación, nodo). Haciendo autocombinaciones, se puede construir caminos de longitud dos, y así sucesivamente, lo cual es llamado búsqueda en amplitud *(breadth-first search*). Sin embargo, se obtienen rápidamente explosiones cartesianas para trayectos más largos, y las consultas puede perderse en un bucle sin fin si hay ciclos. Por otra parte, es muy difícil escribir consultas SQL para el análisis de grafos si las longitudes de trayectoria son largas, teniendo en cuenta, además, que en ciertas ocasiones dicho tamaño es desconocido y variable [25].

Algunas comparativas entre bases de datos relacionales y bases de datos orientadas a grafos como la planteada en [47] presenta resultados a nivel experimental en donde las segundas presentan mejores resultados que las primeras, avalando lo ya presentado sobre la conveniencia de uso de bases de datos orientadas a grafos en vez de bases de datos relacionales cuando se trabajan con datos modelados como redes o grafos.

1.2.8 Lenguajes de consultas orientados a grafos

A medida que fueron apareciendo bases de datos orientadas a grafos, también se generaron lenguajes de consultas optimizados para grafos, en la mayoría basados en lenguajes populares como SQL o lenguajes surgidos de intentos de estandarizaciones como SPARQL.

Como se puede observar en algunas comparativas como la planteada en [48], el comportamiento de lenguajes de consulta optimizados para grafos versus consultas mediante SQL o acceso nativo a los datos, puede varias según el tipo de consultas realizadas, ya que cada una está optimizada para determinadas operaciones. En general se observan que en promedio los lenguajes como cypher y gremlin presentan mejores resultados respecto a otros medios de consulta en operaciones de tipo traversa a pesar de que en algunos casos el acceso nativo resulta más rápido, con el efecto negativo de que el código que lo genera tiene un costo de mantenimiento no existente en consultas de lenguajes optimizados para grafos que hacen más simples e intuitivas las consultas, incluso comparándolo con consultas SQL sobre estructuras relacionales que representan grafos.

A continuación, mencionaremos algunos de los lenguajes orientados a grafos más representativos, enfocando la descripción en dos de los usados en el presente trabajo: CYPHER y GREMLIN.

1.2.8.1 SPARQL y SPASQL

SPARQL es un acrónimo recursivo para *SPARQL Protocol and RDF Query Languaje*. Es un lenguaje de consulta para formato RDF *(Resource Description Framework)* o marco de descripción de recursos que hace parte de la *W3C Semantic Web* (www.w3.org). Es semejante al SQL y usa palabras reservadas parecidas al lenguaje C con algunos

caracteres de tipo ASCII y cálculos lambda.

SPASQL es una extensión del SQL estándar el cual a su vez permite ejecución de consultas SPARQL, típicamente tratados como subconsultas o cláusulas de función. Igualmente permite consultas SPARQL a través de *APIs* de acceso a datos tradicionales como ODBC, JDBC, OLE DB, ADO.NET, entre otras.

1.2.8.2 CYPHER

Cypher es un lenguaje declarativo de grafos que está en proceso de crecimiento y maduración, siendo muy aceptado por programadores que han usado SQL con anterioridad. Forma parte de la base de datos de grafos NEO4J. Algunas de sus palabras reservadas, como WHERE y ORDER BY, son inspiradas en SQL, mientras que la coincidencia de patrones toma prestado enfoques y expresión de SPARQL. Entre las clausulas más usadas se encuentra [25] [49]:

- **START**: punto de inicio en el grafo, obtenido a partir de *index lookups* o ID del elemento.
- **MATCH**: patrón para encontrar resultados coincidentes.
- **WHERE**: criterios de filtros.
- **RETURN**: atributos o elementos que retorna la consulta.
- **CREATE**: crea nodos y relaciones.
- **DELETE**: remueve nodos, relaciones y propiedades.
- **SET**: asigna valores a propiedades.
- **FOREACH**: realizar acciones de actualizaciones para un conjunto de elementos de una lista, uno por uno.
- **WITH**: divide a una consulta en múltiples y distintas partes.

En Cypher existen cuatro características que lo definen [26]:

1. **Declarativo (*Declarative*):** Cypher es un lenguaje de consultas declarativo, el cual es muy diferente a alternativas de tipo imperativo. Se declara el patrón que se está buscando, es decir, “se le dice” a cypher que se quiere y no como buscar lo que se quiere. Un ejemplo clásico de lenguaje declarativo es SQL (*Structured Query Language*).
2. **Expresividad (*Expressive*):** Cypher es un lenguaje altamente expresivo, con múltiples palabras claves, funciones y utilidades que enriquecen las consultas de una manera muy amplia.
3. **Coincidencia de patrones (*Pattern Matching*)**: Cypher es un lenguaje de consulta para búsqueda de coincidencia de patrones en grafos.
4. **Idempotente (*Idempotent*):** significa que cuando se opera una función (o se ejecuta una consulta) en un algoritmo, los cambios en los datos deberían solo ocurrir en la primera ejecución de la función. Así, múltiples ejecuciones de la misma función sobre los mismos datos no deberían ejercer ningún efecto adicional.

En 2015 fue lanzado el proyecto OpenCypher por Neo Technologies; su objetivo es desarrollar un estándar en la industria de lenguaje de consulta para grafos de tipo Open Source basado en Cypher.

1.2.8.3 Gremlin

Es un lenguaje Open Source basado en propiedades de grafos traversos. Toma sintaxis de la familia de lenguajes de Orientación a Objetos y Programación en C. Tiene sintaxis para aristas dirigidas y más consultas complejas que lucen más como fórmulas matemáticas que consultas SQL regulares [25].

Este lenguaje es parte del desarrollo del proyecto Tinkerpop 3.0 (Ver **Ilustración 23**) que a su vez hace parte del ecosistema *Eurelius Project* [50].

Gremlin puede ejecutarse sobre un sistema de cómputo de grafos de tipo OLTP u OLAP. Provee característica que lo habilitan para soportar tanto consultas de tipo imperativo como declarativo.

Está compuesto por aproximadamente 30 funciones llamadas *steps* que son de tipo traverso, siendo diez las que se aplican en la mayoría de los casos. Existen entonces cinco tipos de estas funciones: *map, flatMap, filter, sideEffect y Branch*. Por otro lado, existe el concepto de estrategias traversas, las cuales son de cinco tipos: *decoration, optimization, vendor-optimization, finalization y verification* [50].

Finalmente, la última versión de Gremlin está escrita en Java 8. Existen algunas variantes del lenguaje como Gremlin-Groovy (que hace parte de Apache TinkerPop3), Gremlin-Scala, Gremlin-JavaScript, Gremlin-Clojure (conocido como Ogre) entre otros.

1.3 BIOINFORMÁTICA Y *Mycobacterium tuberculosis*

1.3.1 Mycobacterium tuberculosis

La tuberculosis es una enfermedad infecciosa causada por el bacilo *Mycobacterium tuberculosis*. Típicamente afecta los pulmones, pero puede afectar otras áreas. La enfermedad se adquiere por vía aérea cuando las personas que están enfermas con tuberculosis pulmonar expulsan las bacterias, por ejemplo, por la tos. En general, una proporción relativamente pequeña de las personas infectadas con *M. tuberculosis* desarrollarán la enfermedad. Sin embargo, la probabilidad de desarrollarla es mucho mayor entre las personas infectadas con el virus de la inmunodeficiencia humana (VIH). *M. tuberculosis* también es más común entre los hombres que entre las mujeres, y afecta principalmente a los adultos en los grupos de edad económicamente más productivos.

Los tratamientos fármacos eficaces se desarrollaron por primera vez en la década de 1940. El medicamento más eficaz de primera línea contra la tuberculosis, la *rifampicina*, estuvo disponible en la década de 1960. El tratamiento actualmente recomendado para los nuevos casos de tuberculosis sensible a los antibióticos es un régimen de seis meses de los cuatro fármacos de primera línea: *isoniazida, rifampicina, etambutol y pirazinamida*. Las tasas de éxito de tratamiento son del 85%. El tratamiento para la tuberculosis multiresistente (MDR- TB), que se define por la existencia de resistencia a la *isoniazida* y la *rifampicina* (los dos medicamentos antituberculosos más potentes) es más largo, y requiere de medicamentos más caros y más tóxicos. Para la mayoría de los pacientes con MDR -TB, los regímenes actuales recomendados por OMS tiene duración de 20 meses, y la tasa de tratamientos exitosos son mucho más bajas [5].

En Europa y en los EE.UU, la mejora general de la salud pública ayudó a reducir la cantidad de infecciones de *M. tuberculosis* mucho antes de la llegada de medicamentos específicos. Las actividades del programa de tuberculosis, reforzados por la quimioterapia exitosa, se tradujeron en una reducción pronunciada de las tasas de infección y muerte. La enfermedad fue en gran medida controlada pero nunca desapareció. Luego, en torno a 1985, los casos de *M. tuberculosis* comenzaron a subir de nuevo en países industrializados. Varias fuerzas interrelacionadas impulsaron este resurgimiento, incluyendo aumento de la población carcelaria, la falta de vivienda, el uso de drogas inyectables, hacinamiento y el aumento de la inmigración procedente de países donde la tuberculosis sigue siendo endémica. Por encima de todo, la disminución de las actividades de control de la tuberculosis y el incremento del síndrome de inmunodeficiencia adquirida (SIDA) causado por el VIH fueron y son en la actualidad los dos factores principales que alimentan el resurgimiento de la tuberculosis.

De hecho, la epidemia del VIH / SIDA ha producido un efecto devastador en la lucha contra la tuberculosis en todo el mundo. Mientras que uno de cada diez personas inmuno-competentes infectados con *M. tuberculosis* caerán enfermos durante su vida, entre las personas con infección por VIH, uno de cada diez al año desarrollarán tuberculosis activa. En los países en desarrollo, el impacto negativo especialmente en el grupo de 20-35 años de edad, es especialmente grave [51].

Se estima que en la actualidad un tercio de la población mundial está infectada con el bacilo de tuberculosis, constituyendo un reservorio a partir del cual se producirán futuros casos. Sin tratamiento, las tasas de mortalidad son altas.

La infección por *M. tuberculosis* consta de cinco etapas de desarrollo de la infección:

1. **Inicio:** Inhalación o adquisición de infección.
2. **Simbiosis:** Inicio de 7 a 21 días después de la infección localizando las bacterias en los pulmones.
3. **Necrosis caseosa:** Se inicia la infiltración de linfocitos liberando IFN para activar los macrófagos que son capaces de destruir la *M. tuberculosis*. Se presenta entones la presencia de anticuerpos a pesar que no controlan la infección. Así, la respuesta inmune celular y las *citoquinas* liberadas (*IFNy, IL1, TNF*) son las responsables de la patología asociada con esta etapa: presencia de tubérculos.
4. **Activación macrófagos:** El tubérculo crece y puede invadir los bronquios y conexiones hemáticas. Posteriormente se puede diseminar a sitios extra pulmonares provocando tuberculosis Miliar (sistema genitourinario, huesos, articulaciones, nódulos linfáticos, peritoneo).
5. **Licuefacción y cavitación:** Licuefacción del centro del tubérculo, fibrosis y calcificación del pulmón.

Para mantener los sistemas complejos que apoyan el mecanismo patogénico ya mencionado, la bacteria necesita mecanismos de regulación igualmente complejos, siendo los multígrafos mixtos una forma apropiada y orgánica para representarlos.

En los últimos años, el conocimiento de *M. tuberculosis* ha ido en aumento, generando gran cantidad de estudios de transcripción global y análisis de interacciones proteína-proteína, entre otros. Por estos motivos, es pertinente aplicar técnicas de análisis conjunto de información, como lo muestra el presente trabajo.

1.3.2 ANÁLISIS BIOLÓGICOS

A continuación, se mencionarán los tres principales análisis biológicos relacionados con el presente trabajo ya que corresponden a los realizados para obtener los datos usados en la sección experimental para construir el multigrafo mixto.

1.3.2.1 Interacción proteína-proteína

Las proteínas controlan y median en la gran mayoría de los procesos biológicos en una célula viva. Actúan como catalizadores, transportadores, proporcionan resistencia mecánica, confieren inmunidad, transmiten señales, y permiten el crecimiento y desarrollo de control. Las proteínas son polímeros de aminoácidos, unidos covalentemente a través de enlaces peptídicos. La función de una proteína está determinada por su estructura tridimensional, que a su vez se define por su secuencia de aminoácidos [1].

Las interacciones proteína-proteína (PPIs) juegan un rol clave en varios aspectos de la regulación celular, siendo así un análisis crucial para el entendimiento de procesos biológicos.

Detección de interacciones proteína-proteína (PPIs) [14]: La calidad y cantidad del material inicial tiene la mayor importancia para todas las aproximaciones. Algunos de los métodos clásicos son aquellos basados en interacciones entre moléculas:

- ***Cromatografía de afinidad (Affinity chromatography)*:** puede proceder a partir de extractos crudos de proteínas, pero, por lo general, sólo se pueden recuperar una cantidad mínima de interacciones.
- ***Coinmunoprecipitación (Coimmunoprecipitation)***: requiere cantidades relativamente grandes de anticuerpos y el posterior aislamiento del complejo puede reducir el rendimiento del procedimiento.

Un método posterior basado en espectrometría de masas ha dado lugar a un renacimiento de los métodos bioquímicos para la identificación de proteínas complejas debido a su superior sensibilidad, velocidad y versatilidad en comparación con los métodos de secuenciación de proteínas tradicionales. Actualmente existen varias estrategias experimentales para general un mapa de interacción proteómico de alto rendimiento, dos de las más comunes son:

a) **Y2H** (Yeast Two-Hybrid system).
b) **AP-MS** (*affinity purification coupled to mass spectrometry*): Análisis de proteínas complejas por afinidad acoplada a espectrometría de masas.

Existen otros caminos para identificar la interacción proteína-proteína:

- Métodos computacionales para predecir interacción proteína-proteína.
- Microarreglos de proteínas (*Protein Microarrays*).

1.3.2.2 Análisis de las interacciones entre regiones promotoras y proteínas

Para la correcta expresión de un gen, es decir, la transcripción de su información genética en una molécula de ARN mensajero, en el momento y cantidad adecuados y en las circunstancias biológicas que lo requieran, se necesita que una o más proteínas se unan a la región regulatoria de dicho gen. Estas proteínas regulatorias pueden actuar activando

o inhibiendo la expresión génica y reconocen secuencias específicas de nucleótidos que se encuentran en la región regulatoria, también llamado región promotora; otras veces, existen proteínas también con función regulatoria que se unen reconociendo proteínas que ya están unidas a su región especifica de ADN. Gracias a ésta riqueza de interacciones la célula puede controlar la expresión génica y responder a situaciones de estrés a la presencia de drogas tanto inhibitorias como antibióticas, y a la respuesta inmune del hospedador, entre otras [14].

Para la detección de interacciones entre proteínas y regiones regulatorias en el ADN existen diferentes técnicas. Una de ellas consiste en clonar secuencias de ADN de regiones promotoras en plásmidos donde estas secuencias quedan "rio arriba" de un gen "*reporter*" y en paralelo se clonan en vectores de expresión genes informados como reguladores. Se co-transforman ambas construcciones en células bacterianas y se siembran en medios de cultivo donde solo sobreviven aquellas donde el gen regulador pudo interactuar exitosamente con la región promotora [8].

Por otro lado, el sistema ChIP-on-chip o ChIP-chip [52, 53], es una tecnología que combina técnicas de inmunoprecipitación de la cromatina *chromatin immunoprecipitation ("ChIP")* con el uso de microarreglos de ADN ("*chip*") para la detección de interacciones entre proteínas y regiones regulatorias en el ADN. Consiste en los siguientes pasos:

1. Se induce la unión específica de proteínas a secuencias de ADN genómico.
2. Las secuencias de ADN se cortan en pequeños fragmentos por ultrasonicación. Luego se agregan anticuerpos que se unen a las proteínas de los complejos DNA-proteína. y se extraen los complejos AND-proteína-anticuerpo.
3. Se separan los fragmentos de ADN de los complejos.

Se amplifican los fragmentos de ADN y se determina su concentración e identidad por hibridación contra microarreglos o por secuenciación.

1.3.2.3 Co-expresión génica

Para llevar a cabo los diferentes procesos biológicos que cualquier célula necesita para reproducirse, crecer y sobrevivir a condiciones adversas, se deben activar y reprimir grupos específicos de genes; estos procesos de expresión coordinada se pueden detectar utilizando técnicas de microarreglos o con frecuencia creciente con técnicas de secuenciación masiva (*RNA-Seq*); mediante estas técnicas se pueden detectar las variaciones de expresión para todos los genes codificantes del genoma; cuando estos análisis se realizan simultáneamente para diferentes tratamientos se puede determinar una matriz de correlación de expresión génica. A partir de esta matriz se pueden determinar genes que comparten patrones de expresión bajo diferentes situaciones. Por ejemplo, se puede observar que los diferentes genes de una misma vía metabólica muestran patrones de expresión similares [1].

1.3.2.3.1 Matrices de adyacencia

Cualquiera de los datos obtenidos por algunos de los análisis biológicos mencionados anteriormente pueden ser representados mediante una matriz de adyacencia como los mencionados en el subcapítulo 5.5 y a partir de ellos construir grafos, especialmente los correspondientes al conjunto de datos 4 (grafo no dirigido 4, numeral 5.5.2.4) y al conjunto de datos 5 (grafo no dirigido 5, numeral 5.5.2.5).

1.3.2.3.2 Método C3NET

Los algoritmos de redes de inferencia de genes o GNI (*Gene network inference*) permiten a los investigadores explorar las interacciones entre los genes y los productos génicos. El principal proceso de los algoritmos GNI es obtener las puntuaciones (*score*) de asociación entre los genes para posteriormente poder construir matrices de adyacencia presentando dicha asociación entre genes como un grafo. Aunque hay varios estimadores de asociación utilizados en diferentes aplicaciones, no hay un estimador comúnmente aceptado como el mejor para las GNI. El algoritmo C3NET [54], comparado con otros métodos [55] es el que mejor rendimiento tiene, haciendo uso del estimador de entropía basado en información mutua (correlación de *Spearman*) [56]. Es por ésta razón que en el presente trabajo se utilizará dicho algoritmo y estimador para encontrar una matriz de adyacencia para los dos conjuntos de datos de co-expresión génica analizados en los puntos 3.5.2.4 y 3.5.2.5 del presente trabajo.

1.3.2.3.3 Coeficiente de variación

Es el radio de la desviación estándar σ y la media μ [57]:

$$C_V = \frac{\sigma}{\mu}$$

Originalmente fue propuesta como un método de comparación de la variabilidad de diferentes distribuciones. En biología, se usa frecuentemente en análisis de resultados de experimentos de microarreglos, en donde su aplicación permite identificar los valores de variación entre varias muestras con respecto a la expresión de un mismo gen [1].

1.4 BASES DE DATOS BIOLÓGICAS PARA *M. tuberculosis*

Existen en general grandes bases de datos públicas orientadas a disponibilizar datos biológicos tales como NCBI-UniGene, UniProt, Gene Disease Database entre una larga lista categorizada de acuerdo a los datos que contiene. A continuación – y dado el contenido del presente trabajo – describiremos brevemente las dos bases de datos más importantes que contienen datos de *M. tuberculosis*: Tuberculist y TBDB.

1.4.1 Tuberculist

Es una base de datos relacional que presenta información derivada principalmente de la cepa H37Rv de *M. tuberculosis*.

Siendo una base de conocimiento, Tuberculist integra detalles de genoma, información de proteínas, drogas, datos transcriptómicos, mutantes, anotaciones y operones, además de bibliografía, vistas estructurales y comparativas genómicas. Toda la información provista es estructurada de la manera requerida para un desarrollo racional de nuevas medidas de diagnósticos, terapéuticos y profilácticos contra la tuberculosis [12].

Tanto la base de datos como el sitio web del proyecto Tuberculist fueron creados en el

año 2000 y mantenidos en los laboratorios del profesor Stewart Cole, parte de *Global Health Institute* en el *Ecole Polytechnique Fédérale de Lausanne* en colaboración con el Instituto de Bioinformática de Suiza.

Actualmente se disponibiliza a través de la página web http://tuberculis.epfl.ch, siendo de gran valor para la comunidad de investigación con 75.000 visitas por mes.

Su última versión fue lanzada en marzo de 2013 correspondiente a la 2.6, *release* 27 y cuenta con las siguientes cifras en su contenido de información:

- 4.411.532 bp de secuencias de DNA que representan a todo el cromosoma de *M. tuberculosis* (cepa H37Rv).
- 4.018 genes que codifican proteínas.
- 13 pseudogenes.
- 45 genes que codifican para tRNAs.
- 3 que codifican para rRNA.
- 30 ncRNA genes.
- 2 miscRNA genes.

1.4.2 TBDB

Tuberculosis Database (TBDB) es una base de datos en línea que provee acceso integrado a secuencias de genoma, datos de expresión, y datos de literatura para tuberculosis. Actualmente tiene 20 sepas distintas de *M. tuberculosis*. Los datos integrados incluidos son:

- Secuencias genómicas y publicaciones asociadas disponibles para *M. tuberculosis*.
- Secuencias genómicas para cerca de 20 cepas relacionadas a MTB incluidas *M. africanum*, *M. avium*, *M. leprae*, y *M. smegmatus*.
- Secuencia global de datos de polimorfismos para *M. tuberculosis*.
- Anotaciones de secuencia para todo el genoma, incluido genes, proteínas, RNA y epítopos.
- Información estructural de proteínas.
- Anotaciones funcionales incluyendo funciones de enzimas, reacciones y vías metabólicas y anotaciones ontológicas GO.
- Datos de expresión génica - incluyendo datos procesados y sin procesar - para M. tuberculosis incluyendo más de 3.000 microarreglos y 95 conjuntos de datos de tipo RT-PCR.
- Datos de expresión génica de 2.700 microarreglos para humanos y ratones relativos a experimentos de *M. tuberculosis*, además de 260 arreglos de *Streptomyces coelicolor.*
- Literatura verificada sobre 2.656 genes y 45 conjuntos de datos de expresión génica.

TBDB provee acceso a los datos a través de integración con motores de búsqueda y un conjunto de herramientas de visualización, análisis y adquisición de datos [13] [58].

1.5 PLATAFORMA DE INTEGRACIÓN SEMÁNTICA DE DATOS BIOLÓGICOS: BIO4J

Como me expone en [59], Bio4j es una plataforma para la integración semántica de datos

biológicos enriquecidos usando modelos tipados de grafos. Contiene un modelado de datos públicos relacionados basados en proteínas y conformado por un conjunto de grafos interdependientes (Ver **Ilustración 25**). Las consultas de datos son posibles a través de un modelo de datos y un lenguaje de dominio específico implementado en java (Angulillos). Su distribución es compatible con almacenamiento en la nube y se usa la base de datos orientada a grafos Titan DB.

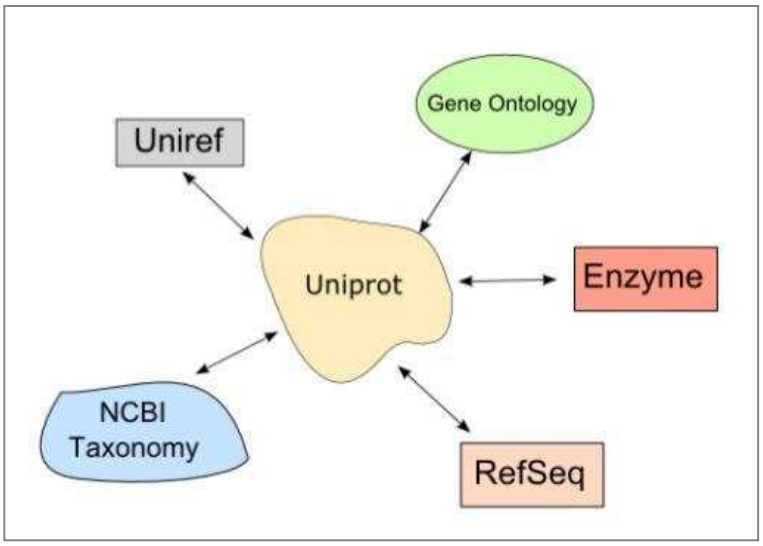

Ilustración 25: Conjuntos de datos usados en BIO4J. (Imagen Tomada de http://bio4j.com).

El objetivo general de Bio4j es brindar un recurso a investigadores en bioinformática presentando soluciones a problemas claves de: integración de datos, expresividad (caminos para acceder a la estructura de datos fácilmente, lo cual en general se traduce en expresividad del lenguaje de consulta usado para acceder al modelo planteado), acceso a datos con alto rendimiento y un despliegue de modelo escalable con bajo costo y alta efectividad.

Una de las características más importantes de Bio4j es la integración de datos públicos de diferentes fuentes: UniProtKB (SwissProt + Trembl), Gene Ontology (Go), UniRef (50, 90,100), NCBI Taxonomy, Req Seq y Expasy EnzymeDB (Ver **Ilustración 26**).

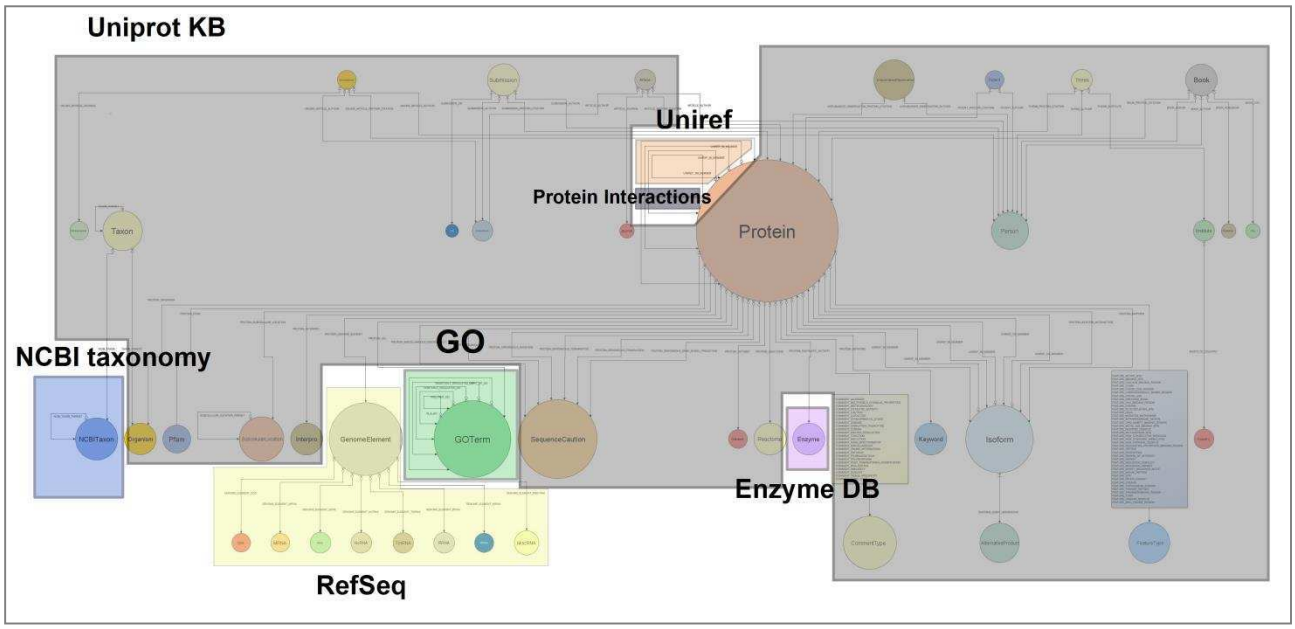

Ilustración 26: Modelo de datos de BIO4J. (Imagen Tomada de http://bio4j.com).

El proyecto Bio4j, pasó por amplias modificaciones; una de las más importantes fue el cambio de base de datos orientada a grafos de NEO4J a TitanDB. El último conteo de los componentes de la base de datos fue:

- 1.216.’993.547 relaciones.
- 190.625.351 nodos.
- 584’436.429 propiedades.

Como se puede ver, el enfoque de Bio4J es totalmente distinto a las clásicas bases de datos orientadas a datos biológicos, en general de tipo relacional y no enfocadas a grafos. A su vez, es una base de datos que abarca gran cantidad de información biológica de tipo más general. Por ejemplo, mucha de la información que almacena Tuberculist o MTDB no está presente en Bio4j. Así, plantear una base de datos de grafos orientada a *M. tuberculosis H37Rv* enfocada a datos específicos siguiendo algunos de los lineamientos de Bio4J es uno de los resultados del presente trabajo.

2. OBJETIVOS

1. Explorar la utilidad del uso de multigrafos mixtos y bases de datos no relacionales orientadas a grafos en biología.

2. Verificar que mediante el uso de multigrafos mixtos que integren información de expresión génica, de interacciones proteína-proteína y de interacciones proteína-ADN es posible descubrir redes regulatorias de procesos en biología molecular.

3. Desarrollar y evaluar técnicas de análisis de cliques y quasi-cliques en multigrafos para el descubrimiento de redes regulatorias.

3. HIPÓTESIS GENERAL

Las preguntas biológicas que involucran integrar información de gran volumen y de diferente tipo de experimentos biológicos, se pueden resolver con consultas y métricas especiales para quasi-cliques en multigrafos mixtos.

4. ORGANIZACIÓN DE LA TESIS

El presente trabajo está dividido en cuatro partes. En la primera, se realiza una introducción y planteamiento teórico de los conceptos usados; posteriormente en la segunda se definen los objetivos, la hipótesis general y la estructura global. En la tercera parte, se listan los materiales y métodos, describiendo los datos a usar, planteando el diseño de los grafos y el multigrafo y especificando la importación de los datos a dos bases de datos no relacionales orientadas a grafos. En la cuarta parte, correspondiente a la parte final del trabajo se plantean algunas definiciones sobre quasi-cliques en multigrafos mixtos y se exponen patrones biológicos mediante siete prototipos generales de consultas en el multigrafo mixto construido, presentando finalmente las conclusiones y trabajo futuro subyacentes.

5. MATERIALES Y MÉTODOS

5.1 HERRAMIENTAS

Para el desarrollo del presente trabajo se usaron máquinas ejecutando sistema operativo Linux Ubuntu 14.04 LTS / 3.13.0-43-generic GNU/Linux en las cuales se instalaron dos tipos de bases de datos no relacionales orientadas a grafos:

- **Neo4j Enterprise 2.3.1:** Versión lanzada el 11 de noviembre de 2015. Ejecutada sobre un sistema UNIX java 1.8.0_66.
- **Titan 1.0.0**: Se elige el sistema de almacenamiento Cassandra 2.1.9 basado en el teorema CAP [25] aplicado a la arquitectura planteada por Titan DB (Ver **Ilustración *22***), en donde dicho motor de base de datos permite las velocidades más altas de respuesta de las consultas orientado a un OLTP *(OnLine Transaction Processing)*. Para el procesamiento de índices se usó el motor de búsqueda Elasticsearch 1.5.1; todo el software mencionado se ejecutó sobre un sistema tipo UNIX en java 1.8.0_66. Adicionalmente, la versión 1.0.0 de Titan, lanzada el 23 de septiembre de 2015 trae nativamente el lenguaje Gremlin con soporte para *groovy* sobre el encapsulado TinkerPop 3.0.1-incubating.

Para el análisis de los conjuntos de datos se usó R 3.0.2 (Platform x86_64-pc-linux-gnu , 64 bits) mediante el entorno de desarrollo RStudio 0.98.1102. Para la generación de procesos de limpieza de datos, construcción de conjuntos de datos para grafos y ejecución de algoritmos de análisis se hace uso de Python 2.7.6 y Java SE 1.8.0_66. Finalmente, para graficar los distintos grafos se utilizó Gephi 0.8.2 con algunos algoritmos para optimizar la visualización, propios del software.

5.2 LIBRERÍAS

En cuanto a librerías de R, se usaron:

- Para importación y manipulación de datos biológicos:
  - Biobase 2.20.0
  - BioGenerics 0.8.0
  - GEOQuery 2.28.0
  - Minet 3.16.0
  - c3net 1.1.1
  - genefilter 1.44.0
- Para acceso a NEO4J:
  - RNeo4j 1.5.0
    - [http://neo4j.com/developer/r/]
    - [https://github.com/nicolewhite/RNeo4j]
- Para análisis SNA
  - sna 2.3.2
- Para análisis de cliques:
  - igraph 0.7.1
    - [http://www.inside-r.org/packages/cran/igraph/docs/maximal.cliques]

En java se usaron las siguientes librerías:

- Graph-algo 2.2.4 [https://github.com/neo4j/neo4j/tree/2.2.4/community/graph-algo]: para ejecución de algoritmos de tipo SNA sobre multigrafo.
- Neo4j 2.2.4 JAR: para invocación de bases de datos embebidas.
- neo4j-shell-tools_2.2 [https://github.com/jexp/neo4j-shell-tools]: para exportación e importación de grafos a varios formatos, entre ellos el estándar graphML.

5.3 SERVICIOS

Se usaron los siguientes servicios:

- Servicio web proporcionado por string-db [60] para enriquecer datos de proteínas de los grafos y el multigrafo.

5.4 DESCRIPCIÓN DE LOS DATOS BASE

Se eligieron tres tipos de datos relacionados con metodologías comunes en biología molecular:

- **Red de interacción Proteína - Proteína o PPI**: En total dos conjuntos de datos (conjuntos de datos 1 y 3). En ambos casos se presenta un listado en donde cada registro corresponde a una interacción de dos proteínas con información adicional - dependiendo del conjunto de datos - como el nombre oficial, número de acceso o sinónimo de la proteína, el método de detección, las publicaciones en las cuales aparece y anotaciones generales entre otros.

- **Red de regulación de la expresión de genes a nivel transcripcional:** Un conjunto de datos (conjunto de datos 2). Se presenta un listado en el cual cada registro corresponde a una proteína regulatoria, su sinónimo, el o los genes promotores al igual que detalles del inicio y fin de la región promotora y la secuencia de la misma a nivel DNA.

- **Red de co-expresión génica:** Dos conjuntos de datos (conjuntos de datos 4 y 5). En ambos casos, los datos base se componen de datos agrupados por conjunto de muestras las cuales son correspondientes a experimentos con microarreglos de M. tuberculosis en los cuales se aplicaron distintas drogas o sustancias para analizar su reacción en términos de co-expresión génica.

Conjunto de datos inicial	Nombre del trabajo base	Tipo datos	Número de registros
1	*Uncovering new signaling proteins and potential drug targets through the interactome analysis of Mycobacterium tuberculosis* [7]	Interacción Proteína-Proteína o PPI (*Protein-Protein Interaction*)	17.684
2	*A genome-Wide Regulator−DNA Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv* [8]	Regulación de la expresión de genes a nivel transcripcional	5.414

3	*Global Protein-Protein Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv* [9]	Interacción Proteína-Proteína o PPI (*Protein-Protein Interaction*)	8.042
4	*Delayed bactericidal response of Mycobacterium tuberculosis to bedaquiline involves remodelling of bacterial metabolism* [10]	Datos de microarreglos: expresión génica	6.022
5	*A Novel In Vitro Multiple-Stress Dormancy Model for Mycobacterium tuberculosis Generates a Lipid-Loaded, Drug-tolerant, Dormant Pathogen* [11]	Datos de microarreglos: expresión génica	10.800

Tabla 3: Datos resumidos de conjuntos de datos seleccionados para experimentación.

Para los tres tipos de datos se seleccionaron 5 conjuntos de datos (Ver **Tabla 3**) de la cepa H37Rv de la bacteria *M. tuberculosis* que a continuación serán descritos con detalle:

1. **Conjunto de datos 1**: En el trabajo [7] se plantea un estudio de una red de interacción proteína-proteína o PPI de la cepa ya mencionada, construida usando un método de mapeo homogéneo de proteínas. La red construida se compone de 738 proteínas y 5.639 pares de interacciones. El estudio - como es mencionado por sus autores - provee importantes indicios en la exploración de señalización de proteínas, vías de virulencia y drogas objetivo. A continuación, para ver más detalles del conjunto de datos ver la **Tabla 4**.

Código del conjunto de datos	1
Nombre trabajo base	Uncovering new signaling proteins and potential drug targets through the interactome analysis of Mycobacterium tuberculosis [7]
Tipo datos	Interacción Proteína - Proteína o PPI(*Protein-Protein Interaction*)
Método de obtención	Método de mapeo homogéneo de proteínas
Atributos	1. MTB interaction partner protein A 2. MTB interaction partner protein B 3. Corresponding DIP interaction ID 4. DIP interaction detection method(s) 5. Publication(s)
Descripción de conjunto de datos	El conjunto de datos corresponde al archivo adjunto en el trabajo correspondiente a la tabla S5 llamada: "El interactoma de *M. tuberculosis* obtenido a partir de las interacciones verificadas experimentalmente desde DIP(Database of interacting protein)"
Archivo original - Url origen	1471-2164-10-118-s5.xls - http://www.genomebiology.com/content/supplementary/1471-2164-10-118-s5.xls

Número de registros	17.684

Tabla 4: Detalles de conjunto de datos 1 de tipo PPI.

2. **Conjunto de datos 2**: En el trabajo [8] los autores construyen una red de regulación de la expresión de genes a nivel transcripcional usando el método B1H *(Bacterial One/Hybrid*) de alto rendimiento. Una red de interacciones a nivel de factores de transcripción se derivó sobre la base de la identificación de más de 5.400 interacciones proteína-DNA. Los hallazgos resaltan los mecanismos de regulación del mce1 (*mammalian cell entry 1*) el cual incluye el operón *mce1R* y el *mce1*. Para más detalles del conjunto de datos ver la **Tabla 5**.

Código del conjunto de datos	2
Nombre trabajo base	A Genome-Wide Regulator−DNA Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv [8]
Tipo datos	Regulación de la expresión de genes a nivel transcripcional
Método de obtención	B1H *(Bacterial One/Hybrid*) de alto rendimiento
Atributos	1. Regulator ID 2. Regulator name 3. Target promoter 4. Promoter coordinates in the genome 5. Gene in the defined operon 6. promoter sequence
Descripción de conjunto de datos	El conjunto de datos corresponde al archivo adjunto en el trabajo correspondiente a la tabla llamada: "Tabla S3. Lista de las interacciones proteína-DNA generadas desde *bacterial one-hybrid screen* en este estudio".
Archivo original - Url origen	pr3006233_si_008.xls - http://pubs.acs.org/doi/suppl/10.1021/pr3006233/suppl_file/pr3006233_si_008.xls
Número de registros	5.414

Tabla 5: Detalles de conjunto de datos 2 de tipo regulación de la expresión de genes a nivel transcripcional.

3. **Conjunto de datos 3**: Al igual que el conjunto de datos 1, en el trabajo [9] se plantea el estudio de una red de interacción proteína-proteína o PPI, sin embargo, hace uso de datos obtenidos mediante la técnica Y2H (*yeast two-hybrid*) de alto rendimiento. Casi todo el transcriptoma fue clonado, y se identificaron más de 8.000 nuevas interacciones. EL trabajo genera el mismo tipo de datos y aplicaciones que los del conjunto de datos 1. Para más detalles del conjunto de datos ver la **Tabla 6**.

Código del conjunto de datos	3
Nombre trabajo base	Global Protein-Protein Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv [9]
Tipo datos	Interacción Proteína - Proteína o PPI(Protein-Protein Interaction)
Método de obtención	Y2H(yeast two-hybrid). Tomando conjunto de datos parcial de PPI
Atributos	1. MTB interaction partner protein A 2. MTB interaction partner protein B 3. Gene for interactor A 4. Gene for interactor B 5. Annotation of interactor A 6. Annotation of interactor B
Descripción de conjunto de datos	El conjunto de datos corresponde al archivo adjunto en el trabajo correspondiente a la tabla llamada: "Tabla S2 Pares de interacción proteína-proteína de Mycobacterium tuberculosis H37Rv".
Archivo original - Url origen	pr100808n_si_008.xls - http://pubs.acs.org/doi/suppl/10.1021/pr100808n/suppl_file/pr100808n_si_008.xls
Número de registros	8.042

Tabla 6: Detalles de conjunto de datos 3 de tipo PPI.

4. **Conjunto de datos 4**: En el trabajo [10] se utilizan microarreglos para confirmar la ausencia de actividad bacteriolítica significativa durante los primeros 3-4 días de exposición a BDQ (*Gedaquilina*), la cual es la primera droga aprobada para tratamiento de tuberculosis multi-resistente en décadas. Los resultados mostraron que la exposición a la BDQ desencadena una remodelación metabólica en *M. tuberculosis*, que aunque permite de este modo la supervivencia bacteriana transitoria durante los primeros 3-4 días de la aplicación, posteriormente conlleva a la muerte de la misma. Para ver más detalles del conjunto de datos ver la **Tabla 7**.

Código del conjunto de datos	4
Nombre trabajo base	*Delayed bactericidal response of Mycobacterium tuberculosis to bedaquiline involves remodelling of bacterial metabolism* [10]
Tipo datos	Datos de microarreglos: datos de expresión génica
Método de obtención	Red de co-expresión génica usando librería y método C3NET [54] [55] [56]

Atributos	1. ID 2. GeneChip Array 3. Species Scientific Name 4. Annotation Date 5. Sequence Source 6. ORF 7. Target Description 8. Gene Title 9. Gene Symbol 10. Chromosomal Location 11. ENTREZ_GENE_ID 12. RefSeq Protein ID 13. Annotation Description 14. Annotation Note 15.SPOT_ID
Nombre de base de datos	*GeoMiame - Gene Expression Omnibus (GEO)*
Instituto de la base de datos	NCBI NLM NIH
Código de la serie	GSE43749
Nombre de la serie	*Global Gene expression data from Mycobacterium tuberculosis treated with new TB drug named Bedaquiline*
Resumen de la serie	*Sirturo or Bedaquiline inhibe la ATP sintetasa de Mycobacterium tuberculosis. Se utilizaron microarreglos para estudiar cambios inducidos por compuesto en la expression génica general y especialmente sobre genes de ATP sintetasas y de otras vias emtabólicas.*
Descripción del diseño general de la serie	*Se cultivaron células de Mycobacterium tuberculosis in caldo Middlebrook 7H9 y tratados con 1 µM Bedaquiline. Se extrajo ARN a cinco tiempos diferentes después del tratamiento: 0 min (T0), 30 min (T30), 60 min (T60), 180 min (T180) and 360 min (T360).*
Número de muestras	17
Archivo original - Url origen	GSE43749_family.soft.gz - ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE43nnn/GSE43749/soft/GSE43749_family.soft.gz
Número de registros	6.022

Tabla 7: Detalles de conjunto de datos 4 de tipo co-expresión génica.

5. **Conjunto de datos 5**: En el trabajo [11] se desarrolla un nuevo modelo de latencia por múltiples estreses *in vitro* para *M. tuberculosis*, aplicando tensiones combinadas: baja de oxígeno (5%), alta de CO_2 (10%), baja concentración de nutrientes (10% de medio Dubos) y pH ácido (5,0), planteando las condiciones de la bacteria cuando ésta se encuentra en el huésped. Se concluye que el método induce un estado que reúne todos los criterios de latencia, siendo adaptable a la selección de alto rendimiento de medicamentos que pueden matar la bacteria en

estado latente. Para más detalles del conjunto de datos ver la **Tabla 8**.

Código del conjunto de datos	5
Nombre trabajo base	A Novel In Vitro Multiple-Stress Dormancy Model for Mycobacterium tuberculosis Generates a Lipid-Loaded, Drug-tolerant, Dormant Pathogen [11]
Tipo datos	Datos de microarreglos: datos de expresión génica
Método de obtención	Red de co-expresión génica usando librería y método C3NET [54] [55] [56]
Atributos	1. ID 2. MetaRow 3. MetaColumn 4. SubRow 5. SubColumn 6. ORF 7. GeneSymbol 8. CommonName 9. Strain 10. SwissPROT_TrEMBL 11. GenBank_ID 12. SEQUENCE 13. Well_ID
Nombre de base de datos	GeoMiame - Gene Expression Omnibus (GEO)
Instituto de la base de datos	NCBI NLM NIH
Código de la serie	GSE10391
Nombre de la serie	*In vitro dormancy achieved by multiple stresses in Mycobacterium tuberculosis*
Resumen de la serie	*M. tuberculosis* desarrolla una infección durmiente por la cual el organismo sobrevive dentro del hospedero. Se estableció un modelo *in vitro* de dormición basado en la caracterización de la resistencia a las drogas INH y rifampina. Se mantuvieron células de *M. tuberculosis* en condiciones de control y de estrés causado por bajo oxígeno (tensión de oxígeno disuelto 5%), acidez (pH 5) junto con hambreado de glicerol en el medio de cultivo. Para monitorear los cambios en la expresión génica en respuesta a los múltiples estreses se realizaron análisis de microarreglos a los días 1, 2, 3, 6 y 12 post-tratamiento. La adaptación de *M. tuberculosis* a múltiples estreses mostró características asociadas a la persistencia *in vivo* incluyendo la entrada en un estado no-replicativo y la represión de genes involucrados en la regeneración de energía. Bajos estreses múltiples, *M. tuberculosis* moduló la expresión génica de manera significativa, en especial en respuesta al estrés por hambreado. Las células sometidas a estas condiciones de estrés mostraron una significativa resistencia a drogas. La comparación con otros perfiles de expresión *in vivo* demuestra la inducción de varios genes en común con los inducidos en condiciones de dormición *in vitro*.

Descripción del diseño general de la serie	*Se tomaron muestras sometidas a diferentes condiciones de estrés a los días 1,2,3,6 y 12 para realizar hibridaciones a microarreglos.*
Número de muestras	75
Archivo original - Url origen	GSE10391_family.soft.gz - ftp://ftp.ncbi.nlm.nih.gov/geo/series/GSE10nnn/GSE10391/soft/GSE10391_family.soft.gz
Número de registros	10.800

Tabla 8: Detalles de conjunto de datos 5 de tipo co-expresión génica.

5.5 CONJUNTOS DE DATOS PARA GRAFOS INDIVIDUALES

Para la construcción de cada conjunto de datos correspondiente a cada grafo, se siguieron distintos tipos de procesos dependiendo de las características del conjunto de datos base del cual se parte para generar la red.

A nivel general, se opta por un diseño de los campos de cada registro optimizados para ser usados en bases de datos no relacionales en donde los atributos son almacenados y consultados de forma ***clave:valor***. Se define entonces un "tipo" de entidad asociada a los nodos llamada PARTICLE en donde se pueden almacenar datos tanto de proteínas como genes - dependiendo de la red -.

A nivel relaciones se definen relaciones asociadas a cualquiera de los tres tipos de datos del cual se extrajo:

- **INTERACTION_PROTEINES**
- **INTERACTION_REGULATION**
- **INTERACTION_CO_EXPRESSION**

Se definen posibles pares de ***clave:valor*** tanto para los nodos (PARTICLE) como para las relaciones, los cuales son variables según el conjunto de datos y permite la consulta flexible de los datos dependiendo si existe o no dicho atributo en el nodo o la relación. En la **Ilustración *63*** se puede ver más claramente el diseño conceptual del proceso de generación de los grafos individuales y su correspondencia con los pares atributo-valor de los nodos y relaciones. Una explicación del proceso mostrado en la ilustración puede ser visto en el apartado: *3.6.1 - Construcción de multigrafo.*

5.5.1 Composición de datos de los nodos

A continuación, se describe la composición de datos en los nodos de los grafos:

- **Tipos de nodos [PARTICLE]:** tipo de nodo planteado en la red. En éste caso, dado que éstos pueden representar tanto genes como las proteínas para los que codifican, se decide que el término "PARTÍCULA" generaliza el concepto.

- **Atributos para los nodos:**
    - **Información de la proteína o gen:** mediante la consulta a la base de datos Stringdb [60] haciendo uso de un servicio de consulta se realiza la agregación de datos descriptivos de cada proteína o gen.
        - **ID:** correspondiente al código de la proteína o gen. Ejemplo Rv0002.
        - **PREFERED_NAME:** Nombre de la proteína o sinónimo. Ejemplo "dnaN" correspondiente a la proteína con código o ID Rv0002.
        - **NCBI_TAXON_ID:** Código genérico para identificar el taxón - organismo al cual pertenece la proteína -, en nuestro caso 83332 (*M. tuberculosis* H37Rv). Si en algún momento se extiende el modelo éste atributo tendrá valores de distintos organismos y podrá ser usado para procesos de filtro.
        - **TAXON_NAME:** Nombre genérico del taxón, en éste caso *M. tuberculosis* H37Rv. Si en algún momento se extiende el modelo éste atributo tendrá valores de distintos organismos.
        - **ANNOTATION:** Anotaciones con respecto a funciones de la proteína y descripciones generales de su comportamiento y posibles mecanismos en los cuales está relacionada.

5.5.2 Composición de datos de las relaciones

A continuación, se describe la composición de datos de las relaciones en los grafos:

- **Tipo de relaciones [INTERACTION_PROTEINES,INTERACTION_REGULATION, INTERACTION_CO_EXPRESSION]**: corresponde al tipo de red que representa la relación y está asociado a los tres tipos de datos relacionados con procesos en biología molecular descritos anteriormente. Para facilitar su identificación y las consultas, se les ha agregado al final del nombre un número correspondiente al orden del conjunto de datos según el tipo de red, es decir:
    - **INTERACTION_PROTEINES_[1-2].**
    - **INTERACTION_REGULATION_[1].**
    - **INTERACTION_CO_EXPRESSION[1-2].**

- **Atributos de información de las relaciones**: Debido a que en el multigrafo construido en etapas posteriores se tienen grafos dirigidos y no dirigidos, se generan en las relaciones atributos para identificar correctamente el sentido de la relación y sus atributos.
    - **INTERACTION_TYPE [PROTEIN-PROTEIN, PROTEIN-GEN, GEN-GEN-COEXPRESION]:** tipo de relación presentada entre las dos partículas que representan el nodo.
    - **PARTICLE_A:** código de la partícula inicial origen.
    - **PARTICLE_B:** código de la partícula final destino.
    - **DIP_INTERACTION_ID:** código de la interacción proteína - proteína en la base de datos DIP (*Data of Interacting Proteins*) [61] .
    - **DETECTION_METHODS:** Método de detección de la relación entre las partículas.
    - **PUBLICATIONS:** Publicaciones en las cuales aparece mencionada o descrita la relación o las proteínas o genes.
    - **TARGET_PROMOTER:** Nombre de la región promotora.

- **PROMOTES_COORDINATES_GENOMA:** coordenadas de inicio y fin de la región de DNA que inicia la transcripción de un gen en particular.
- **GENE_IN_OPERON:** listado de genes identificados en el operón.
- **PROMOTER_SEQUENCE:** secuencia de ADN de la región promotora correspondiente a la hebra codificante de los genes río arriba.
- **GENE_INTERACTOR_A:** nombre del gen correspondiente a la proteína A en el registro.
- **GENE_INTERACTOR_B:** nombre del gen correspondiente a la proteína B en el registro.
- **ANNOTATION_INTERACTOR_A:** Anotaciones con respecto a funciones de la proteína inicial origen y descripciones generales de su comportamiento y posibles mecanismos en los cuales está relacionada.
- **ANNOTATION_INTERACTOR_B:** Anotaciones con respecto a funciones de la proteína final destino y descripciones generales de su comportamiento y posibles mecanismos en los cuales está relacionada.
- **FOUND [DATASET_1, DATASET_2, DATASET_3, DATASET_4, DATASET_5]:** Conjunto de datos en el cual se encontró la relación entre los nodos.

- **Datos de grafo**: son métricas relacionadas con el grafo que permiten eliminar conexiones del mismo sin perder información**.**
  - **COUNTER_REPITED:** Número de repeticiones que se presentan de la relación en el grafo. Se incluyen relaciones de un nodo consigo mismo, generando solo una relación.
  - **STD_COUNTER_REPITED:** Valor estandarizado del número de repeticiones de la relación calculada basado en el máximo número de repetición. Se incluyen relaciones de un nodo consigo mismo, generando solo una relación.

A continuación, se presenta en la **Tabla 9** la correspondencia entre atributos en las relaciones dependiendo del tipo de datos al cual pertenecen las mismas.

	Nombre del atributo en la relación	**Tipo de dato**	**Conjunto de datos base**				
			1	**2**	**3**	**4**	**5**
Atributos con información de las relaciones	INTERACTION_TYPE	*List*	X	X	X	X	X
	PARTICLE_A	*String*	X	X	X	X	X
	PARTICLE_B	*String*	X	X	X	X	X
	DIP_INTERACTION_ID	*String*	X				
	DETECTION_METHODS	*String*	X				
	PUBLICATIONS	*String*	X				
	TARGET_PROMOTER	*String*		X			
	PROMOTES_COORDINATES_GENOMA	*String*		X			
	GENE_IN_OPERON	*String*		X			
	PROMOTER_SEQUENCE	*String*		X			
	GENE_INTERACTOR_A	*String*			X		
	GENE_INTERACTOR_A	*String*			X		
	ANNOTATION_INTERACTOR_A	*String*			X		
	ANNOTATION_INTERACTOR_B	*String*			X		

	FOUND	*String*	X	X	X	X	X
Atributos del grafo	COUNTER_REPITED	*Int*	X	X	X	X	X
	STD_COUNTER_REPITED	*Double*	X	X	X	X	X

Tabla 9: Correspondencia de atributos en las relaciones según el tipo de datos.

Como se puede observar, la información del tipo de objeto que representa el nodo (proteína o gen) no hace parte de la definición estructural de los atributos del nodo, esto debido a que en pasos posteriores, al usar los distintos grafos para obtener un multigrafo, se tendrá un nodo que dependiendo de la relación, representa una proteína o un gen; es por éste motivo que la mayor parte de atributos están asociados a la relación, y es justamente mediante ellos que se puede determinar - entre otras cosas - que tipo de objeto (proteína o gen) representa la partícula para esa interacción en particular que se describe mediante el atributo INTERACTION_TYPE.

Los detalles antes mencionados acerca de la agregación de atributos en los nodos y la relación permiten las siguientes consideraciones en el diseño:

1. Posible inclusión de otros conjuntos de datos correspondientes a tipo de datos distintos.
2. Posible inclusión de nuevos conjuntos de datos de un tipo ya planteado en el esquema del trabajo, por ejemplo, otro conjunto de datos de tipo Interacción proteína - proteína o PPI.
3. Variabilidad de pares ***clave:valor*** tanto en el nodo como en la relación.
4. Posible optimización de consultas a través de índices a nivel ***clave:valor*** en bases de datos no relacionales y a nivel de nodo/relación en bases de datos no relacionales orientadas a grafos.

A todos los cuatro grafos no dirigidos (1, 3, 4, 5) se les realiza un pre-procesamiento que a nivel general incluye:

- Eliminación de relaciones duplicadas, es decir, [A relacionado con B] = [B relacionado con A] y actualización de los datos de repeticiones correspondientes a los atributos COUNTER_REPITED y STD_COUNTER_REPITED.
- Eliminación de relaciones de un nodo consigo mismo, es decir [A relacionado con A] y actualización de datos de repeticiones como en el punto anterior.
- Agregado de información del gen o proteína. En algunos casos no se encuentran datos de la proteína mediante el servicio de Stringdb, por lo cual se hace una búsqueda en ***Tuberculist.epfl.ch*** [12] [62] y ***UNIPROT*** [63]. Solo en dos casos no se encuentra información por lo cual se marcan datos del PREFERED_NAME como: undefined_[ID]. En total solo se marcaron dos proteínas de ésta forma y 7 fueron consultadas en Tuberculist:
    - **Conjunto de datos 5:** Rv1792, Rv0947c, Rv1000, Rv1489c (*undefined*), Rv2099c, Rv2233 (*undefined*), Rv3294.
    - **Conjunto de datos 4:** Rv3128c.
    - **Conjunto de datos 5:** R3018A.

En la **Tabla 10** se puede ver el resumen de los cinco grafos basados en los conjuntos de datos con los nodos resultantes, el número de relaciones y el tipo de grafo obtenido.

Código del conjunto de datos	Nombre trabajo base	Tipo datos	Método	Tipo de conjunto de datos para el grafo final	Número de nodos	Número de relaciones
1	Uncovering new signaling proteins and potential drug targets through the interactome analysis of Mycobacterium tuberculosis	Interacción proteína-proteína o PPI(Protein-Protein Interaction)	Método de mapeo homogéneo de proteínas	Grafo no dirigido proteína-proteína	793	6.091
2	Genome-Wide Regulator−DNA Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv	Regulación de la expresión de genes a nivel transcripcional	B1H (Bacterial One/Hybrid) de alto rendimiento	Grafo dirigido proteína-Gen	1.244	8.500
3	Global Protein-Protein Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv	Interacción proteína-proteína o PPI(Protein-Protein Interaction)	Y2H(yeast two-hybrid). Tomando dataset parcial de PPI	Grafo no dirigido proteína-proteína	2.907	8.042
4	Delayed bactericidal response of Mycobacterium tuberculosis to bedaquiline involves remodelling of bacterial metabolism	Datos de Microarray: datos de expresión génica	Red de co-expresión génica usando librería y método C3NET [1][2]	Grafo no dirigido Gen-Gen	881	698
5	A Novel In Vitro Multiple-Stress Dormancy Model for Mycobacterium tuberculosis Generates a Lipid-Loaded, Drug-tolerant, Dormant Pathogen	Datos de Microarray: datos de expresión génica	Red de co-expresión génica usando librería y método C3NET [1][2]	Grafo no dirigido Gen-Gen	833	685

Tabla 10: Correspondencia entre conjunto de datos base y conjuntos de datos generados para construir los grafos individuales.

A continuación, se describen los procesos realizados para la generación de cada conjunto de datos para construir los grafos y el análisis SNA correspondiente a cada uno de ellos. Dependiendo de cada caso, la interpretación de algo grado de centralidad para cada tipo de dato puede ser generalizada así:

- **Información transcriptómica**: se presenta centralidad alta en un gen en particular cuando ese gen tiene un patrón de expresión similar al de varios genes más, lo que determina que tenga coeficiente de correlación altos con esos otros genes.
- **Interacciones Proteína-DNA**: se presenta cuando una proteína puede interactuar con la zona promotora de varios genes diferentes.

5.5.2.1 Grafo no dirigido 1

Conjunto de datos

Dado que se trata de un conjunto de datos base con relaciones proteína-proteína, no existe ningún proceso adicional para realizar antes de obtener el listado de relaciones inicial. Se realiza el filtrado genérico para resumir las relaciones repetidas y hacer agregación de información y se obtienen 6.091 relaciones distintas y 793 nodos o proteínas identificadas que serán nodos en el grafo.

Durante el filtrado, es notable el número de relaciones de proteínas consigo mismo, 1.344, es decir un 7,6% de las relaciones totales iniciales, en las cuales hay 577 proteínas distintas, es decir, un 72,7% del total de proteínas del conjunto de datos. Claramente la gran mayoría de proteínas interactúa consigo mismo al menos una vez. Este fenómeno puede ser real, ya que existen proteínas que forman complejos multiméricos, y esto podría ser una explicación. Sin embargo, tampoco puede descartarse que sea un artefacto de la técnica experimental. Para más información sobre las características generales del grafo ver la **Tabla *12***. En cuanto a la frecuencia de repetición de interacciones, como vemos en la **Ilustración 27**, existe un gran número con más de cinco repeticiones, siendo la interacción Rv0435C, Rv3610c la que se repite más veces: 74.

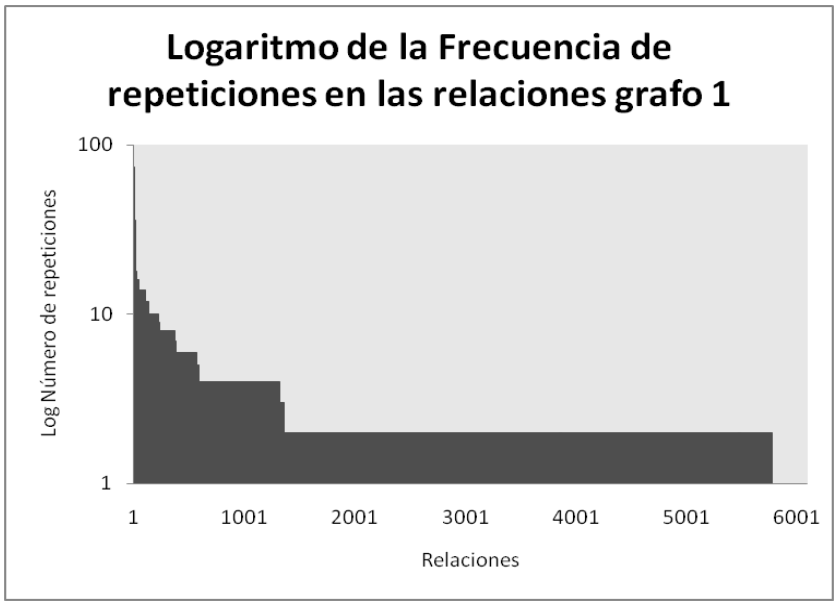

Ilustración 27: Logaritmo de la frecuencia de repeticiones de relaciones de proteínas en el grafo 1.

Las proteínas Rv0435c y Rv3610c tienen 37 relaciones consigo mismo, siendo las que más tienen, al igual que la Rv2115c, que tiene 36 relaciones, (Ver **Ilustración 28** con las frecuencias); adicionalmente se puede observar que hay un gran número de nodos que tiene más de 5 relaciones consigo mismo. En este caso, se podrían plantear las dos siguientes explicaciones:

- La interacción se da debido a que tanto la proteína Rv0435c y la Rv3610c son dos ATPasas y adicionalmente está descrito que ciertas ATPasas forman dímeros, es decir, estaríamos en presencia de proteínas diméricas o multiméricas.

- La interacción es producto de problemas en la calidad de los datos generados por un artefacto en el laboratorio, por ejemplo, generación de fosforilación entre proteínas con interacción.

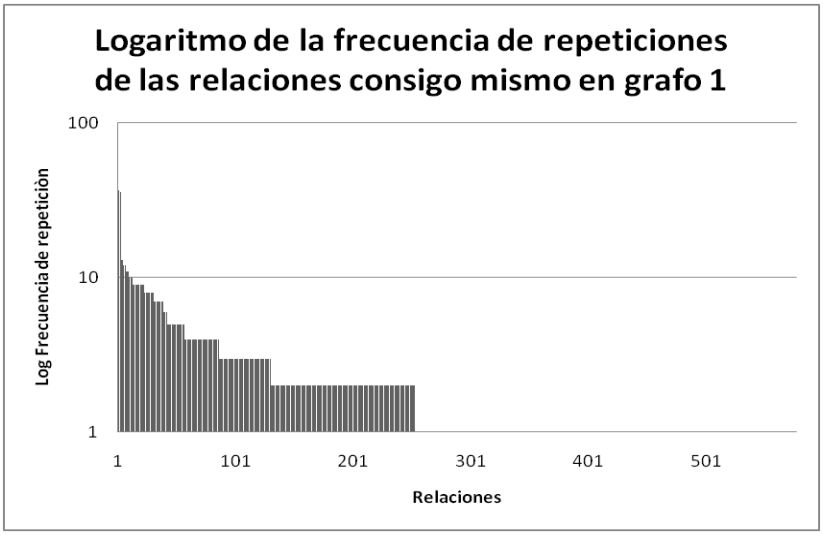

Ilustración 28: Logaritmo de la frecuencia de repeticiones de las relaciones de proteínas consigo mismo en el grafo 1.

Como se puede ver en la **Tabla 12**, comparado con los demás grafos, es el más altamente conectado y con mayor cantidad de repeticiones de relaciones.

Viendo la **Ilustración 29**, podemos notar grandes acumulaciones de nodos altamente conectados entre sí; también vemos la existencia de nodos con alto grado. Existen igualmente regiones interconectadas localmente (*clústeres*) como se ve en la **Ilustración 30**.

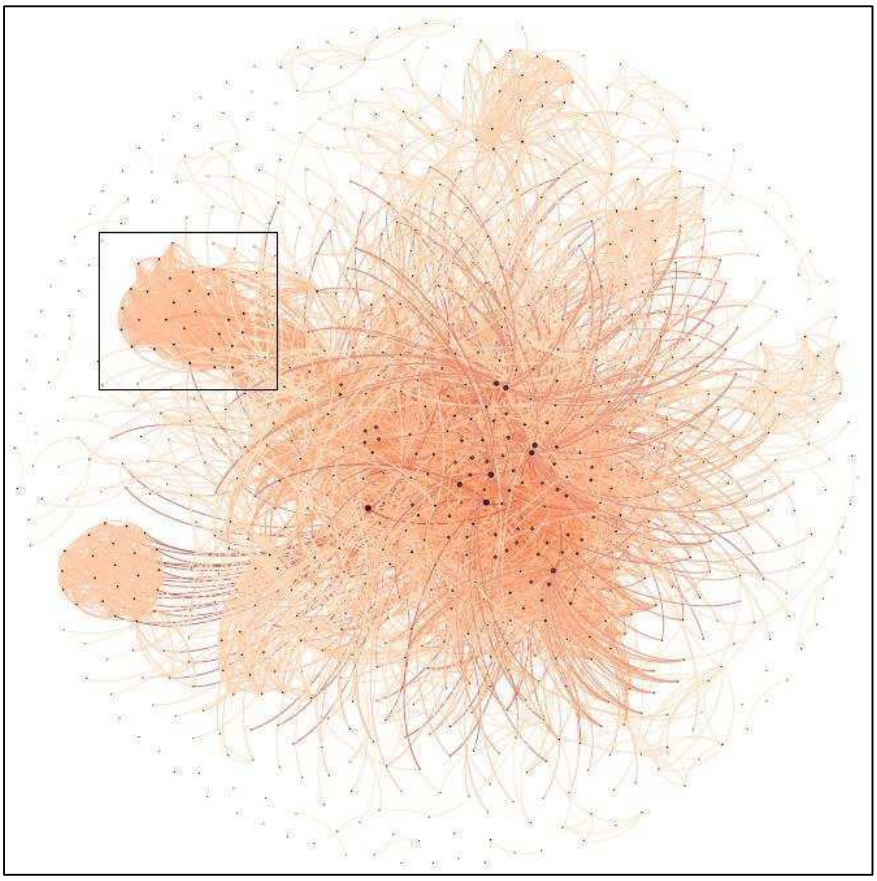

Ilustración 29: Grafo 1 con tamaño de nodos y color según el grado.

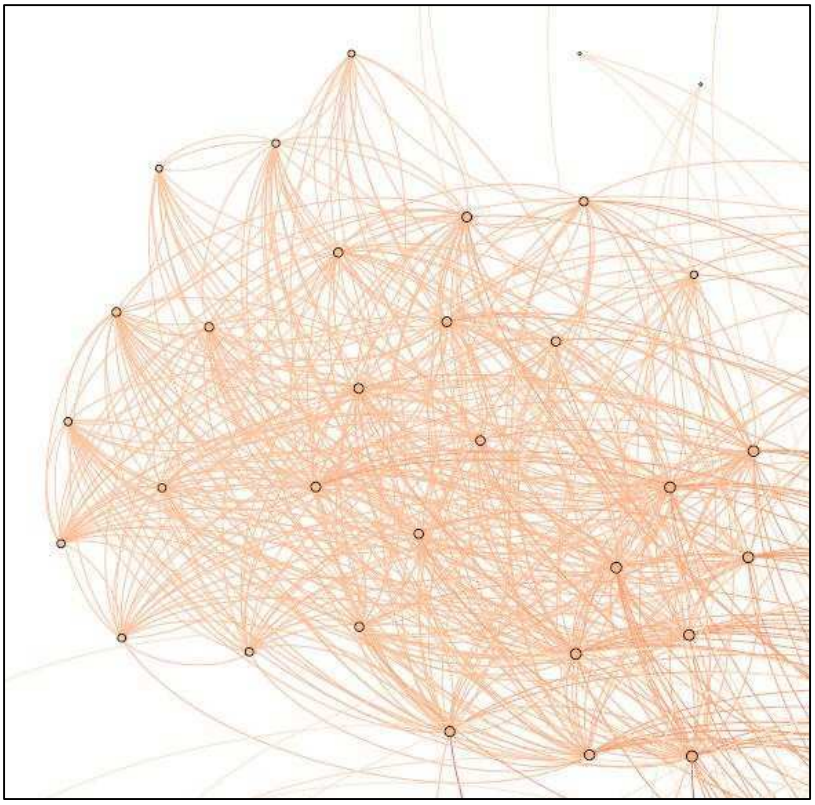

Ilustración 30: Zona ampliada del grafo 1 correspondiente a región demarcada en **Ilustración 29**.

SNA (Social Network Analysis)

Para ver el resumen completo ver **Tabla *12*.**

- **Distribución de grado:** Se presentan nodos con grados máximos de 131 (Rv0685) y con un promedio de 15,36, lo cual unido a que existen gran número de nodos con grados mayores de 5 (Ver **Ilustración 31**) muestra al grafo como el más altamente enlazado de los 5 analizados en el presente trabajo.

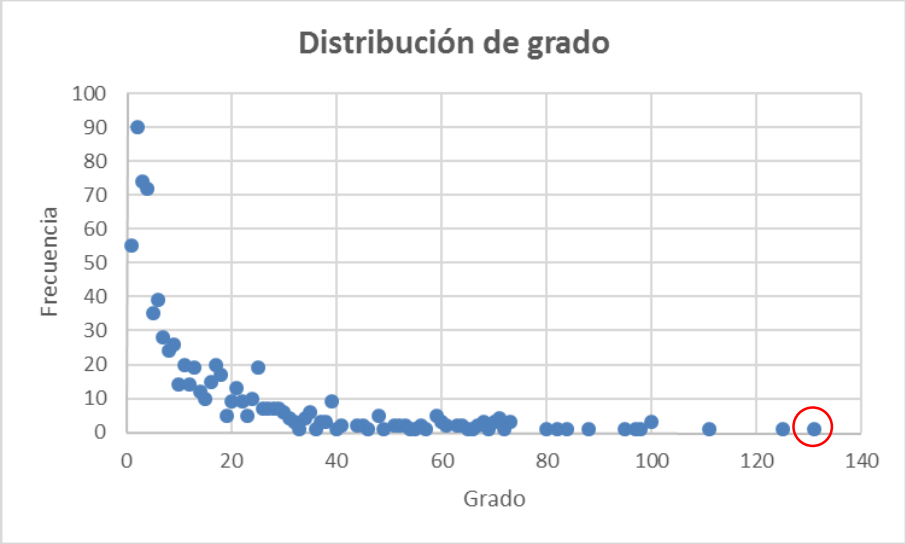

Ilustración 31: Distribución de grado de grafo 1.

- **Centralidad de intermediación (Betweenness Centrality):** En la **Ilustración 32**, vemos como existen algunos nodos con una medida de intermediación muy alta, siendo el Rv2890c con 7.028,73 la proteína con la más alta medida; adicionalmente, 396 proteínas tienen valores de intermediación mayores de cero, es decir un 49,93%.

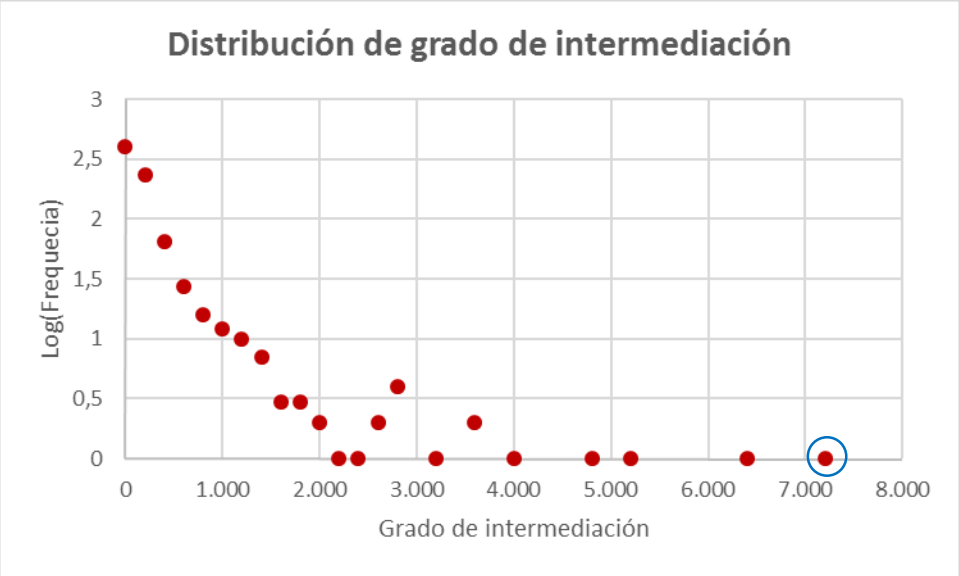

Ilustración 32: Distribución de centralidad de intermediación (*betweenness centrality*) para el grafo 1, usando escala logarítmica en la frecuencia.

- **Centralidad de cercanía (Closeness Centrality):** En la **Ilustración 33**, se muestra una aglomeración de nodos con medida de cercanía entre 2 y 5, existiendo algunos puntos *outliers* en donde se tiene 0 o 1 grado de cercanía. Las proteínas con grados de cercanía mayores a 5 son Rv05442c, Rv1843c, Rv3411c, Rv2534c, Rv3914 y Rv1086, siendo éste último el mayor grado con 5,59 de valor de cercanía. Para el caso de la proteína Rv1086, con el mayor grado de centralidad de cercanía, vemos que esta es muy relevante para la síntesis de muchos compuestos de la pared celular, crecimiento celular y en la biosíntesis de componentes de la pared celular [64], por lo cual se puede decir que tiene un rol central y por ende es lógico el hecho de que tenga tal grado de centralidad de cercanía en el grafo.

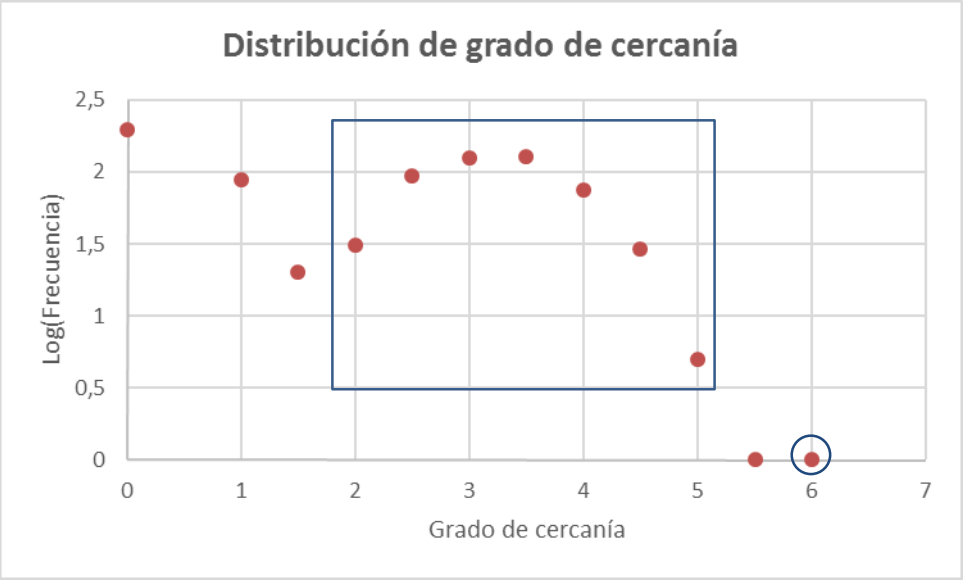

Ilustración 33: Distribución de centralidad de cercanía (*closeness centrality*) para el grafo 1, usando escala logarítmica en la frecuencia.

- **Modularidad (*Modularity*):** En la **Ilustración 34**, se pueden ver las 83 comunidades detectadas con un grado de modularidad de 0,54, siendo 19 de ellas compuestas por más de 2 nodos y 1 de ellas compuesta por 151 nodos. Existe por ende un agrupamiento a pesar de la característica altamente conectada del grafo.

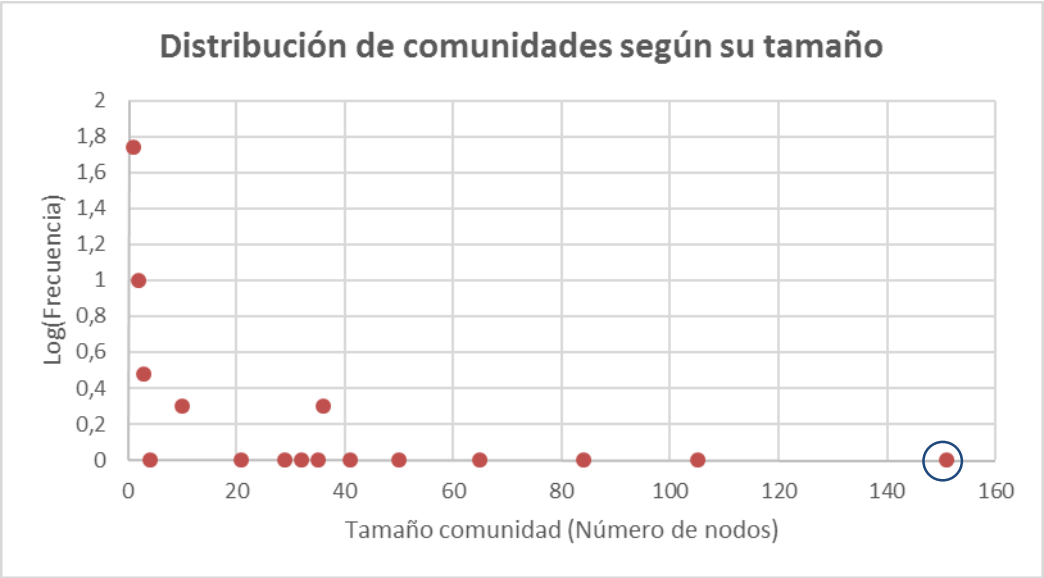

Ilustración 34: Distribución de comunidades según su tamaño para el grafo 1, usando escala logarítmica de la frecuencia.

- **Cliques:** Al realizar el análisis de cliques, el máximo cliqué tiene un tamaño de 25, lo cual demuestra una vez más lo altamente conectado que es el grafo. Un análisis inicial del número de cliques para los primeros 9 k-cliques muestra un crecimiento acelerado en cada k-clique (Ver **Ilustración 35**), también reflejado en el tiempo necesario para el procesamiento de la búsqueda de los mismos mostrado en la **Ilustración 36**, en donde se toma como base el menor de tiempo de procesamiento correspondiente al k clique igual a 1 para construir los valores de tiempo relativos, esto se hace debido a que se usa un equipo informático determinado y lo que se quiere mostrar es la complejidad que existe en el cálculo de los k-cliques a medida que el número k aumenta, por lo cual no se calcularon los demás hasta llegar a el máximo clique, es decir, 25.

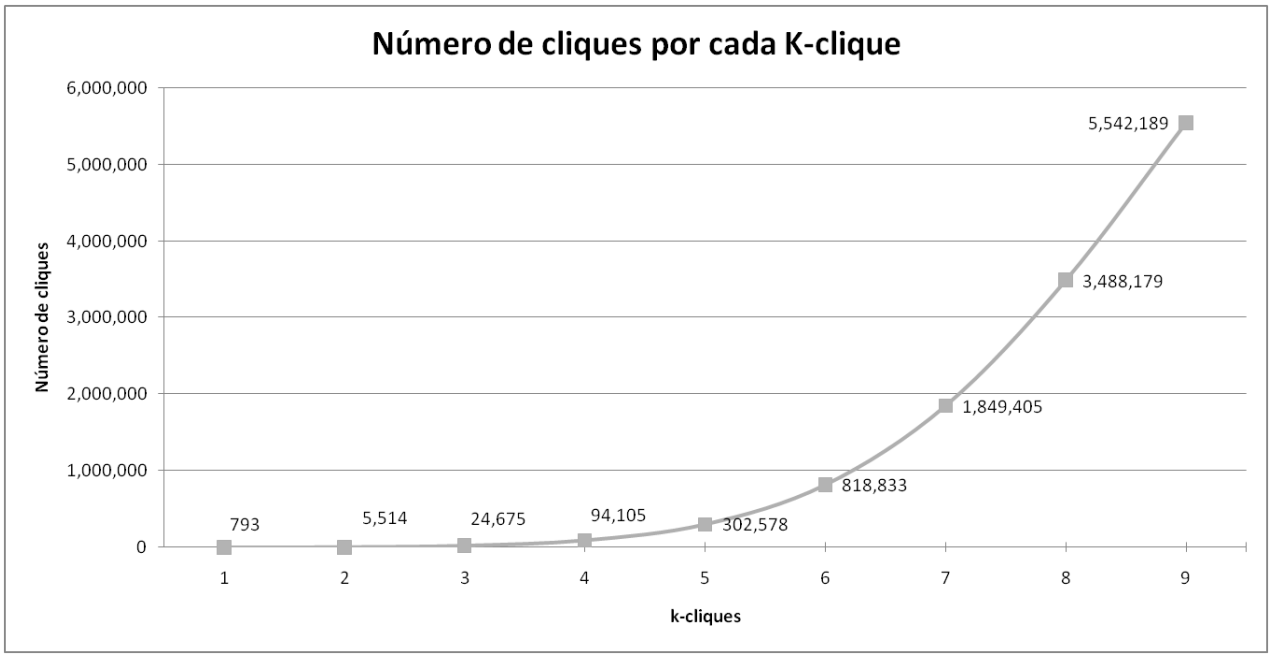

Ilustración 35: *Número de cliques del Grafo 1 por cada k-clique con k desde 1 hasta 9.*

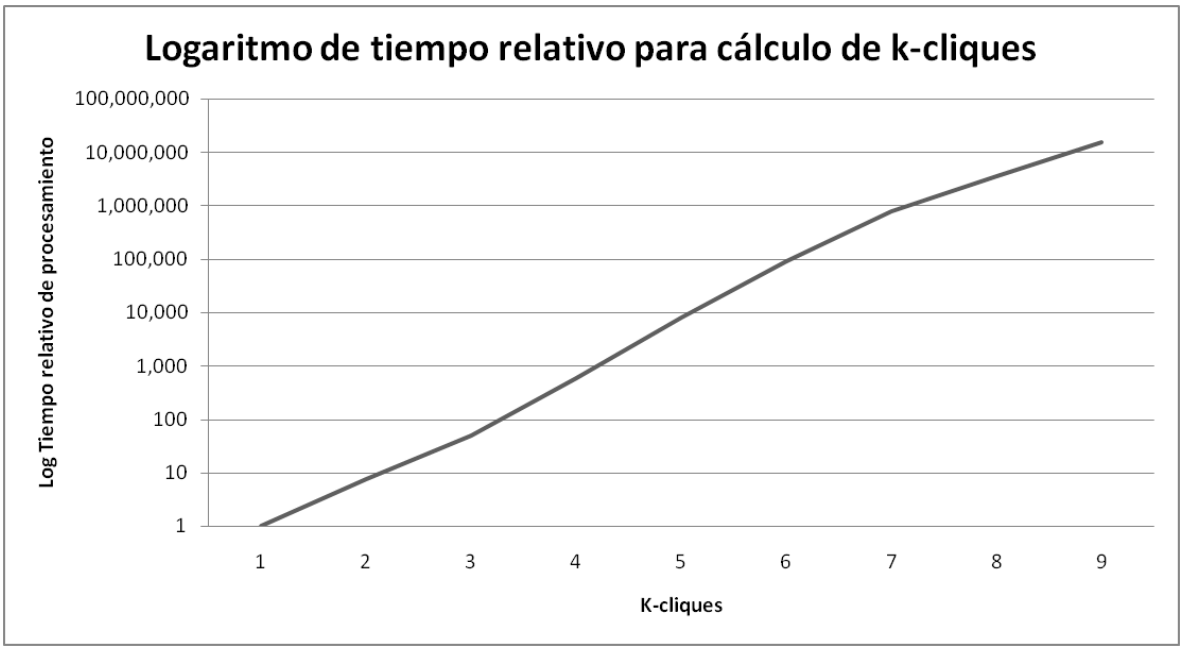

Ilustración 36: *Logaritmo de tiempo relativo de procesamiento en base al menor tiempo para cálculo de k-cliques.*

5.5.2.2 Grafo dirigido 2

El conjunto de datos 2, presenta por cada registro una proteína y el primer gen del operón correspondiente, además si la zona de ADN regulatorio corresponde a un gen que forma parte de un operón, se incluye un listado de genes que componen un operón, por lo que inicialmente se realiza una generación de relaciones entre la proteína base y cada gen miembro del operón, éste proceso incrementa en un 157% el número de relaciones y en un 672% el número de nodos ya que el listado de genes que componen cada operón en el conjunto de datos original en un 29% de los casos tiene más de un gen(3 genes en promedio) y en algunos casos de presentan listados de hasta 10 genes por operón. Posteriormente se realiza el filtrado genérico para resumir las relaciones repetidas y realizar agregación de información; de ésta manera se pasa de 5.414 relaciones a 8.500 y de 185 nodos distintos a 1.244.

Las repeticiones de nodos y relaciones son despreciables, presentándose tan solo 2 repeticiones con solo dos proteínas distintas. Para más información sobre las características generales del grafo ver la **Tabla *12***.

Por otro lado, en la **Ilustración 37** vemos como se presenta el efecto conocido como "súper nodo" [39], en donde debido al alto grado que tienen la mayoría de nodos, se genera una acumulación en el centro del grafo. También se pueden ver nodos con alto grado que tienen conexiones con nodos terminales (recuadro ampliado en **Ilustración 37**), debido a que en general dicho nodo es la proteína y los nodos terminales corresponden a los promotores, y si se trata de un operón, tendrá una conexión con cada uno de los genes de dicho operón.

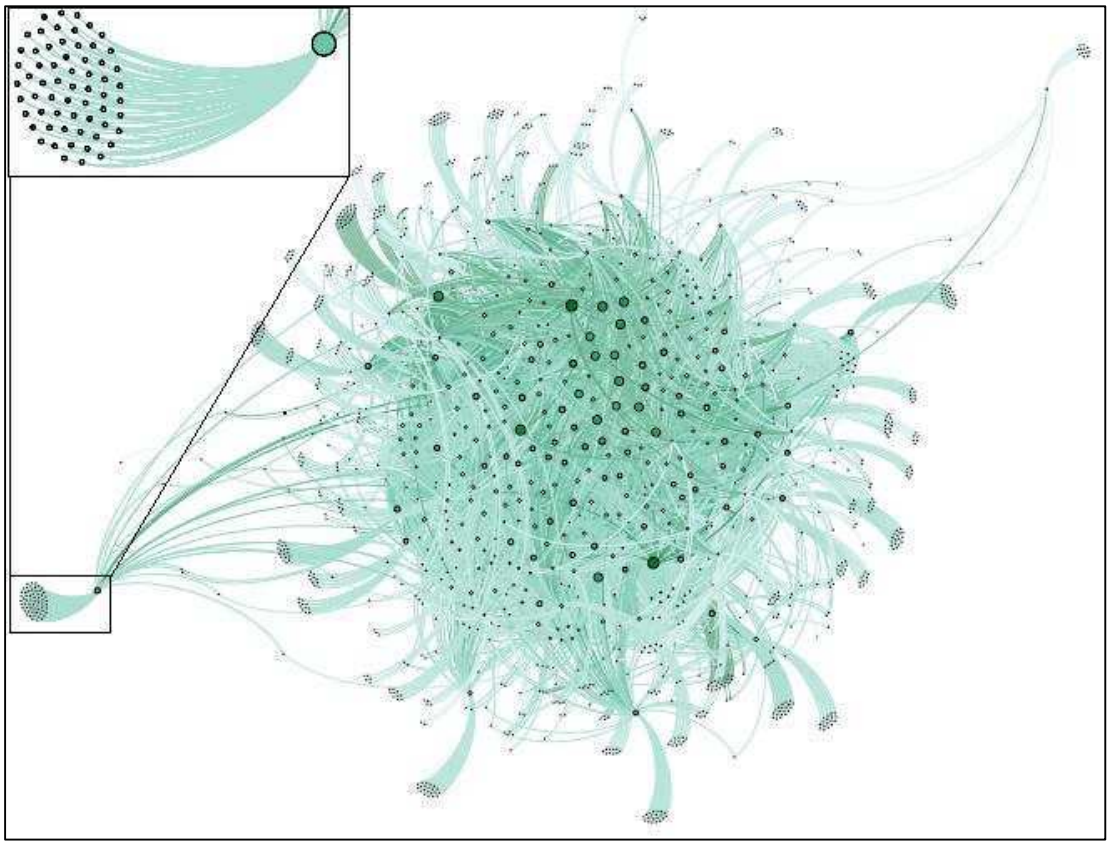

Ilustración 37: *Grafo 2 con tamaño de nodos y color según el grado.*

SNA (Social Network Analysis)

Para ver el resumen completo ver **Tabla *12***.

Distribución de grado: Se presentan nodos con grados máximos de 176 (Rv0685) y 171 (Rv1990C), con un promedio de 13,66. Un 31,18% de los nodos tienen un grado mayor o igual a 3 y un 53,6% tiene valor de 1 en su grado. En ambos casos: Rv1219c y Rv1990c son marcadas como probables proteínas transcripcionales [62], lo cual podría explicar su alto grado, siendo Rv1990c la proteína con máximo grado de intermediación en el subgrafo. Ver **Ilustración 38**.

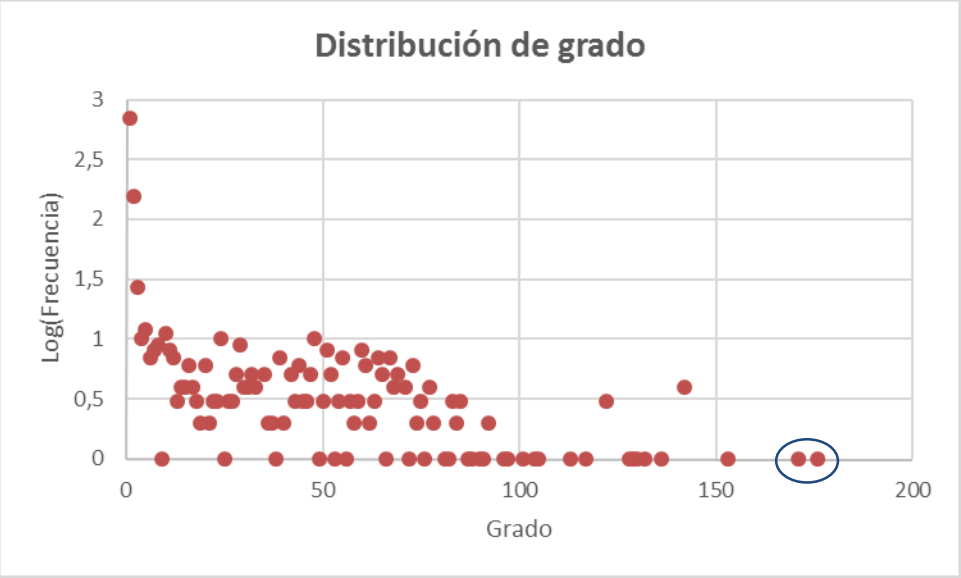

Ilustración 38: Distribución de grado del grafo 2, usando escala logarítmica para la frecuencia.

- **Centralidad de intermediación (*Betweenness Centrality*):** En la **Ilustración 39**, vemos que el nodo con máxima intermediación corresponde a la proteína Rv1990c con 16.117,93, sin embargo, con relación al grafo 1 el valor máximo no resulta ser tan alto. Tan solo 56 de las proteínas tiene un valor de intermediación distinto de cero, es decir, un 4,5% de todos los nodos.

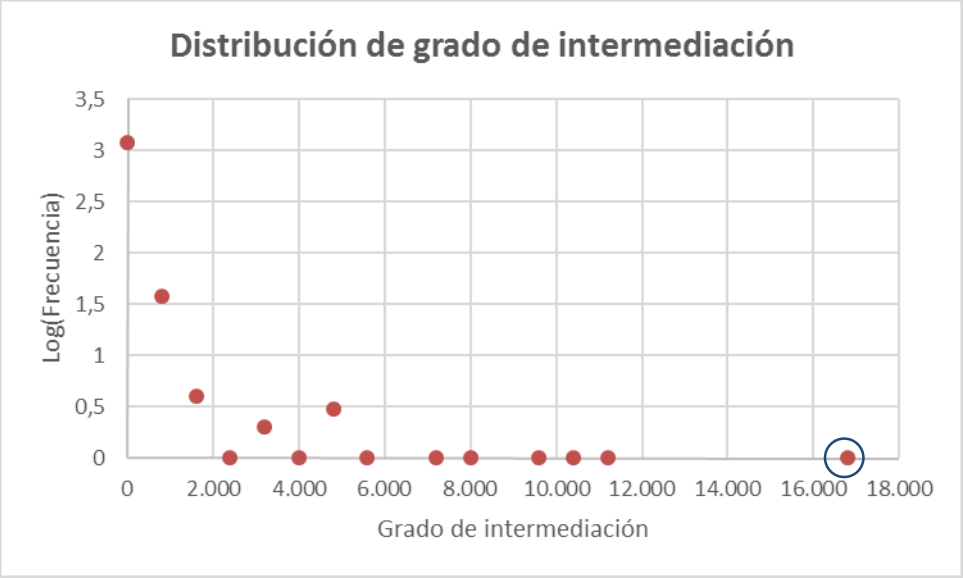

Ilustración 39: Distribución de centralidad de intermediación (*betweenness centrality*) para el grafo 2, usando escala logarítmica para la frecuencia.

- **Centralidad de cercanía (*Closeness Centrality*):** En la **Ilustración 33**, se muestra una aglomeración de nodos con medida de cercanía entre 2 y 4, existiendo un punto *outlier* en donde se tiene 0 grado de cercanía. Al igual que en el punto anterior, un bajo porcentaje tiene valores mayores a cero, es decir, 184 (14,8%). Las proteínas con grados de cercanía mayores a 4,6 son Rv3095, Rv3676, Rv3862c, Rv3765c, Rv0348 y Rv3855, siendo éste último el mayor grado

con 4,67 de valor de cercanía.

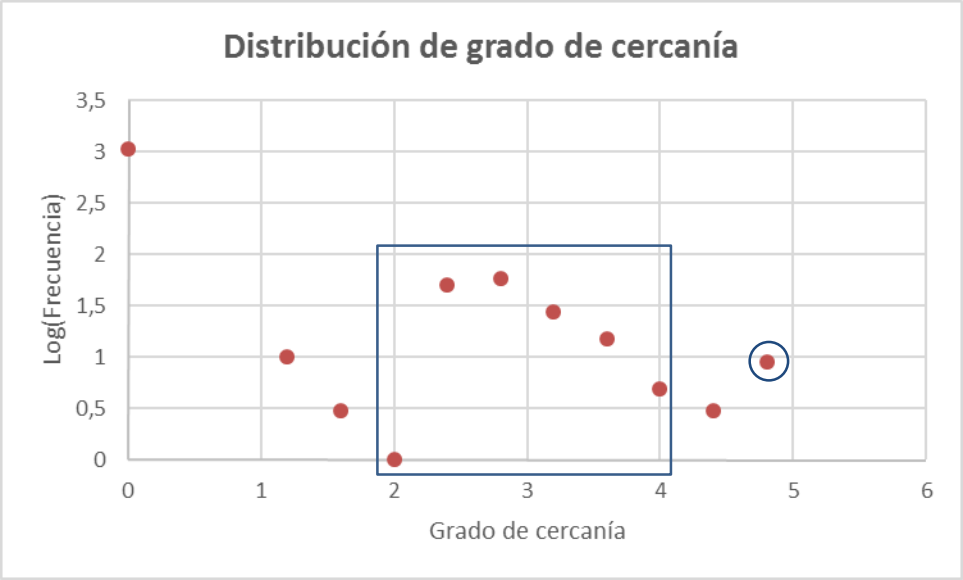

Ilustración 40: Distribución de centralidad de cercanía (*closeness centrality*) para el grafo 2, usando escala logarítmica para la frecuencia.

- **Modularidad (*Modularity*):** En la **Ilustración 41**, se pueden ver las 15 comunidades detectadas con un grado de modularidad de 0,23, estando todas ellas compuestas por más de dos nodos y una de ellas compuesta por 251 nodos. Como vemos, existe un número menor de comunidades, sin embargo, el valor de modularidad es más bajo que en el grafo 1 por lo que no pareciera representativo.

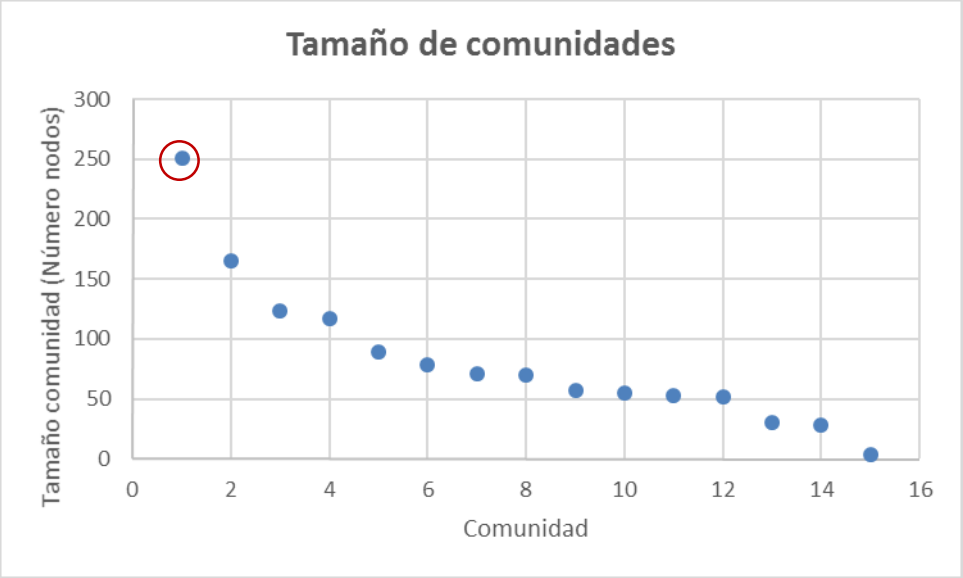

Ilustración 41: Tamaño de comunidades para el grafo 2.

- **Cliques:** Al realizar el análisis de cliques, el máximo cliqué es 6. Un análisis total del número de cliques que se puede ver en **Ilustración 42** muestra cómo debido a que no es un grafo tan altamente conectado como en el caso del grafo 1 se generan cliques pero con un máximo en número de k igual a 3.

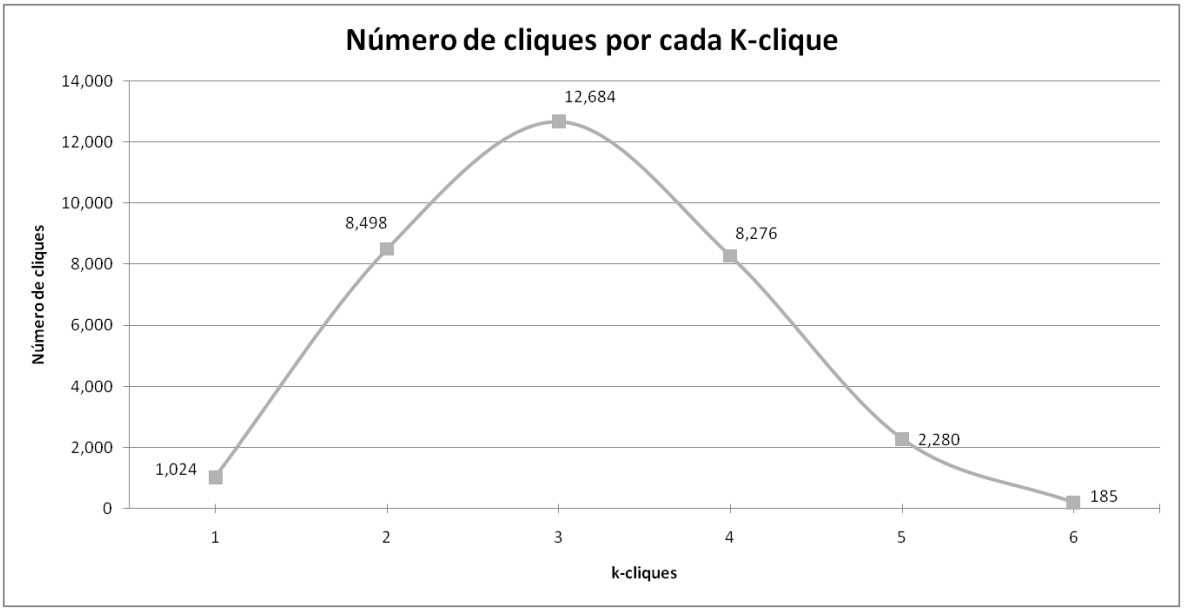

Ilustración 42: Número de cliques del Grafo 2 por cada k-clique con k desde 1 hasta 6 que es el máximo clique.

5.5.2.3 Grafo no dirigido 3

Conjunto de datos

El conjunto de datos se compone de relaciones proteína-proteína al igual que el grafo 1, y de manera similar a éste, no fue necesario un proceso adicional para obtener el listado de relaciones inicial. Se realiza el filtrado genérico para resumir las relaciones repetidas y realizar agregación de información y se obtienen 8.042 relaciones distintas y 2.907 nodos o proteínas identificadas que serán nodos en el grafo. Tan solo existen 35 relaciones de nodos consigo mismo, no existiendo relaciones entre nodos distintos repetidas.

Para más información sobre las características generales del grafo ver la **Tabla *12***.

Viendo la **Ilustración 43**, podemos ver que el grafo cuenta con muchas conexiones que no permiten distinguir regiones o clústeres, además de dos nodos con valores muy altos de grados:

- Rv2773c con 153 conexiones (izquierda parte media).
- Rv3416 con 113 conexiones (parte derecha superior).

Para el caso de Rv3416, se sabe que es un regulador transcripcional de las proteínas WhiB3 y de la familia WhiB (con nombre alternativo del gen: whmB) [62], además, está relacionado con mecanismos de crecimiento celular [65], por lo que la alta conectividad se podría explicar por su importancia y sus funciones asociadas.

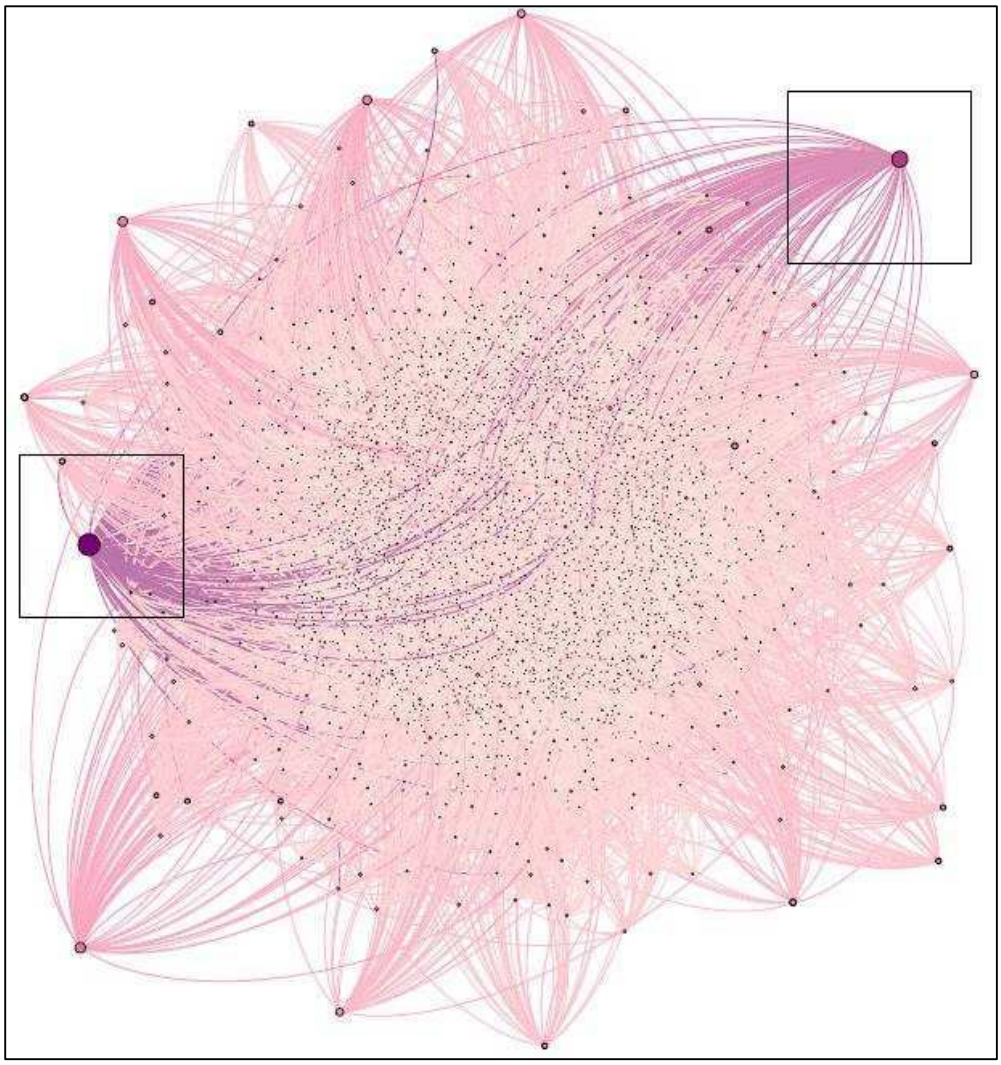

Ilustración 43: Grafo 3 con tamaño de nodos y color según el grado.

SNA (*Social Network Analysis*)

Para ver el resumen completo ver **Tabla *12*.**

- **Distribución de grado:** Se presentan nodos con grados máximos de 153 y 113 (ya mencionados en el párrafo anterior) y con un promedio de 5,53. Un 66,39% de los nodos tienen un grado mayor o igual a 3 y un 18,1% tiene valor de 1 en su grado, siendo el restante de grado 2. Ver **Ilustración 44**.

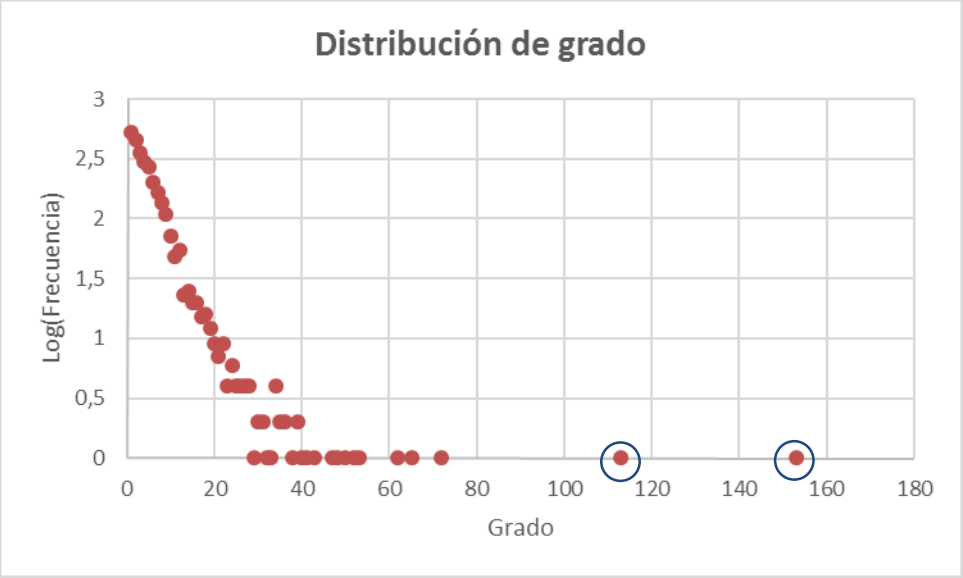

Ilustración 44: Distribución de grado del grafo 3, usando escala logarítmica para la frecuencia.

- **Centralidad de intermediación (*Betweenness Centrality*):** En la **Ilustración 45**, vemos que el nodo con máxima intermediación corresponde a la proteína Rv2773c con 529.766,96, siendo más de 11 veces el mayor grado existente en relación al grafo 1. A diferencia de otros grafos, 2.372 nodos tiene un valor de intermediación distinto de cero, es decir, un 81,59% de todos los nodos lo cual muestra la interrelación de conexiones entre nodos tan alta que existe, evidenciada por la proporción entre relaciones sobre nodos de 2,76.

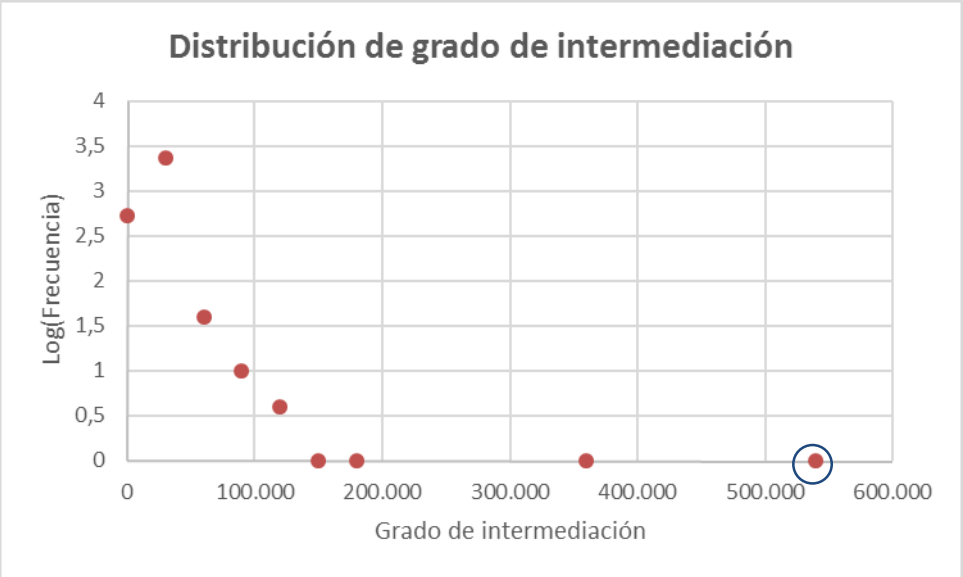

Ilustración 45: Distribución de centralidad de intermediación (*betweenness centrality*) para el grafo 3, usando escala logarítmica para la frecuencia.

- **Centralidad de cercanía (*Closeness Centrality*):** En la **Ilustración *46***, se muestra una aglomeración de nodos con medida de cercanía entre 3 y 7, existiendo algunos puntos *outliers*. Al igual que en el punto anterior, un bajo porcentaje tiene valores mayores a cero dada la correlación entre ambas medidas.

Las proteínas con grados de cercanía mayores a 6,9 son Rv3852, Rv0904c, siendo éste último el mayor grado con 7,01 de valor de cercanía. En el caso de Rv0904c, se sabe que es una carboxilasa que forma un complejo proteico, siendo importante para la biosíntesis de la membrana celular [66]; estas propiedades explicarían el alto número de genes que están relacionados, permitiendo así que Rv0904c conecte dos o más subgrafos.

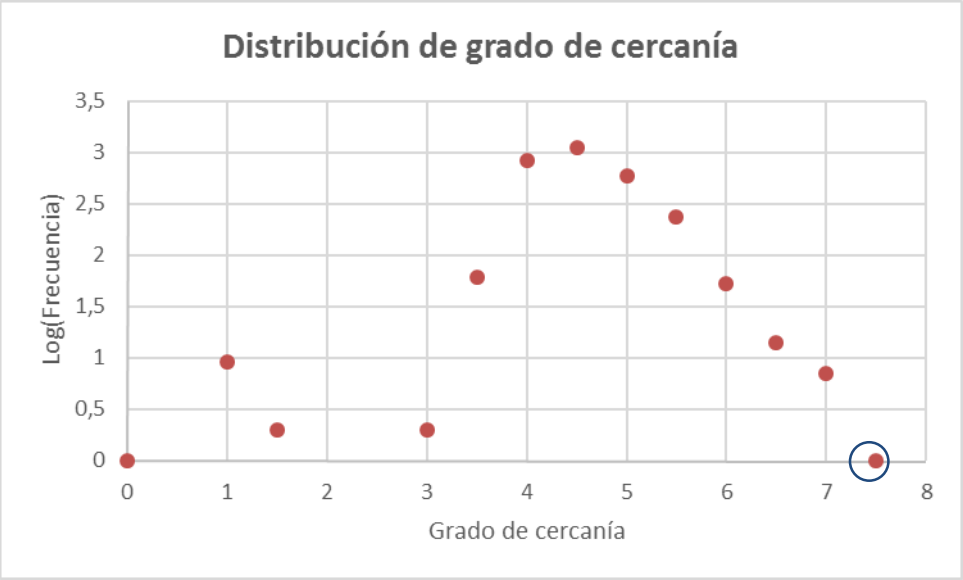

Ilustración 46: Distribución de centralidad de cercanía (*closeness centrality)* para el grafo *3, usando escala logarítmica para la frecuencia.*

- **Modularidad (*Modularity*):** En la **Ilustración 47**, se pueden ver las 30 comunidades detectadas con un grado de modularidad de 0,397, siendo 25 compuestas por más de dos nodos y una de ellas compuesta por 184 nodos. Como vemos, existe un número menor de comunidades, sin embargo, el valor de modularidad es más bajo que en el grafo 1, no siendo su valor tan significativo como en el caso de grafo 1.

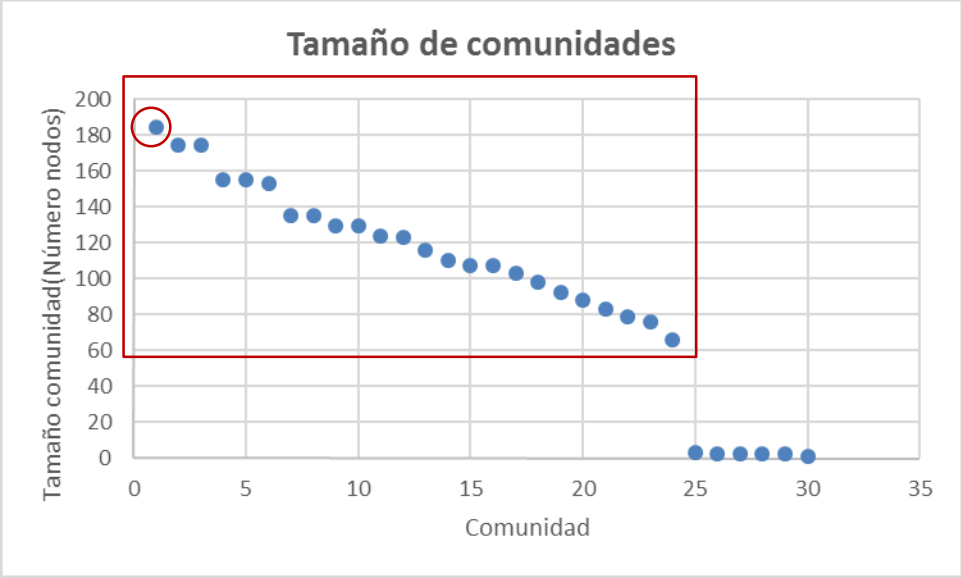

Ilustración 47: Tamaño de comunidades para el grafo 3.

- **Cliques:** Al realizar el análisis de cliques, el máximo cliqué tiene tamaño tres. Un análisis total del número de cliques, que se puede ver en **Ilustración 48**, muestra cómo debido a que no es un grafo tan altamente conectado como en el caso de los grafos 1 y 2, se generan cliques pero con un pico máximo en número de k igual a 2, correspondiente a simples pares de conexiones proteína-proteína.

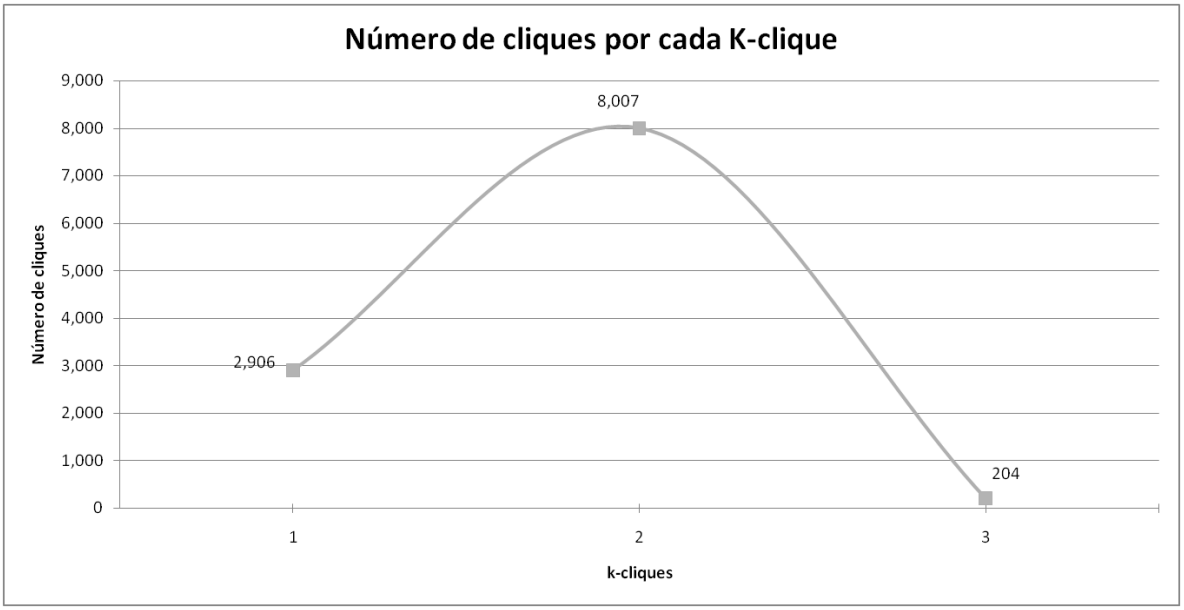

Ilustración 48: Número de cliques del grafo 3 por cada k-clique con k desde 1 hasta 3 que es el máximo clique.

5.5.2.4 Grafo no dirigido 4

Conjunto de datos

El conjunto inicial de datos corresponde a un experimento de microarreglos, el cual consta de 18 muestras, cada una de las cuales se representa por una fila, que tiene valores de reacción de los tratamientos para cada gen, representados en columnas; en primera instancia se calcula el coeficiente de variación (Ver 1.3.2.3.3) generándose la distribución mostrada en el **Ilustración 49**. Para nuestro caso se tomaron los datos más representativos, que corresponden a un corte del 75% en la distribución (Ver **Ilustración 50**). De éste forma se realiza un filtro para remover los genes con poca variación entre tratamientos. El umbral del filtro se fijó en 0,0298, aquellos genes con coeficientes de variación menores, se eliminaron.

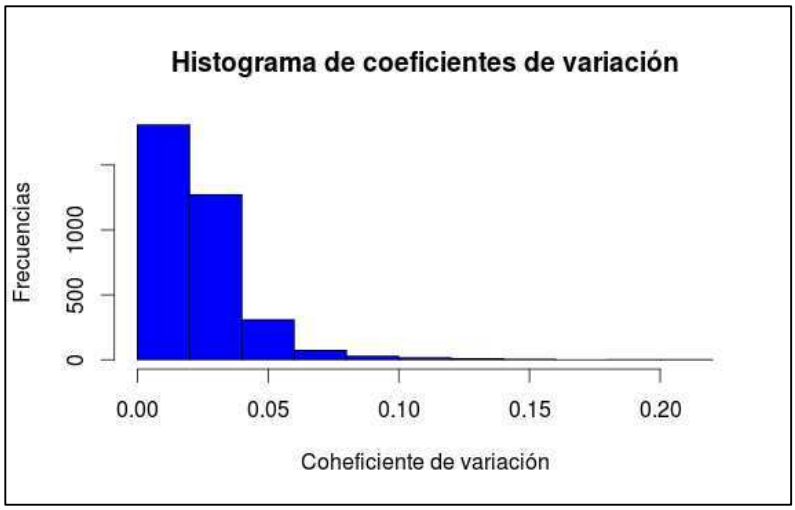

Ilustración 49: Distribución de coeficiente de variación, grafo 4.

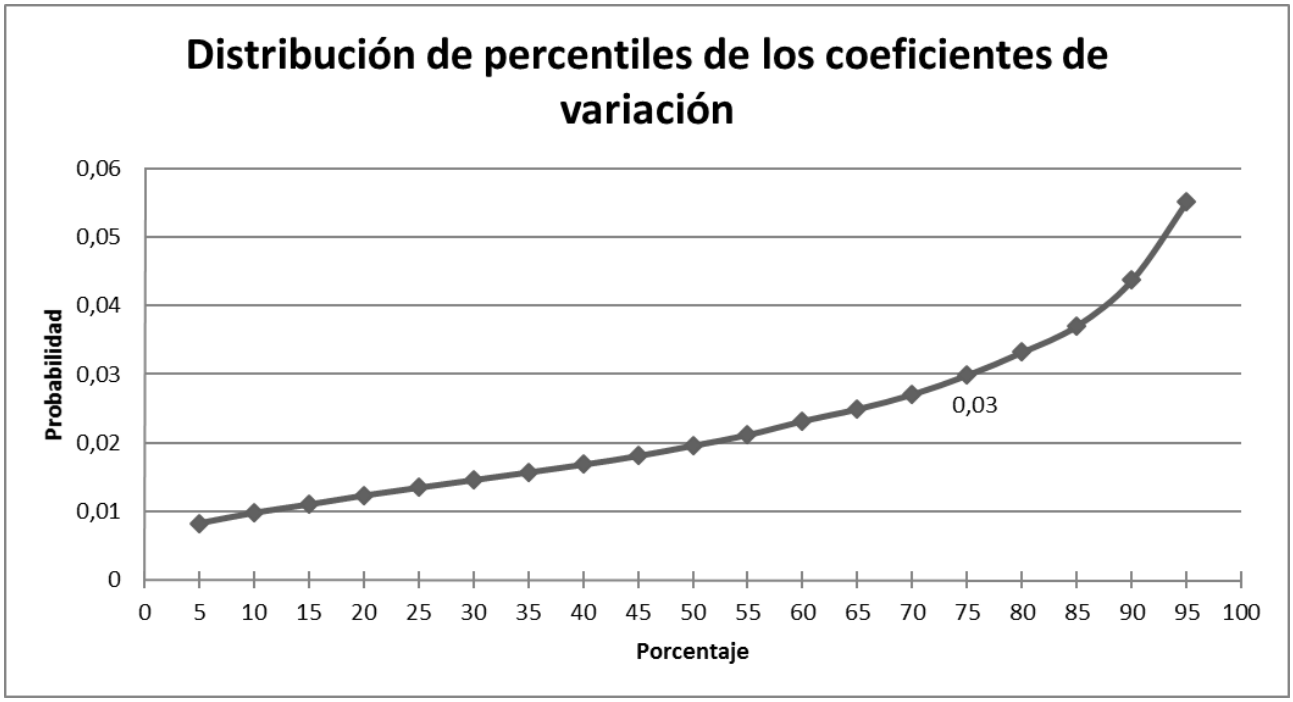

Ilustración 50: Distribución de percentiles de los coeficientes de variación para el grafo 4.

Seguidamente, se construye una matriz de información mutua, utilizando la correlación de Spearman como medida de asociación, la cual servirá como base para construir una matriz de adyacencia [54] [55] [56].

Se genera un conjunto de datos correspondiente al grafo, entre las columnas y filas de la matriz cuyo valor es distinto de cero; de ésta forma se obtienen 685 relaciones y 833 nodos.

En éste grafo en particular, el filtrado para resumir las relaciones repetidas y realizar agregación de información no tiene ningún efecto ya que no existen nodos o relaciones repetidas, que es lo esperado cuando la fuente de datos correspondiente es una matriz de adyacencias.

Para más información sobre las características generales del grafo ver la **Tabla *12***.

En la **Ilustración 43**, podemos ver que el grafo tiene muy baja conectividad entre sus nodos, presentándose nodos con grado menores o iguales a seis. Se experimentó con realizar un corte menor en el coeficiente de variación, sin embargo, el grafo seguía teniendo la misma arquitectura, generando eso sí, conexiones y representaciones de nodos con poca variación, por lo que se decidió tomar el ya mencionado valor de corte de 75%.

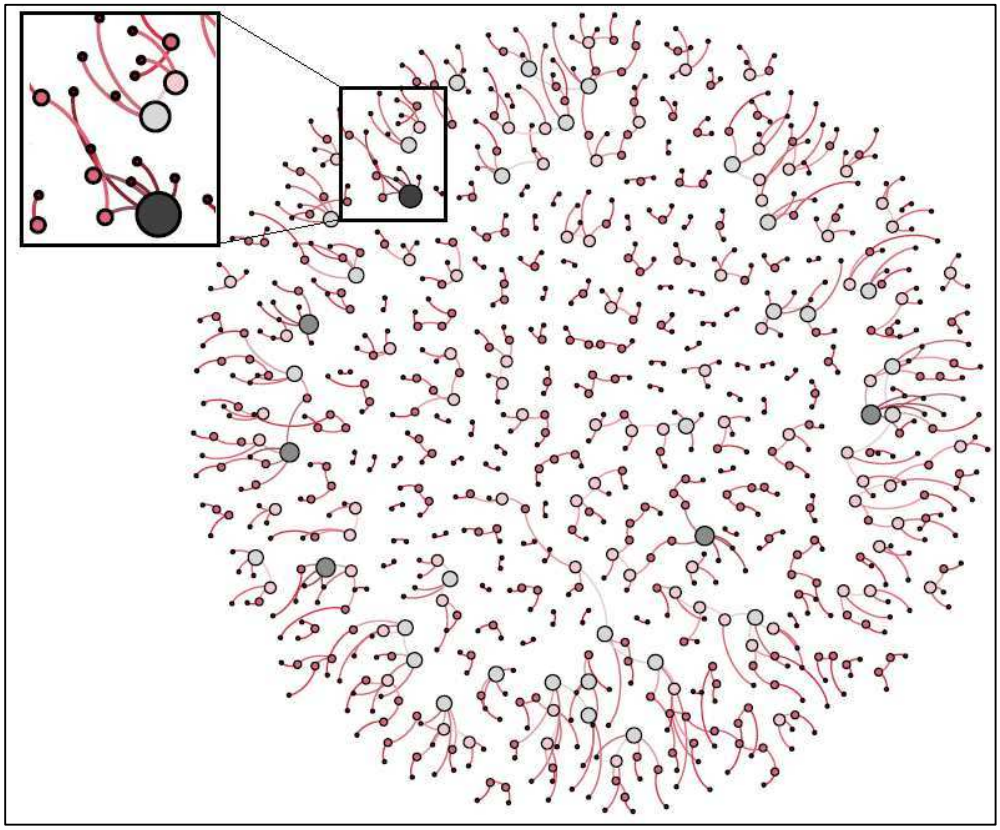

Ilustración 51: Grafo 4 con tamaño de nodos y color según el grado.

SNA (Social *Network Analysis*)

Para ver el resumen completo ver **Tabla *12*.**

- **Distribución de grado:** Se presentan nodos con grados máximos de 6, con un promedio de 1,58, siendo éste causado por la baja conectividad del grafo. El gen Rv2194 es el de mayor grado con 6, y los siguientes con grado 5 son: Rv2245, Rv0144, Rv1643, Rv0810c y Rv0287. Ver **Ilustración 52**. Para el caso de Rv2194, ha sido identificado como encargado de funciones respiratorias y de intermediación en el metabolismo [62], además de ser marcado como gen esencial para el crecimiento celular [67], todo esto podría explicar su alta conectividad y por ende el grado de importancia que presenta en el grafo.

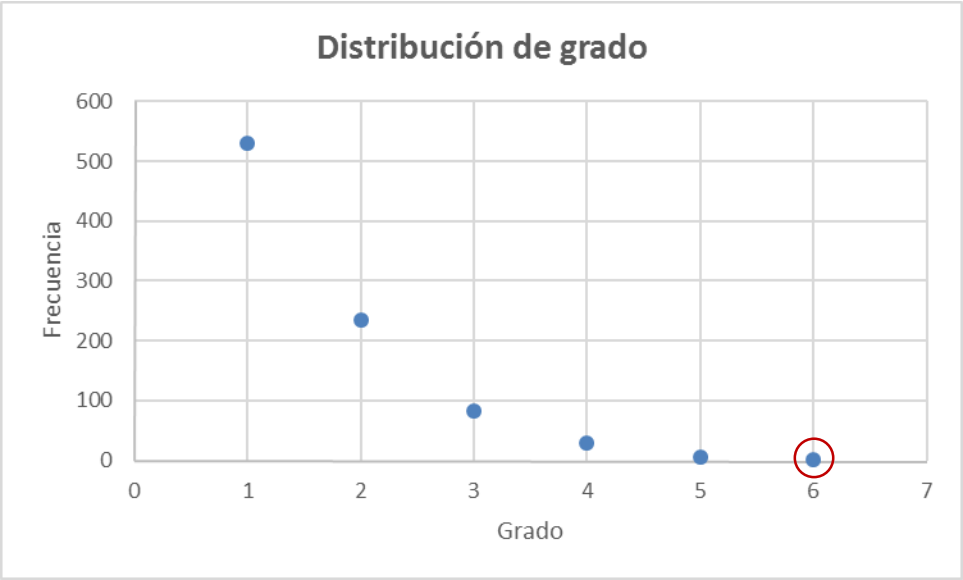

Ilustración 52: Distribución de grado de grafo 4.

- **Centralidad de intermediación (*Betweenness Centrality*):** En la **Ilustración 53** vemos que el nodo con máxima intermediación corresponde a la proteína Rv0298 con 143. También se observan valores de intermediación de hasta 80 en la mayoría de los nodos, lo que una vez más refleja la baja conectividad que existe en el grafo.

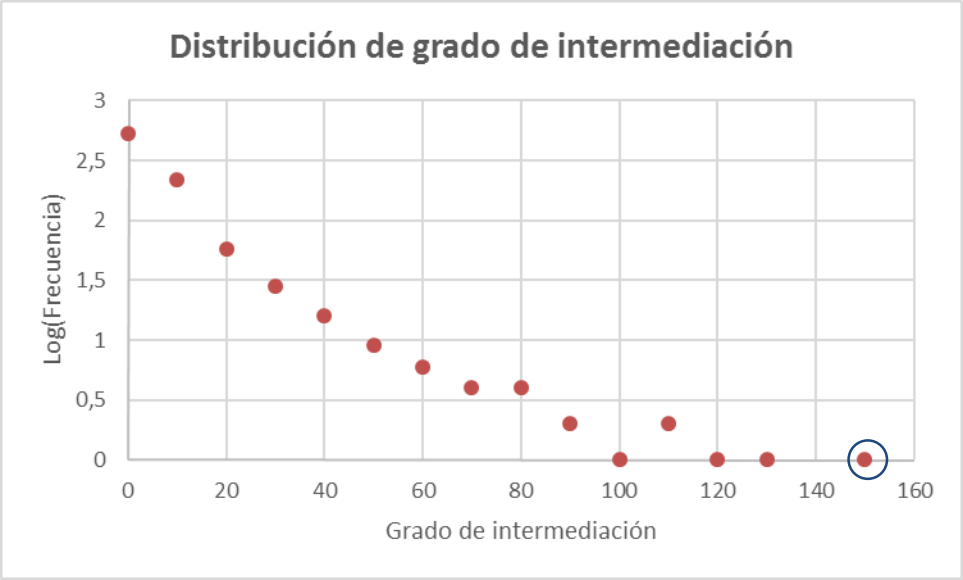

Ilustración 53: Distribución de centralidad de intermediación *(betweenness centrality)* para el grafo 4, usando escala logarítmica para la frecuencia.

- **Centralidad de cercanía (*Closeness Centrality*):** En la **Ilustración 54**, se muestra una aglomeración de nodos con medida de cercanía entre 1 y 6, existiendo algunos puntos *outliers* con respecto a la repetición del valor de la cercanía. El gen Rv2253 tiene el valor más alto de cercanía correspondiente a 6,82.

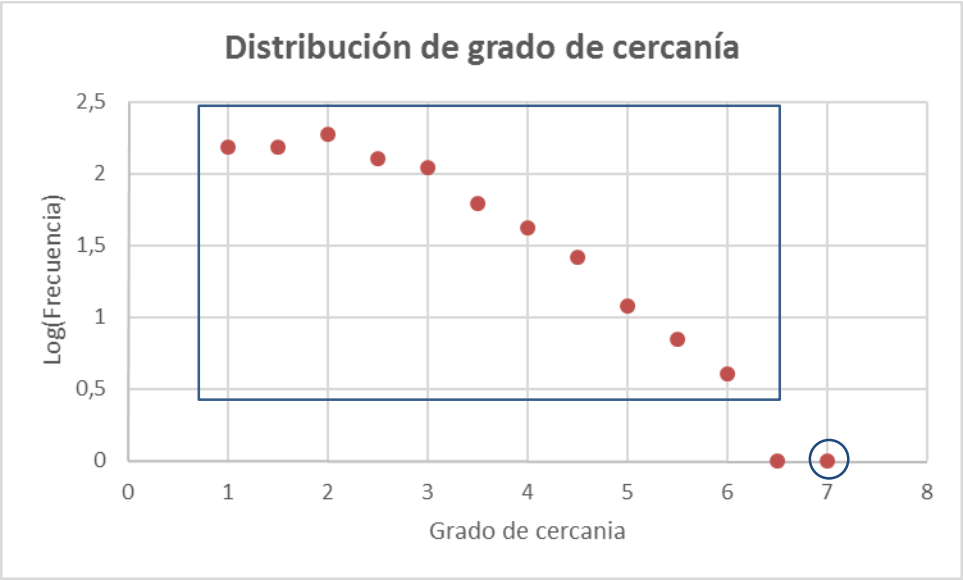

Ilustración 54: Distribución de centralidad de cercanía (*closeness centrality*) para el grafo *4, usando escala logarítmica para la frecuencia.*

- **Modularidad (*Modularity*):** En la **Ilustración 55**, se pueden ver las 184 comunidades detectadas con un grado de modularidad de 0,99, siendo 129 compuestas por más de 2 nodos. Se pueden ver entonces gran número de aglomeraciones de pocos nodos (teniendo la comunidad con más nodos, 22 de ellos), todas ellas desconectadas entre sí, por lo que la modularidad es muy alta.

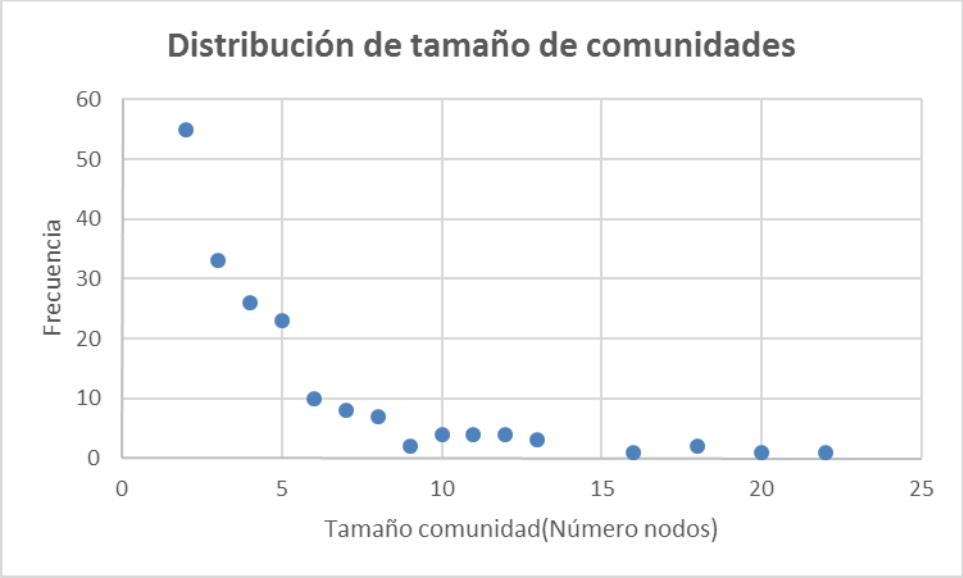

Ilustración 55: Distribución de tamaño de comunidades para el grafo 4.

- **Cliques:** Al realizar el análisis de cliques, solo existen en el grafo cliques con k=2, que representan una simple conexión entre dos nodos. Esto se debe a la poca interconexión entre nodos del grafo.

5.5.2.5 Grafo no dirigido 5

Conjunto de datos

Al igual que el conjunto de datos 4, corresponde a un experimento de microarreglos, el cual consta de 75 muestras, cada una de las cuales se representa por una fila, que tiene valores de reacción para cada gen, los cuales a su vez son columnas. Inicialmente existen dos tipos de plataformas usadas en las 75 muestras, donde la primera cuenta con 28 muestras y 1.711 datos y la segunda 47 muestras con 3.322 datos. Se elige el primer conjunto de muestras, dado que el segundo presenta un alto número de datos faltantes que no es recomendable imputar, ya que no se cuenta información sobre la causa de estos datos faltantes. Podría deberse a un problema experimental o a una falla de diseño del microarreglo.

En primera instancia se calcula el coeficiente de variación (Ver 1.3.2.3.3) generándose la distribución mostrada en la **Ilustración 56**. Para nuestro caso se tomaron los datos más representativos, que corresponden a un corte del 75% en la distribución (Ver **Ilustración 57**). De éste forma se realiza un filtro con valor de 5,506 que corresponde a filtrar los genes con poca variación entre los arreglos o muestras del conjunto de datos.

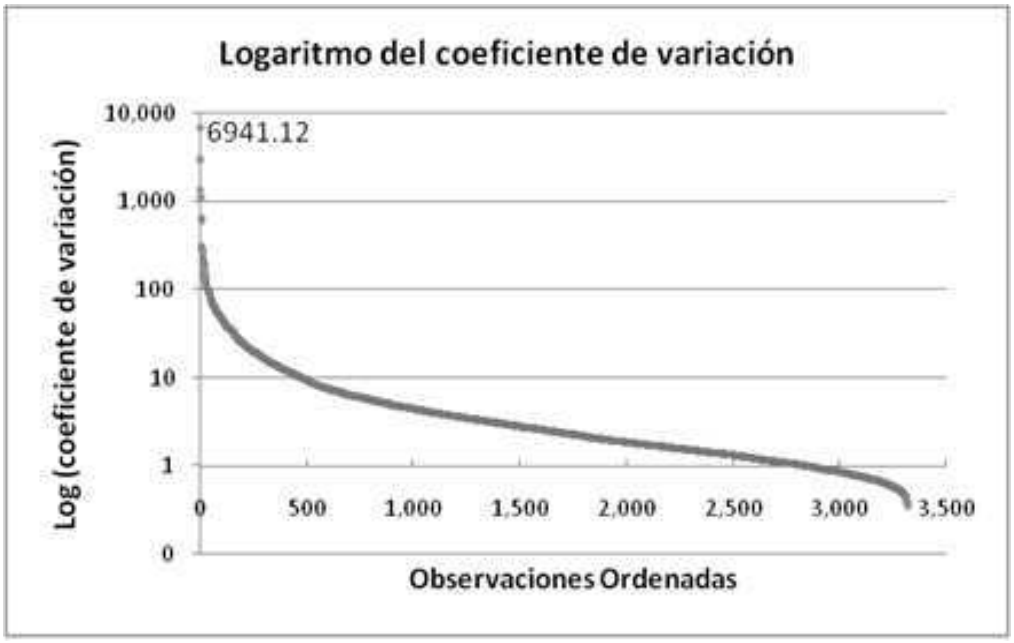

Ilustración 56: Distribución del logaritmo del coeficiente de variación, grafo 5.

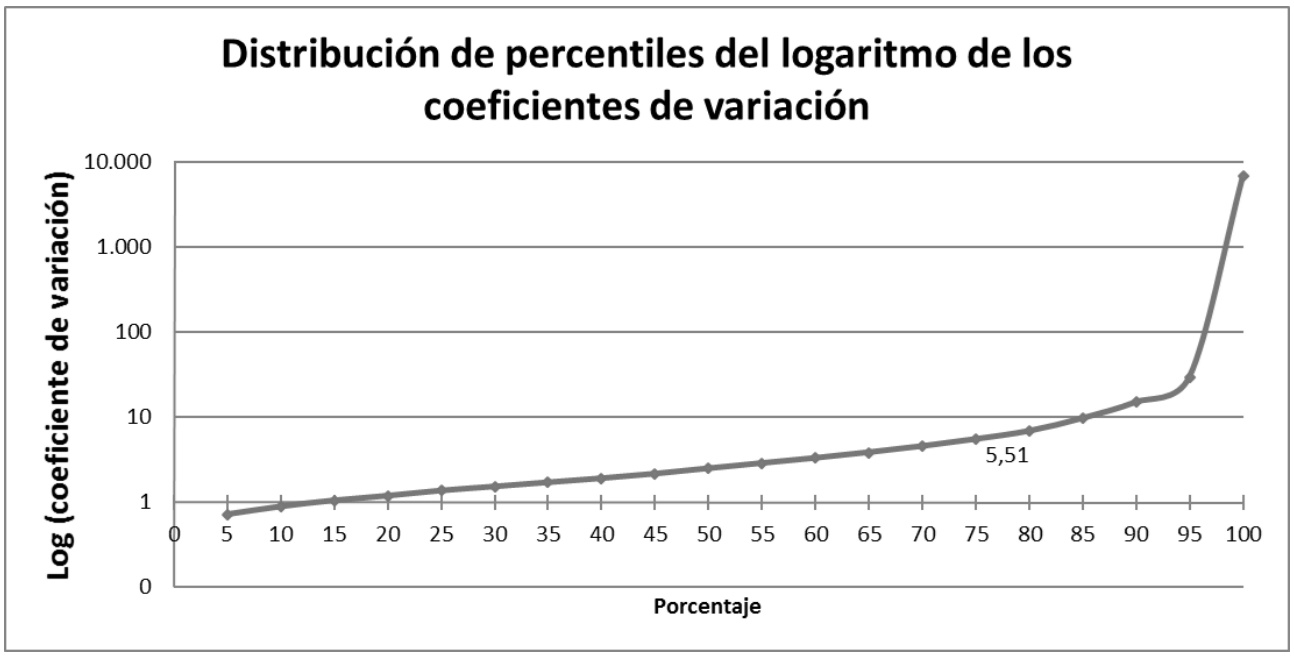

Ilustración 57: Distribución de percentiles del logaritmo de los coeficientes de variación para el grafo 5.

Seguidamente, se construye una matriz de información mutua utilizando el coeficiente de Spearman y a continuación se construye la matriz de adyacencia, tal como se hizo con el otro conjunto de datos de trascriptómica [54] [55] [56].

Se genera un conjunto de datos correspondiente al grafo, entre las columnas y filas de la matriz cuyo valor es distinto de cero; de ésta forma se obtienen 685 relaciones y 833 nodos.

En éste grafo en particular, el filtrado para resumir las relaciones repetidas y realizar agregación de información no tiene ningún efecto ya que no existen nodos o relaciones repetidas.

Para más información sobre las características generales del grafo ver la **Tabla *12***.

Viendo la **Ilustración 58**, podemos ver que el grafo tiene muy baja conectividad entre sus nodos, presentándose nodos con grado menores o iguales a 7. Se experimentó con realizar un corte menor en el coeficiente de variación; sin embargo, el grafo seguía teniendo la misma arquitectura, por lo que se decidió tomar el ya mencionado valor de corte de 75%.

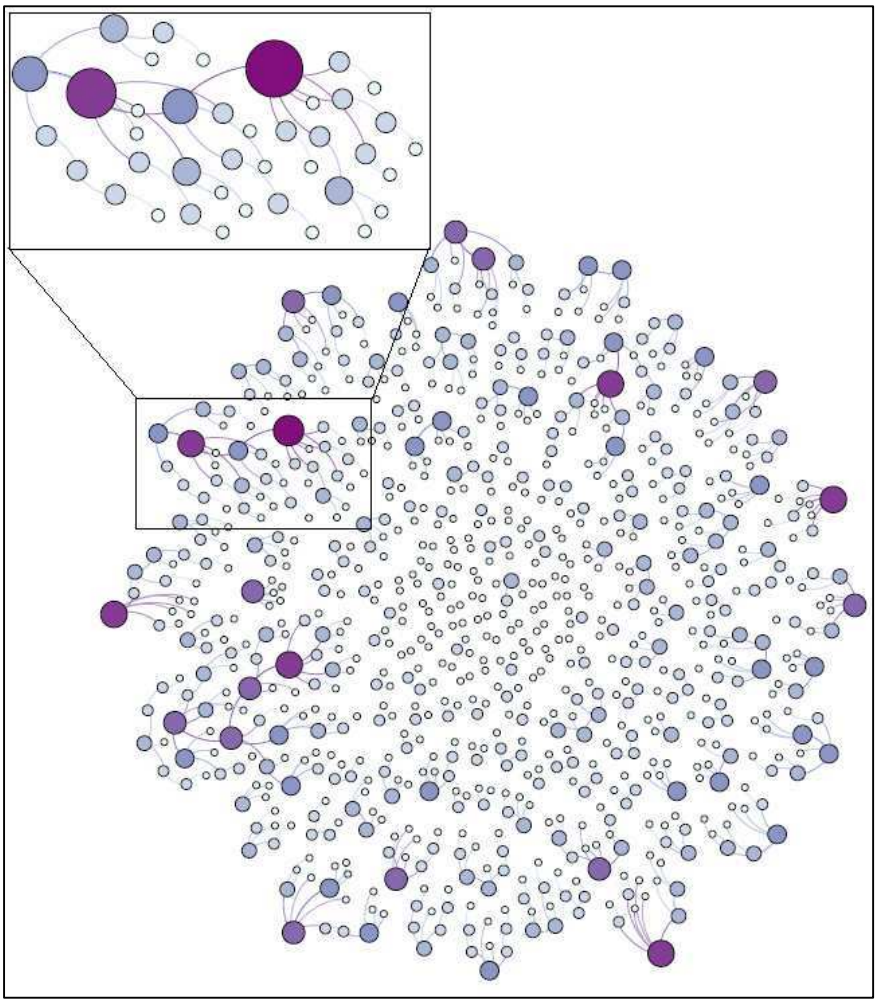

Ilustración 58: Grafo 5 con tamaño de nodos y color según el grado.

SNA (*Social Network Analysis)*

Para ver el resumen completo ver **Tabla *12* .**

- **Distribución de grado:** Se presentan nodos con grados máximos de 7, con un promedio de 1,64, siendo éste causado por la baja conectividad del grafo. El gen Rv3613c es el de mayor grado con 7, y los siguientes con grado 6 son: Rv1410c, Rv1376, Rv3506, Rv2509, Rv1419, Rv3261. Ver **Ilustración 59**.

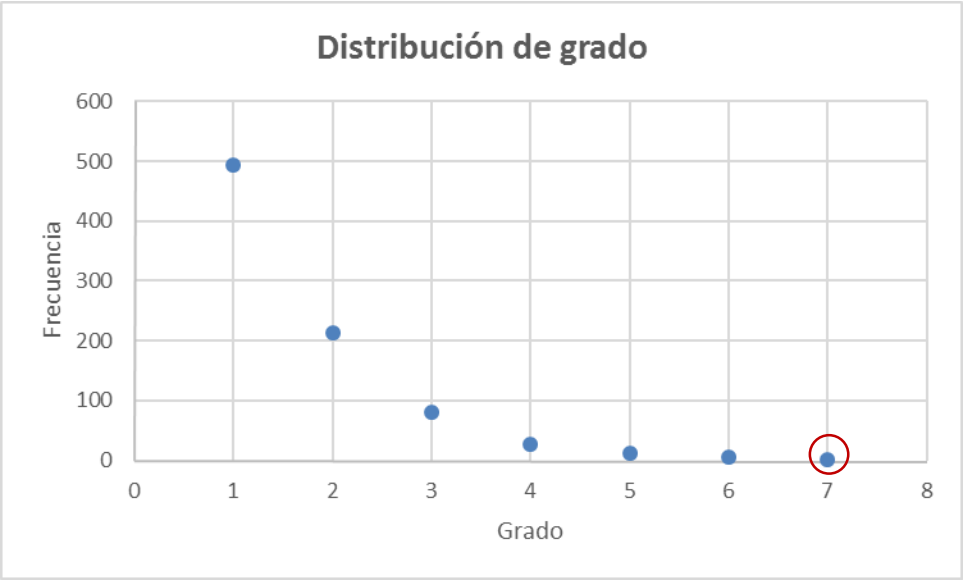

Ilustración 59: *Distribución de grado de grafo 5.*

- **Centralidad de intermediación (*Betweenness Centrality):*** En la **Ilustración *60*** vemos que el nodo con máxima intermediación corresponde a la proteína Rv2150c con 973, la cual ha sido identificada como esencial para la división celular, lo cual hace pensar que la concentración intracelular de la proteína FTSZ es crítica para la producción y formación del septum celular [68].

  También se observan valores de intermediación desde 100 en 26 proteínas, lo que una vez más refleja la baja conectividad que existe en el grafo dado el bajo valor de los demás nodos.

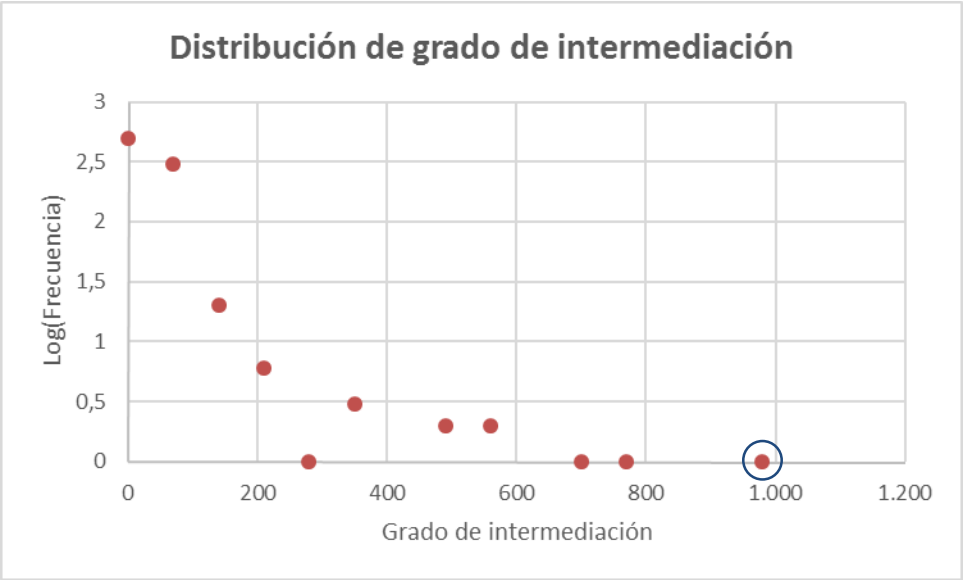

Ilustración 60: Distribución de centralidad de intermediación *(betweenness centrality)* para el grafo 5, usando escala logarítmica para la frecuencia.

- **Centralidad de cercanía *(Closeness Centrality)*:** En la **Ilustración 61**, se muestra una aglomeración de nodos con medida de cercanía entre 1 y 7, existiendo algunos puntos *outliers* con respecto a la repetición del valor de la

cercanía. El gen Rv3755c tiene el valor más alto de cercanía correspondiente a 6,94.

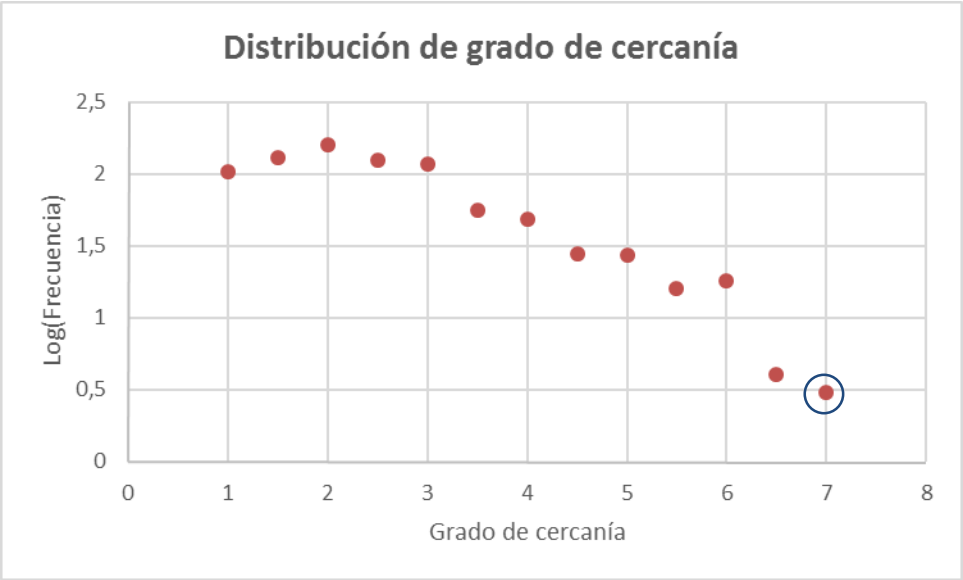

Ilustración 61: Distribución de centralidad de cercanía (*closeness centrality*) para el grafo 5, usando escala logarítmica para la frecuencia.

- **Modularidad (*Modularity*):** En la **Ilustración 62**, se muestran las 149 comunidades detectadas con un grado de modularidad de 0,98. Del total, 113 están compuestas por más de 2 nodos. Se observa un gran número de aglomeraciones de pocos nodos (la comunidad con más nodos, contiene 40), todas ellas desconectadas entre sí, por lo cual la modularidad es muy alta.

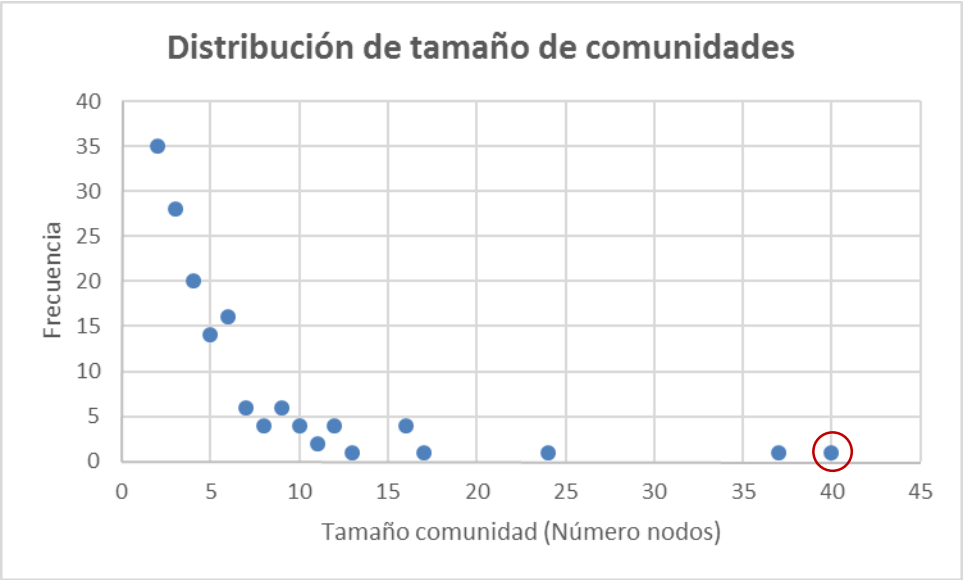

Ilustración 62: Distribución de tamaño de comunidades para el grafo 5.

- **Cliques:** Al realizar el análisis de cliques, al igual que el conjunto de datos 4, solo existe en el grafo cliques con k=2, lo cual se constituye como una simple conexión entre dos nodos. Esto se debe a la poca interconexión entre nodos del grafo.

5.6 MULTIGRAFO: CONSTRUCCIÓN Y ANÁLISIS

5.6.1 Construcción de multigrafo

Para la construcción del multigrafo se llevó a cabo el proceso mostrado en la **Ilustración 63**. A continuación, se explica cada paso:
c

1. Se partió de tres tipos de conjuntos de datos - como ya se describió antes - generados en *M. tuberculosis*, sumando cinco en total.

2. Luego de un análisis exploratorio inicial se plantearon una serie de procedimientos ETL (*Extraction, Transformation, Loading*) en donde se realizaron depuraciones de datos, agregación de información en nodos y relaciones y transformaciones de tipos de datos de atributos. Todo éste proceso se realiza inicialmente sobre los cinco conjuntos de datos y se detalla en las secciones anteriores

3. Como un paso posterior se realizan sobre los grafos generados los siguientes procedimientos:
   - Adición de información mediante el servicio web proporcionado por string-db [60], se agregan los atributos PREFERED_NAME y ANNOTATION a los nodos, buscados a partir del nombre del ID de cada proteína.
   - Manipulación de datos a través de scripts python. Se realizan transformaciones simples de archivos de datos iniciales para compatibilizar su formato con herramientas como: Neo4j, Titan y gephi. Específicamente se corrige el uso de comillas dobles en los campos de texto, se cambian los separadores de columnas, se modifica para mejor legibilidad el orden de algunas columnas y se generan archivos intermedios con los nodos del multigrafo para su correcta importación a las bases de datos Neo4j y Titan.
   - Adición de información mediante consulta a bases de datos externas relacionales.
   - Generación de análisis SNA en distintos lenguajes como java y R.

   Aunque en nuestro caso no se realiza un uso de BIO4J [59] para realizar enriquecimiento de datos desde la base no relacional biológica, se plantea como una posible fuente de información para ser usada en proyecto similares (Ver recuadro rojo punteado en la **Ilustración 63** en la parte derecha inferior).

4. En procesos paralelos se realiza una carga como parte final del punto anterior, en dos bases de datos no relacionales orientadas a grafos: NEO4j y Titan DB mediante un script de importación, generando estructuras de datos no relacionales de tipo **clave:valor** tanto en las relaciones como en los nodos.

5. Finalmente, y luego de tener los grafos por separado en ambas bases de datos no relacionales, se realiza un proceso de conversión en los que los cinco grafos se unen en un multigrafo mixto (aristas dirigidas y no dirigidas) compartiendo nodos y generándose distintos tipos o categorías de relaciones descritas anteriormente para cada grafo.

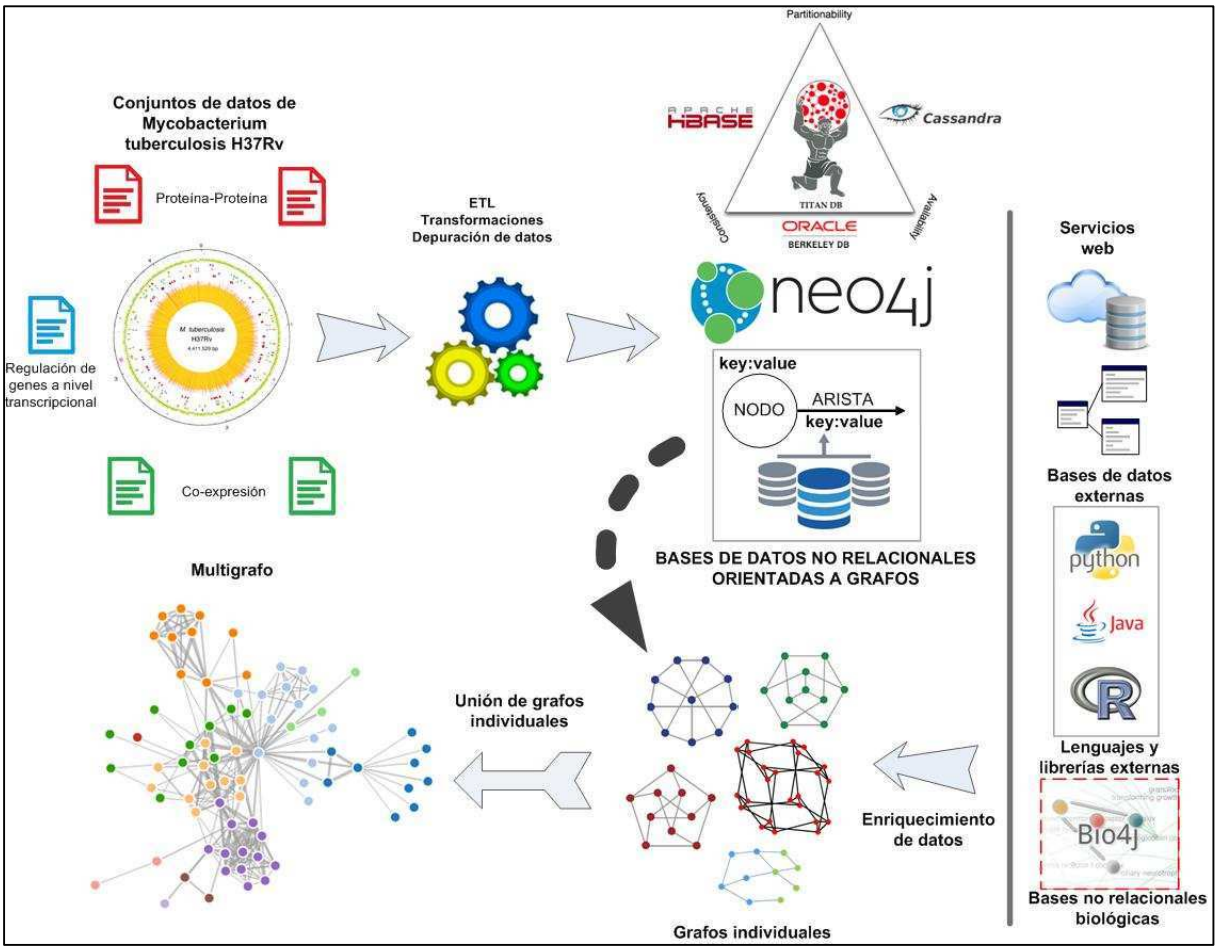

Ilustración 63: Proceso de enriquecimiento de datos para construcción de multigrafo.

5.6.2 Composición de datos de los nodos

Debido a que el multigrafo se construye mediante la combinación de los grafos 1 al 5, éste hereda las propiedades de sus nodos (Ver sección 3.5.1). Adicionalmente se agregan tres medidas de SNA a cada nodo con el fin de enriquecer la información y permitir así generar consultas más específicas y un entendimiento más general del multigrafo.

Información adicional del nodo en el multigrafo:

- **BETWEENNESS_CENTRALITY:** corresponde al valor de la medida de centralidad de intermediación.
- **CLOSENESS_CENTRALITY:** corresponde al valor de la medida de centralidad de cercanía.
- **DEGREE_CENTRALITY:** corresponde al valor de la medida de centralidad de grado del nodo.

5.6.3 Composición de datos de las relaciones

En cuanto a los atributos de los grafos, se heredan todos los definidos en los grafos 1-5. (Ver sección 3.5.2).

5.6.4 Conjunto de datos

El conjunto de datos correspondiente al multigrafo se generó a partir de los grafos parciales anteriormente analizados (1-5), obteniendo de éste manera 24.016 relaciones y 3.653 nodos.

Para más información sobre las características generales del grafo ver la **Tabla *12***.

Por otro lado, en la **Ilustración 64** podemos observar que el multigrafo es altamente conectado, lo cual era de esperarse dado que es la composición de los cinco grafos iniciales. Vemos como existen nodos con altos grados, en el caso de la ilustración ampliada se muestra a la proteína/gen Rv3416 que tiene un valor de 196 y en la parte inferior la proteína Rv0350 con el máximo valor de grado correspondiente a 200.

En primera instancia vemos que se presentan claramente tres regiones correspondientes a los grafos 1, 2 y 3 (magenta, azul y verde, respectivamente), los cuales tienen mayor número de interconexiones y nodos. En cuanto a los grafos 4 y 5, se pueden ver un menor número de conexiones que se encuentran uniformemente distribuidas a lo largo del multigrafo.

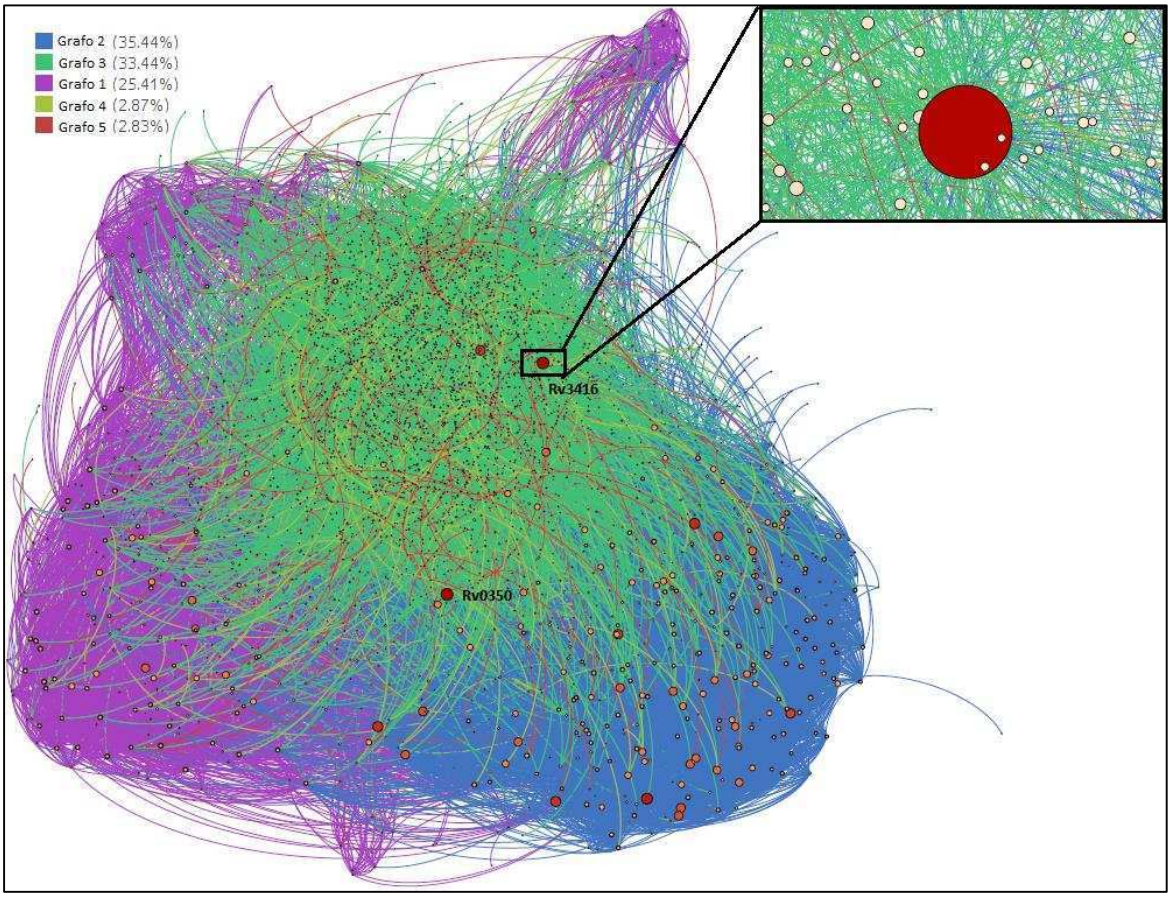

Ilustración 64: Multigrafo con tamaño de nodos y color según el grado en escala de rojos y grafos 1-5 que lo componen con colores diferenciados.

Para identificar mejor cómo se comportan los nodos y relaciones de cada tipo de grafo en el multigrafo, se visualizan en distintas ilustraciones cada uno de ellos, así, en la

Ilustración 65 vemos solo los nodos y aristas correspondientes a los datos de interacciones proteína-proteína, (grafos 1 y 3). Podemos identificar varios súper nodos [39] correspondientes a cada subgrafo (ver nodos con círculos rojos en la ilustración). Sin embargo, se puede notar que entre ellos comparten un gran número de proteínas y relaciones, lo que es esperable debido a que ambos grafos tienen propiedades similares y son construidos mediante datos del mismo tipo.

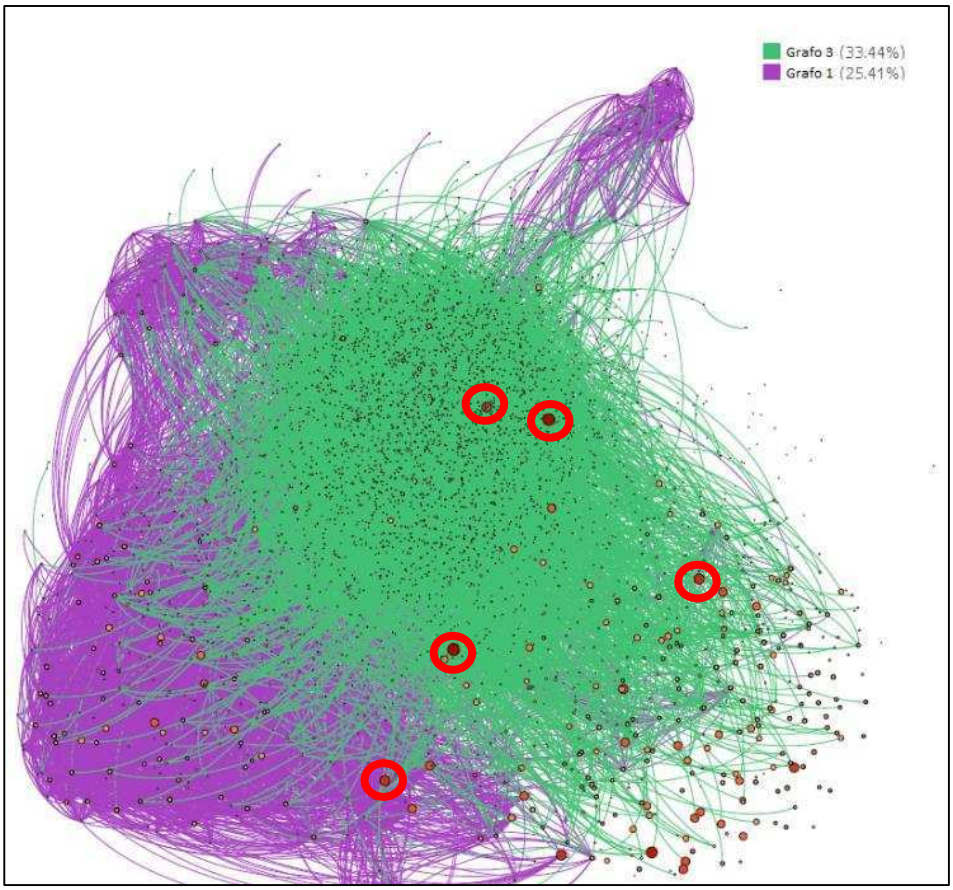

Ilustración 65: Multigrafo con componentes de **datos de tipo 1: Proteína-Proteína** con tamaño de nodos y color según el grado en escala de rojos y grafos 1 y 3 que lo componen con colores diferenciados.

Para el caso de datos correspondientes a regulación de la expresión de genes(interacciones proteína – zona regulatoria, grafo 2), vemos en la **Ilustración 66** varios súper nodos [39] formados, con gran cantidad de conexiones hacia los nodos de grafos generados con datos de interacciones proteína-proteína y también nodos con alto grado.

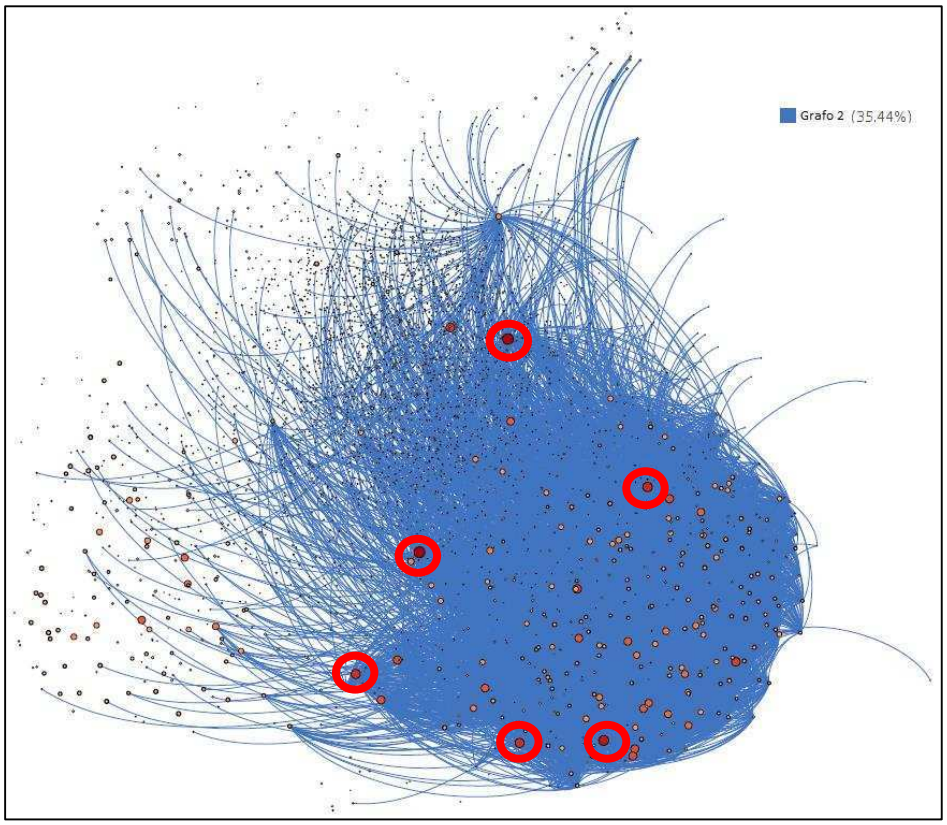

Ilustración 66: Multigrafo con componentes de **datos de tipo 2: regulación de la expresión de genes** con tamaño de nodos y color según el grado en escala de rojos y grafo 2 que lo compone de color azul.

Finalmente, para el caso de grafos de coexpresión, es decir, los correspondientes a datos de expresión génica en microarrays (grafo 4 y 5), vemos que existe una zona central de acumulación, sin embargo, sus conexiones y nodos relacionados están ubicados en todo el multigrafo de una manera casi homogénea. Ver **Ilustración 67**.

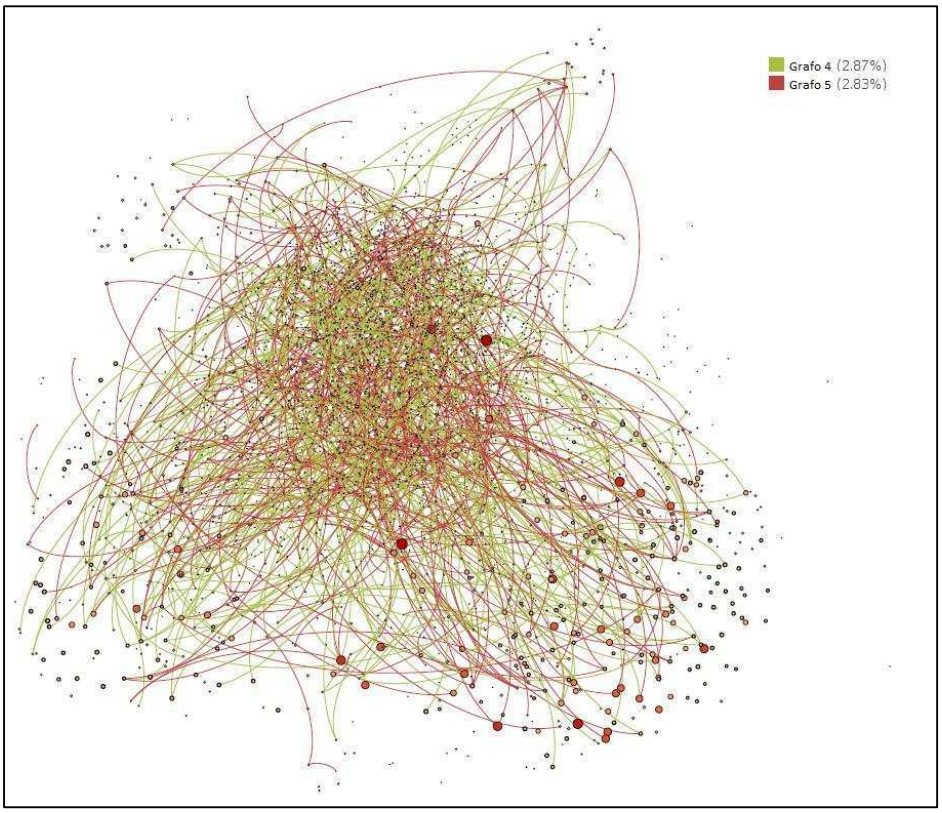

Ilustración 67: Multigrafo con componentes de **datos de tipo 3: Expresión génica** con tamaño de nodos y color según el grado en escala de rojos y grafos 4 y 5 que lo componen con colores diferenciados.

5.6.5 SNA (*Social Network Analysis)*

Para ver el resumen completo ver **Tabla *12*.**

- **Distribución de grado:** Se presentan nodos con grados máximos de 200 y con un promedio de 13,1. Para más detalle ver de las diez partículas (proteínas / genes) con mayor grado ver la **Tabla 11**. Un 32,80% de los nodos tienen un grado mayor o igual a 10 y tan solo un 10,83% tiene valor de 1 en su grado. Ver **Ilustración 68**.

Proteína	Grado
Rv0350	200
Rv3416	196
Rv1219c	181
Rv1990c	171
Rv0702	165
Rv1218c	165
Rv2773c	157
Rv3290c	154
Rv2034	154
Rv0232	146

Tabla 11: Listado de las 10 proteínas con máximo grado en multigrafo.

Así mismo, de las 10 proteínas con grados más altos, la mitad son reguladores transcripcionales (Rv3416, Rv1219c, Rv1990c, Rv2034, Rv0232), además, otros dos son relacionados con la intermediación entre el metabolismo y la respiración, funciones vitales en el organismo celular (Rv2773c, Rv3290c) [62]. Por otro lado, es importante destacar que Rv3416 tiene un mayor grado de centralidad en el multigrafo que en cualquiera de los grafos individuales, por ejemplo, tiene un 59% más grado de centralidad que en el grafo individual 3.

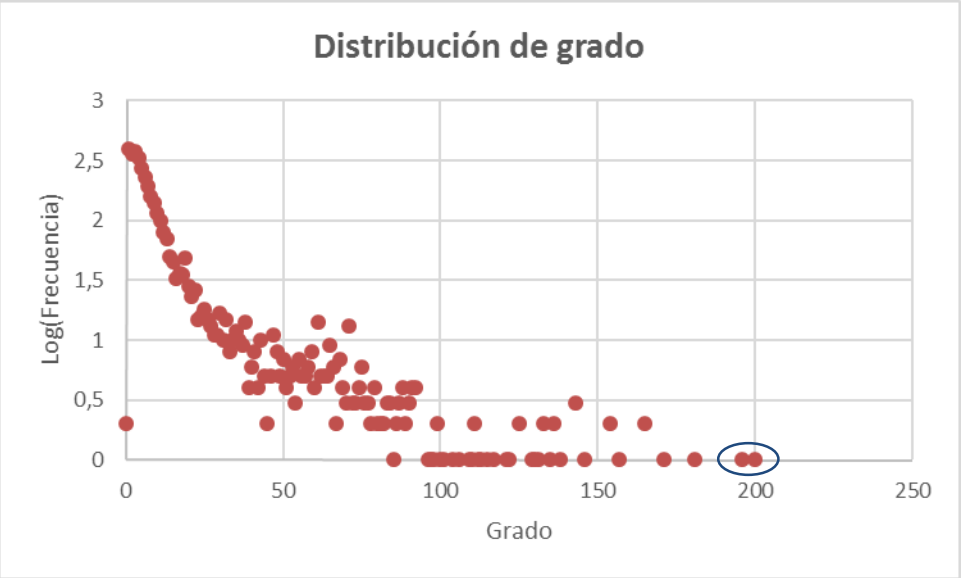

Ilustración 68: Distribución de grado del multigrafo, usando escala logarítmica para la frecuencia.

- **Centralidad de intermediación (*Betweenness Centrality):*** En la **Ilustración 69** vemos que el nodo con máxima intermediación corresponde al gen/proteína Rv3416 con 368.287,56, siendo un valor en el mismo orden de magnitud de los máximos presentados en los grafos 3 y 4 de manera individual. Además 3.207 de los nodos tienen un valor de intermediación distinto de cero, es decir, un 87,74% del total, un valor esperable dado lo altamente conectado del multigrafo.

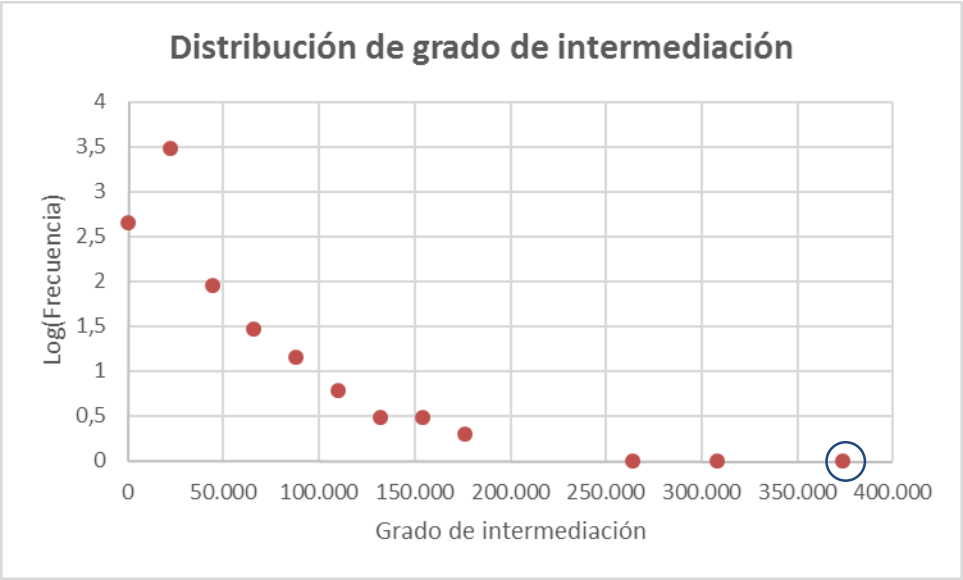

Ilustración 69: Distribución de centralidad de intermediación *(betweenness centrality)* para el multigrafo, usando escala logarítmica para la frecuencia.

- **Centralidad de cercanía (*Closeness Centrality*):** En la **Ilustración 70**, se muestra una aglomeración de nodos con medida de cercanía entre 3 y 6, existiendo puntos *outliers* en donde se tiene 0, 1 y 4 grado de cercanía aproximadamente. Los nodos con grados de cercanía mayores a 5,53 son Rv3155, Rv3059, Rv3360, Rv1092c y Rv2627c, siendo el primero el mayor grado con 5,90 de valor de cercanía.

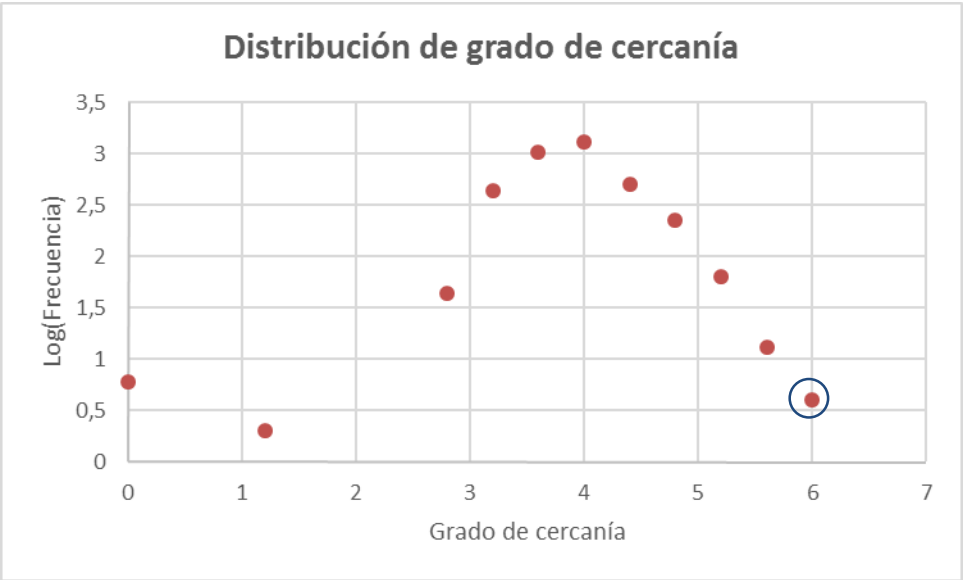

Ilustración 70: Distribución de centralidad de cercanía *(closeness centrality)* para el multigrafo, usando escala logarítmica en la frecuencia.

- **Modularidad (*Modularity*):** En la **Ilustración 71**, se pueden ver las 15 comunidades detectadas con un grado de modularidad de 0,481, tan solo siete están compuestas por más de dos nodos y una de ellas compuesta por 2.203 nodos. Como vemos, existe un número menor de comunidades, sin embargo, el valor de modularidad es parecido al de los grafos 1 y 3.

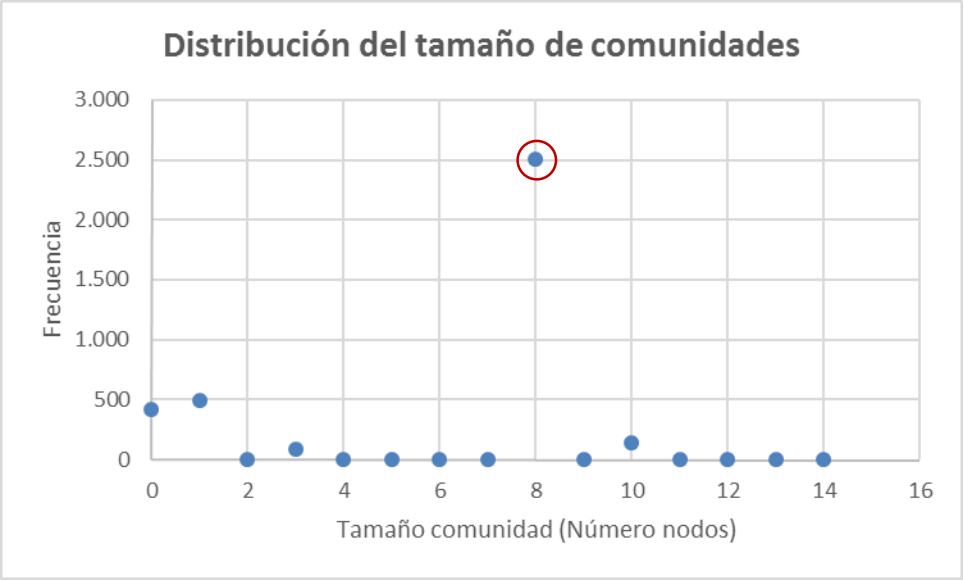

Ilustración 71: Distribución del tamaño de comunidades para el multigrafo.

- **Cliques:** Al realizar el análisis de cliques, el máximo cliqué es de tamaño 25, igualando a lo presentado en el grafo 1, lo que demuestra una vez más lo altamente conectado que es el multigrafo y se refleja la herencia de conectividad que pasa el grafo 1 al multigrafo. Un análisis inicial del número de cliques para los primeros nueve k-cliques muestra un crecimiento acelerado en cada k-clique (Ver **Ilustración 72**), también reflejado en el tiempo necesario para el procesamiento de búsqueda de los mismos. Éste comportamiento es similar al analizado en el grafo 1, en la **Ilustración 36**.

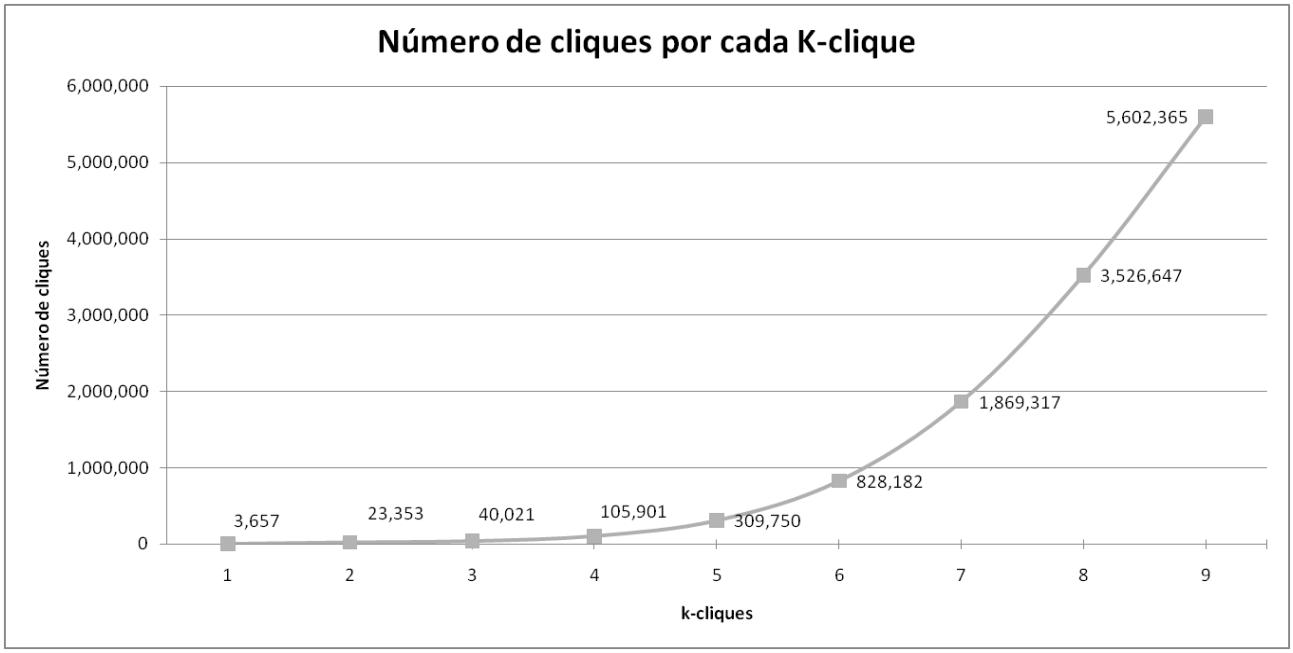

Ilustración 72: Número de cliques del multigrafo por cada k-clique con k desde 1 hasta 9.

5.7 IMPORTACIÓN DE DATOS A BASES DE DATOS DE GRAFOS Y EXPORTACIÓN DE GRAFOS A FORMATO GRAPHML

Para la importación de los datos a las distintas bases de datos se generó un único conjunto de datos compuesto de los siguientes archivos de tipo CSV separados por coma:

1. Un archivo con todos los distintos nodos o PARTICLES (partículas, es decir, la combinación gen y proteína codificada) y sus atributos básicos.
2. Un archivo que contiene distintas medidas SNA para cada nodo o PARTICLE en el multigrafo.
3. Por cada subgrafo un archivo de relaciones que tiene los atributos asociados de acuerdo al conjunto de datos determinado, es decir, cinco archivos.

Para cada caso se usaron las estrategias optimizadas para importación masiva de archivos. Sin embargo – y como se mencionará más adelante –, según la plataforma se presentan limitaciones que en determinados casos hace posible plantear un diseño de importación más eficiente para cada plataforma en particular.

En cuanto a la exportación de los grafos (subgrafos y multígrafos) y consultas se eligió el formato estándar graphML dada su amplia implementación en diversas herramientas tanto de diagramación como manipulación de grafos; en el caso del presente trabajo, Neo4j, Titan DB y gephi tiene soporte para dicho formato.

5.7.1 Importación a NEO4J

Para este caso, la importación se realizó mediante lenguaje CYPHER y usando la cláusula LOAD CSV optimizada para importación masiva de archivos. A continuación, se detallarán cada una de las importaciones de los diferentes archivos:

a) **Importación de nodos individuales con atributos básicos.**

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
CREATE (n:PARTICLE {ID : line[0], PREFERED_NAME : line[1], NCBI_TAXON_ID : line[2], TAXON_NAME : line[3],
ANNOTATIONS : line[4]})
```

b) **Importación de atributos SNA del multígrafo para cada nodo**. Por cada línea se busca un nodo y se agregan tres atributos SNA actualizándolo.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MATCH (n:PARTICLE)
WHERE n.ID = line[0]
SET n.DEGREE_CENTRALITY = line[4], n.CLOSENESS_CENTRALITY = line[9], n.BETWEENNESS_CENTRALITY =
line[10];
```

c) **Importación de subgrafo número 1**.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MATCH (n:PARTICLE),(m:PARTICLE)
WHERE n.ID = line[0] AND m.ID = line[1]
MERGE (n)-[r:INTERACTION_PROTEINES_1 {INTERACTION_TYPE : "PROTEIN-PROTEIN", PARTICLE_A :
"PROTEIN", PARTICLE_B : "PROTEIN", DIP_INTERACTION_ID : line[2], DETECTION_METHODS : line[3],
PUBLICATIONS : line[4], FOUND : "DATASET_1",
COUNTER_REPITED: line[5], STD_COUNTER_REPITED: line[6]}]-(m);
```

d) **Importación de subgrafo número 2.** A diferencia de los demás 4 grafos que son no dirigidos, en éste se plantea una relación entre nodos de tipo dirigida, especificada mediante el operador: “->”.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MERGE (n:PARTICLE {ID : line[0]})
MERGE (m:PARTICLE {ID : line[1]})
MERGE (n)-[r:INTERACTION_REGULATION_1 {INTERACTION_TYPE : "PROTEIN-GEN", PARTICLE_A : "PROTEIN",
PARTICLE_B : "GEN", TARGET_PROMOTER : line[3], PROMOTES_COORDINATES_GENOMA : line[4],
GENE_IN_OPERON : line[5], PROMOTER_SEQUENCE : line[6], FOUND : "DATASET_2",
COUNTER_REPITED: line[7], STD_COUNTER_REPITED: line[8]}]->(m);
```

e) Importación de subgrafo número 3.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MERGE (n:PARTICLE {ID : line[0]})
MERGE (m:PARTICLE {ID : line[1]})
MERGE (n)-[r:INTERACTION_PROTEINES_2 {INTERACTION_TYPE : "PROTEIN-PROTEIN", PARTICLE_A :
"PROTEIN", PARTICLE_B : "PROTEIN", FOUND : "DATASET_3", GENE_INTERACTOR_A:line[2],
GENE_INTERACTOR_B:line[3], ANNOTATION_INTERACTOR_A:line[4],ANNOTATION_INTERACTOR_B:line[5],
COUNTER_REPITED: line[6], STD_COUNTER_REPITED: line[7]}]-(m);
```

f) Importación de subgrafo número 4.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MERGE (n:PARTICLE {ID : line[0]})
MERGE (m:PARTICLE {ID : line[1]})
MERGE (n)-[r:INTERACTION_CO_EXPRESSION_1 {INTERACTION_TYPE : "GEN-GEN_COEXPRESSION", PARTICLE_A :
"GEN", PARTICLE_B : "GEN",FOUND : "DATASET_4",
COUNTER_REPITED: line[3], STD_COUNTER_REPITED: line[4]}]-(m);
```

g) Importación de subgrafo número 5.

```
LOAD CSV FROM '[Ruta del archivo correspondiente]' AS line
MERGE (n:PARTICLE {ID : line[0]})
MERGE (m:PARTICLE {ID : line[1]})
MERGE (n)-[r:INTERACTION_CO_EXPRESSION_2 {INTERACTION_TYPE : "GEN-GEN_COEXPRESSION", PARTICLE_A :
"GEN", PARTICLE_B : "GEN",FOUND : "DATASET_5",
COUNTER_REPITED: line[3], STD_COUNTER_REPITED: line[4]}]-(m);
```

5.7.1.1 Consideraciones adicionales en NEO4J

- Al realizar la importación del segundo archivo con los atributos SNA de cada nodo, se genera una búsqueda y actualización por cada línea, lo cual conduce a una posible ineficiencia del script. Específicamente en NEO4J esta operación está optimizada de manera nativa a diferencia de otras plataformas. Para importaciones más masivas, lo más recomendable es agregar al archivo de nodos la información SNA y de cualquier otra. Para importaciones del orden de millones de registros, las utilidades como *neo4j-import*, son las más recomendables.
- Los atributos de tipo texto se deben configurar en los archivos usando comillas para evitar importaciones parciales en aquellos textos que contengan comas o separadores especiales.

5.7.2 Exportación de grafos a formato graphML en NEO4J

Mediante la herramienta *neo4j-shell* y haciendo uso de la librería adicional *neo4j-shell-tools_2.2*, se ejecutó el siguiente patrón de comando para cada una de las exportaciones, tanto de subgrafos individuales, como el multígrafo y consultas individuales de interés.

```
export-graphml -r -o [Nombre Archivo].graphml [QUERY CYPHER]
```

Específicamente para la exportación del multígrafo se uso:

```
export-graphml -r -o multigraph_neo4j.graphml match (n)-[r]->()
```

5.7.3 Importación en TITAN DB

Para este caso, la importación se realizó mediante el lenguaje *gremlin-groovy* y usando una combinación de las claúsula ***new File*** y ***graph.tx().onReadWrite*** optimizada para importación masiva de archivos con *commit* de datos en transacciones; adicionalmente se agregan los índices necesarios para la ejecución de las consultas sobre los grafos y el multigrafo.

El modelo de datos se pudo construir según el diseño presentado anteriormente, salvo el atributo ID de los nodos, que es atributo reservado en Titan DB, por lo cual se lo reemplazó por el atributo ID_.

Al igual que en el caso de la importación en Neo4j, primero se hace la importación de los nodos individuales, luego se adicionan los atributos SNA a dichos nodos y finalmente se importan cada uno de los cinco archivos correspondientes a los subgrafos que componen el multigrafo. Para mayor detalle de los procedimientos ver el **ANEXO 1: Scripts de importanción de multigrafo a Titan DB**.

5.7.3.1 Consideraciones adicionales en TITAN

- Todas las importaciones realizadas en *gremlin* se hicieron usando el lenguaje *groovy*, el cual está soportado nativamente y enriquece altamente las consultas y operaciones en los grafos.
- Para la versión 1.0.0 de Titan DB se modificaron los métodos para realizar importaciones masivas, introduciendo el concepto de transacciones de grafos (*graphTransactions*); esto condujo a una mejora evidente en los tiempos de importación tanto en los archivos de grafos como en los datos de SNA.
- Los atributos de tipo texto se deben configurar en los archivos usando comillas para evitar importaciones parciales en textos que contengan comas o separadores especiales. Para el caso de gremlin, se hace uso de expresiones regulares para generar un *split* de las líneas de los archivos teniendo en cuenta las comillas que indican si es una columna de texto.

5.7.4 Exportación de grafos a formato graphML en TITAN DB

Para el caso del multígrafo, la exportación se realizó mediante la función especializada en lenguaje gremlin *GraphMLWriter* que permite guardar nodos y relaciones de un grafo en formato graphML.

```
g = TitanFactory.open('[Directorio titan DB]/conf/titan-cassandra-es.properties')
out = new FileOutputStream("multigraph_titan.graphml")
GraphMLWriter.outputGraph(g, out)
```

	Conjuntos de datos						
	Medidas	**1**	**2**	**3**	**4**	**5**	**Multigrafo**
Sin Filtros	**Nodos distintos**	793	1.244	2.907	881	833	3.653
	relaciones iniciales	17.684	8.510	8.042	698	685	24.016
Filtrados	**Nodos**	793	1.244	2.907	881	833	3.653
	Relaciones	6.091	8.500	8.042	698	685	24.016
	Nodos distintos con *self relations*	577	2	35	0	0	N/A
	Máxima repetición	74	2	1	1	1	N/A
	Relaciones consigo mismo	1.344	2	35	0	0	N/A
SNA	***betweenness* {min} {max} {prom}**	{0} {47.365,75} {695,47}	{0} {16.117,93} {82,79}	{0} {529.766,96} {4.770,13}	{0} {529.766,96} {4.770,13}	{0} {973} {14,80}	{0} {368.287,56} {4.933,61}
	***closeness* {min} {max} {prom}**	{0} {5,59} {2,93}	{0} {4,67} {0,39}	{0} {7,01} {4,29}	{0} {7,01} {4,29}	{1} {6,94} {2,43}	{0} {5,90} {3,70}
	***degree centrality* {min} {max} {prom}**	{1} {131} {15,36}	{1} {176} {13,66}	{1} {153} {5,53}	{1} {6} {1,58}	{1} {7} {1,64}	{1} {200} {13,11}
	Max clique	25	6	3	2	2	25
	Diameter	8	7	10	12	10	9
	Modularidad (*modularity*)	0,539	0,230	0,397	0,999	0,983	0,481
	Total de comunidades	84	15	30	184	149	15
	Comunidades con más de 2 nodos	14	15	25	128	113	7

Tabla 12: Resumen de datos generales y medidas SNA de grafos 1-5 y multigrafo

6. ANÁLISIS DE LOS RESULTADOS

Como se pudo observar en el análisis individual de cliques para cada grafo, la existencia por si sola de éstos no presenta un significado a nivel biológico que pueda ser determinante para encontrar un patrón de interés, adicionalmente su obtención en grafos altamente conectados puede conllevar a un elevado costo computacional y largos tiempos de cálculo. Sin embargo, al usar un multígrafo mixto con datos de diversos tipos, empieza a tener dicho análisis un sentido totalmente distinto, ya que la posibilidad de identificar patrones biológicos que mapeen a mecanismos regulatorios, conjuntos de genes y proteínas con funciones relacionadas o genes o proteínas que puedan anular o activar una función biológica, aumenta considerablemente.

De la misma forma, los análisis de SNA en grafos y multigrafos solo brindan detalles generales y parciales de posibles patrones existentes en los datos, por lo que se hace necesario hallar un mecanismo para facilitar la búsqueda de dichos patrones – en nuestro caso patrones de funciones regulatorias - de una forma rápida y efectiva. Es por esto que a continuación se plantean estrategias basadas en la búsqueda de quasi-cliques en multigrafos ayudados de restricciones de consultas para hacer más compacto el espacio de búsqueda.

6.1 QUASI-CLIQUES EN MULTIGRAFOS

Para el uso de cliques, se tomarán como referencia las definiciones formales de los mismos; sin embargo, con el fin de encontrar patrones biológicos de manera más flexible, se tomaron las definiciones de *quasi-cliques* que, aunque siguen casi completamente la definición formal de clique, tienen consideraciones que hace que tanto su estructura como su posible interpretación cambien, presentando una versión “relajada” del mismo.

Como ya se analizó en el apartado 1.2.3.3 y 1.2.3.4, existen diversos planteos sobre los “cliques débiles” o *weak clique*. Para el presente trabajo adoptaremos la definición tanto de *quasi-complete graph* como de *quasi-clique* [4] [22] [23] [24], a pesar de que la definición de *n-clique* [19] correspondiente a un *k-clique* determinado – para los efectos del presente trabajo – se ajustaría perfectamente a las necesidades a nivel experimental del mismo.

Así, diremos que un *0.5- quasi-clique* de un *k-clique* con $k = 5$ es un clique en el cual se esperan 10 relaciones entre los vértices y solo existen 5 de ellas. Es decir, un *γ quasi-clique* con $\gamma = \frac{5}{10} = 0{,}5$. Así, éste es conceptualmente equivalente a un *n-clique* con $n = 6$, es decir, un 6-clique con k=5, en donde un 1-clique es un subgrafo de tipo cliqué completo o *strong clique* (Ver **Ilustración 73**).

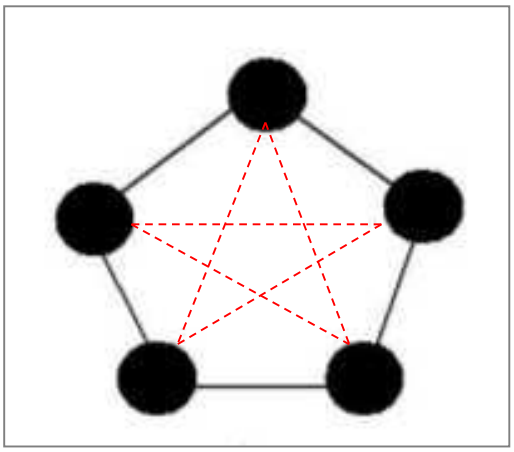

Ilustración 73: Ejemplo de un *0.5- quasi-clique* o un *6-clique* con k=5 en donde se marcan con líneas de puntos las 5 relaciones faltantes para completar un *strong clique* o clique completo.

Dado que en el presente trabajo se plantea el uso de un multígrafo mixto, se presentan tres factores que varían la adquisición y análisis de los quasi-cliques:

a) **Completitud estructural del clique, según sus relaciones**: es decir, correspondencia del subgrafo a un γ *quasi-clique.* Así, en algunos casos se presentarán *1- quasi-cliques* que corresponden a *strong cliques*, y en otros el valor de γ corresponderá a valores menores a 1 y mayores a cero (0).
b) **Tipo de relaciones que componen el *quasi-clique*:** dada la estructura de datos propuesta se pueden presentar dos casos:
    1. **Tipo de relaciones homogéneas al interior del quasi-clique**: éste es el caso más común, debido a que corresponde a quasi-clique de un grafo con conexiones del mismo tipo; dicha estructura es muy usada en la literatura y es descrita en [2]. [4] y [22] entre otros trabajos.
    2. **Tipo de relaciones heterogéneas al interior del quasi-clique:** éste será el caso de las relaciones presentadas en los resultados del presente trabajo en donde el número de tipo de relaciones posibles por cliqué son mayores a 1 y menores o iguales a 3, ya que solo existe en el planteamiento tres tipos de ellas. En una posible expansión del modelo éste número puede variar, y para cada caso se pueden plantear restricciones en los prototipos de consultas que limiten el número de relaciones esperadas en los quasi-cliques incluso para obtener el tipo 1.
c) **Tipo de relaciones:** para el caso del multígrafo mixto planteado, se tendrán relaciones dirigidas y no dirigidas, las cuales a nivel interpretativo podrán tomar un grado de importancia muy alto debido a que en algunos casos la dirección implica que una determinada PARTICLE o nodo es asumido como una proteína o gen en el *quasi-clique*, definición muy relevante a la hora de hablar de procesos regulatorios a nivel celular.

En los prototipos de consulta que serán presentados más adelante, la configuración de los resultados es la mostrada en la **Tabla 13**.

		1. Completitud estructural		2. Relaciones intra quasi-clique		3. Tipo de relaciones
Prototipo de consulta	k-clique	*γ quasi-clique*	n-clique equivalente	Relaciones iguales	Número de relaciones distintas	
1	3	1	1	No	[2,5]	Mixtas
2						
3						
4						
5						
6	4	0.83	2		5	
7	5	1	1		[2,5]	

Tabla 13: Factores variables de los 7 prototipos de consultas planteadas.

Se plantean adicionalmente, medidas internas de los quasi-cliques que permiten realizar filtros para hallar patrones biológicos; de ésta manera se establecen algunas definiciones previas, que son extensiones o generalizaciones de conceptos desarrollados para grafos convencionales:

- **Grado de centralidad conjunto de quasi-clique:** corresponde a la suma total de los grados de los nodos correspondiente al análisis SNA en el multígrafo que compone el quasi-clique.
- **Grado de intermediación conjunto de quasi-clique:** corresponde a la suma total de los grados de los nodos correspondiente al análisis SNA en el multígrafo que componen el quasi-clique.
- **Grado de cercanía conjunto de quasi-clique:** corresponde a la suma total de los grados de los nodos correspondiente al análisis SNA en el multígrafo que componen el quasi-clique.
- **Número de repetición conjunta de quasi-clique:** corresponde a la suma total de las relaciones repetidas entre un nodo A y un nodo B. En el multigrafo se expresa mediante el atributo COUNTER_REPITED o el atributo STD_COUNTER_REPITED; éste último corresponde a la estandarización del número de repeticiones de relaciones tomando el número máximo de repeticiones como valor base para la estandarización.

Los anteriores conceptos, posibilitan la creación y modificación de métricas conjuntas que en algún punto puedan ser calculadas a partir de atributos de nodos o de relaciones, y que adquieran con el análisis significados biológicos representativos.

El hecho de plantear métricas conjuntas a nivel quasi-cliques, abre la puerta a análisis de *clustering* y de comunidades en multígrafos no solo de tipo biológico, sino también de cualquier otro tipo.

6.2 PATRONES BIOLÓGICOS EN QUASI-CLIQUES DE MULTIGRAFOS

Para verificar la flexibilidad y aplicabilidad de la arquitectura propuesta se plantean los siguientes siete prototipos de consultas que se asocian con preguntas biológicas específicas:

1. Relaciones consecutivas específicas.
2. Relaciones consecutivas y maximización de métricas conjuntas.
3. Relaciones consecutivas y proteínas o genes en identificadores.

4. Relaciones consecutivas y proteínas o genes en nombres y anotaciones.
5. Número de relaciones de tipo distinto.
6. Relaciones de conjuntos de datos distintos.
7. Relaciones distintas consecutivas.

Cada uno de las preguntas y prototipos se traduce en una consulta tanto en Cypher como en Gremlin, las cuales se ejecutan en Neo4J y Titan DB respectivamente. Los resultados gráficos que se muestran se generaron con la herramienta de visualización Web de NEO4J.

Se experimentó con algunos otros métodos de importación de datos desde neo4j usando gephi, pero el driver probado no funcionó apropiadamente en las versiones planteadas para los experimentos, al parecer, porque su desarrollo no está tan actualizado como las versiones de dicho motor de base de datos.

En general – y a modo de ejemplo – se tomaron pocos quasi-cliques para ser graficados o analizados. Esto requirió filtrar algunos quasi-cliques interesantes o con propiedades deseadas entre un gran número de ellos. Las estrategias de filtrado usadas fueron:

- Filtros en tipos de relaciones entre nodos. Esto es, determinar alguna característica deseada en las relaciones, por ejemplo, que al menos una arista se corresponda con una relación de co-expresión génica.
- Condicionamiento de relaciones con un específico orden y de determinado tipo en la conformación del quasi-clique.
- Condiciones de número de relaciones obligatorias en cada quasi-clique. Por ejemplo, que el cuasi-clique contenga al menos una relación de interacción proteína-proteína y dos de co-expresión.
- Quasi-cliques que contienen cierta proteína o gen en su estructura.
- Quasi-cliques que tengan nodos con determinada característica topológica en el multígrafo. Por ejemplo, máximo grado, o intermediación o que cumplan con un máximo o mínimo de una medida conjunta a nivel quasi-clique como podría ser un grado de centralidad conjunto o grado de intermediación conjunto.

Para algunos casos, la explicación biológica puede determinarse al inicio hipotéticamente. Sin embargo, existen otras que siguen un patrón más típico de los procesos de minería de datos, ya que solo hasta encontrar los resultados y analizar los atributos de cada nodo y relación del mismo, se puede intentar entender su sentido biológico.

Por otro lado, en caso de tener muchos quasi-cliques de interés, se encontró que el análisis gráfico uno a uno, maximiza el posible entendimiento a nivel biológico y estructural del mismo.

6.2.1 Prototipo de consulta 1 – Relaciones consecutivas específicas.

Pregunta biológica: ¿Cuantos grupos de tres proteínas o genes tienen una interacción proteína-proteína, luego una interacción de co-expresión y finalmente tiene una interacción de regulación o de co-expresión?

Resumen: 1-quasi-cliques k=3, con variaciones PPIs hacia co-expresión y co-expresión o Zona regulatoria.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 1-quasi-clique con k=3.
2. La primera relación corresponde a datos de interacción proteína-proteína.
3. La segunda relación corresponde a datos de co-expresión.
4. La tercera relación corresponde a una co-expresión o a una región reguladora.
5. Los tres nodos del quasi-clique deben ser distintos.
6. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.

En NEO4J mediante el lenguaje de consultas CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTI
ON_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND b<>c AND c<>a
RETURN a,b,c,r1,r2,r3
```

En Titan DB mediante el lenguaje de consultas Gremlin la consulta generada es la siguiente:

```
g g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').b
othV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_C
O_EXPRESSION_2').bothV().sideEffect{C = it}.filter{B!=C &&
A!=C}.bothE('INTERACTION_REGULATION_1','INTERACTION_CO_EXPRESS
ION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffect{A_ =
it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}-
>${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}-
>${it.get(4).property('ID_')}->${it.get(5).label()}"}
```

Se obtienen 47 quasi-cliques compuestos a su vez por 86 nodos y 472 relaciones. De estas relaciones se tiene la siguiente composición:

Para visualizar mejor los resultados, se tomarán los primeros cuatro quasi-cliques como se puede ver en la ***Ilustración 74***.

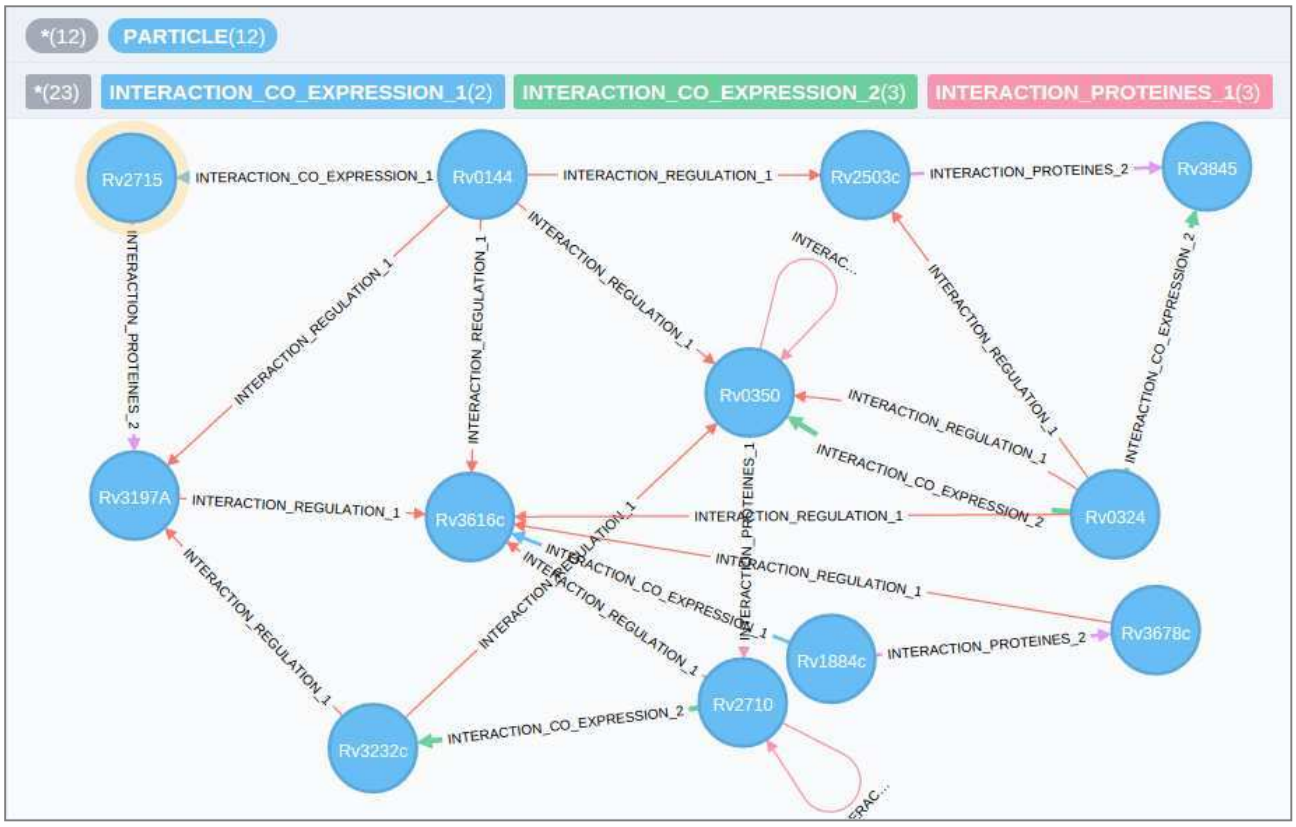

Ilustración 74: Cuatro primeros quasi-cliques con k=3 correspondientes al prototipo de consulta 1.

Como se puede observar, se conforman estructuras de interacción adicionales entre proteínas y/o genes. En la parte superior de la **Ilustración 74**, se ven los colores que identifican cada tipo de relación en el grafo. Un factor a tener en cuenta es que las consultas de este tipo pueden encontrar un gran número de quasi-cliques. Esto podría condicionar las estrategias de búsqueda. Por ejemplo, aplicando esquemas de búsqueda masiva automática, o especificando aún más la consulta. Como se mencionó en el apartado introductorio de éste apartado, analizar muchos cliques puede conllevar a confusiones, incluso, teniendo en cuenta solo 4 quasi-cliques, que por ser k=3, se tienen $3 \times 4 = 12$ nodos sin repetición e interrelaciones entre ellos.

Otro factor importante a tener en cuenta, es la difícil distinción entre los quasi-cliques buscados y los cliques con restricciones distintas que aparecen por relaciones complementarias, es decir, relaciones existentes entre nodos que componen el resultado de la consulta pero que no fueron especificadas en los filtros de la misma; por ejemplo, si se tiene como resultados dos quasi-cliques A y B cada uno con k=3, se esperarían en total 6 relaciones, es decir, dos por clique, sin embargo si existen relaciones entre los nodos de A y de B, éstas son llamadas relaciones complementarias; así, cuando se tienen resultados de quasi-cliques con muchas relaciones complementarias, en ocasiones se forman seudo-cliques adicionales no buscados, por lo que para resultados numerosos se hace difícil su visualización. Es por estas razones que en los siguientes prototipos de consultas se realizan algunos filtros que ayudan a generar una simplificación sobre los resultados y por ende un mejor entendimiento de los mismos.

6.2.2 Prototipo de consulta 2: Relaciones consecutivas y maximización de métricas conjuntas

Pregunta biológica: ¿Cuáles son los grupos de tres proteínas o genes con la máxima métrica conjunta (grado de centralidad conjunta, grado de intermediación conjunta, grado de cercanía conjunto o número de repetición conjunta) que tienen una interacción proteína-proteína, luego una relación de co-expresión y finalmente una interacción de regulación proteína- zona regulatoria de un gen?

Resumen: se buscan 1-quasi-cliques de k=3, con tres tipos de relaciones biológicas, tomando los dos quasi-cliques con mayor o menor:

- Grado de centralidad conjunto.
- Grado de intermediación conjunto.
- Grado de cercanía conjunto.
- Número de repetición conjunta.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones, las cuales en parte son las planteadas en el numeral 4.2.2:

1. Es un 1-quasi-clique con k=3.
2. La primera relación corresponde a datos de interacción proteína-proteína.
3. La segunda relación corresponde a datos de co-expresión.
4. La tercera relación corresponde a una co-expresión o a una región reguladora.
5. Se calcula por cada quasi-clique el grado conjunto, mediante la suma del grado de cada nodo o relación –según aplique - en el multígrafo.
6. Se ordenan de mayor a menor o viceversa los quasi-cliques teniendo en cuenta la medida específica de interés y se toman los 2 primeros. Las medidas a usar son:
   - Grado de centralidad conjunto (DEGREE_CENTRALITY).
   - Grado de intermediación conjunto (BETWEENNESS_CENTRALITY).
   - Grado de cercanía conjunto (CLOSENESS_CENTRALITY).
   - Número de repetición conjunta (COUNTER_REPITED o STD_COUNTER_REPITED).
7. Los tres nodos del quasi-clique deben ser distintos.
8. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.

A continuación, se presentan los resultados de las consultas para cada caso de medida; solo para la medida de grado de centralidad conjunto se explicitan las consultas en NEO4J y Gremlin, ya que en los otros tres casos solo cambian el nombre del parámetro usado en la consulta y el uso de las cláusulas DESC o ASC si se quiere el mayor o menor grado conjunto respectivamente.

1. **Grado de centralidad conjunto**: Se calcula el grado de centralidad conjunta y se toman los 1-quasi-cliques con los mayores dos grados conjuntos.

En NEO4J mediante el lenguaje de consultas CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTI
ON_CO_EXPRESSION_2]-(a)
WITH a,b,c,r1,r2,r3,(t toFloat(a.DEGREE_CENTRALITY)+
toFloat(b.DEGREE_CENTRALITY)+ toFloat(c.DEGREE_CENTRALITY)) as
total_degree
WHERE a<>b AND b<>c AND c<>a
RETURN a,b,c,r1,r2,r3,total_degree
ORDER BY total_degree DESC
LIMIT 2
RETURN a,b,c,r1,r2,r3
```

En Titan DB utilizando el lenguaje de consultas Gremlin la consulta es la siguiente:

```
g.V().sideEffect{A = it}.bothE('INTERACTION_PROTEINES_1',
'INTERACTION_PROTEINES_2').bothV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1',
'INTERACTION_CO_EXPRESSION_2').bothV().sideEffect{C = it}.filter{B!=C &&
A!=C}.bothE('INTERACTION_REGULATION_1','INTERACTION_CO_EXPRESSION_1','I
NTERACTION_CO_EXPRESSION_2').bothV().sideEffect{A_ =
it}.filter{A==A_}.path().sort{it.get(0).property('DEGREE_CENTRALITY').value()+it.g
et(2).property('DEGREE_CENTRALITY').value()+it.get(4).property('DEGREE_CENTR
ALITY').value()}.reverse().collect{"${it.get(0).property('ID_')}->${it.get(1).label()}-
>${it.get(2).property('ID_')}->${it.get(3).label()}->${it.get(4).property('ID_')}-
>${it.get(5).label()}-
>${it.get(0).property('DEGREE_CENTRALITY').value()+it.get(2).property('DEGREE_
CENTRALITY').value()+it.get(4).property('DEGREE_CENTRALITY').value()}"}[0..1]
```

En la **Ilustración *75***, se pueden ver 6 nodos y 16 relaciones, de las cuales 10 son relaciones complementarias entre nodos y 6 corresponden a los dos quasi-cliques con máximo grado, siendo el del recuadro de la izquierda el quasi-clique con mayor grado de centralidad conjunto igual a 477 y el de la derecha el que le sigue con 428 de grado conjunto, teniendo éste último, a su vez el nodo con mayor grado de todos correspondiente a 200 en la proteína Rv0350, cuyas características o atributos se muestran en la parte inferior.

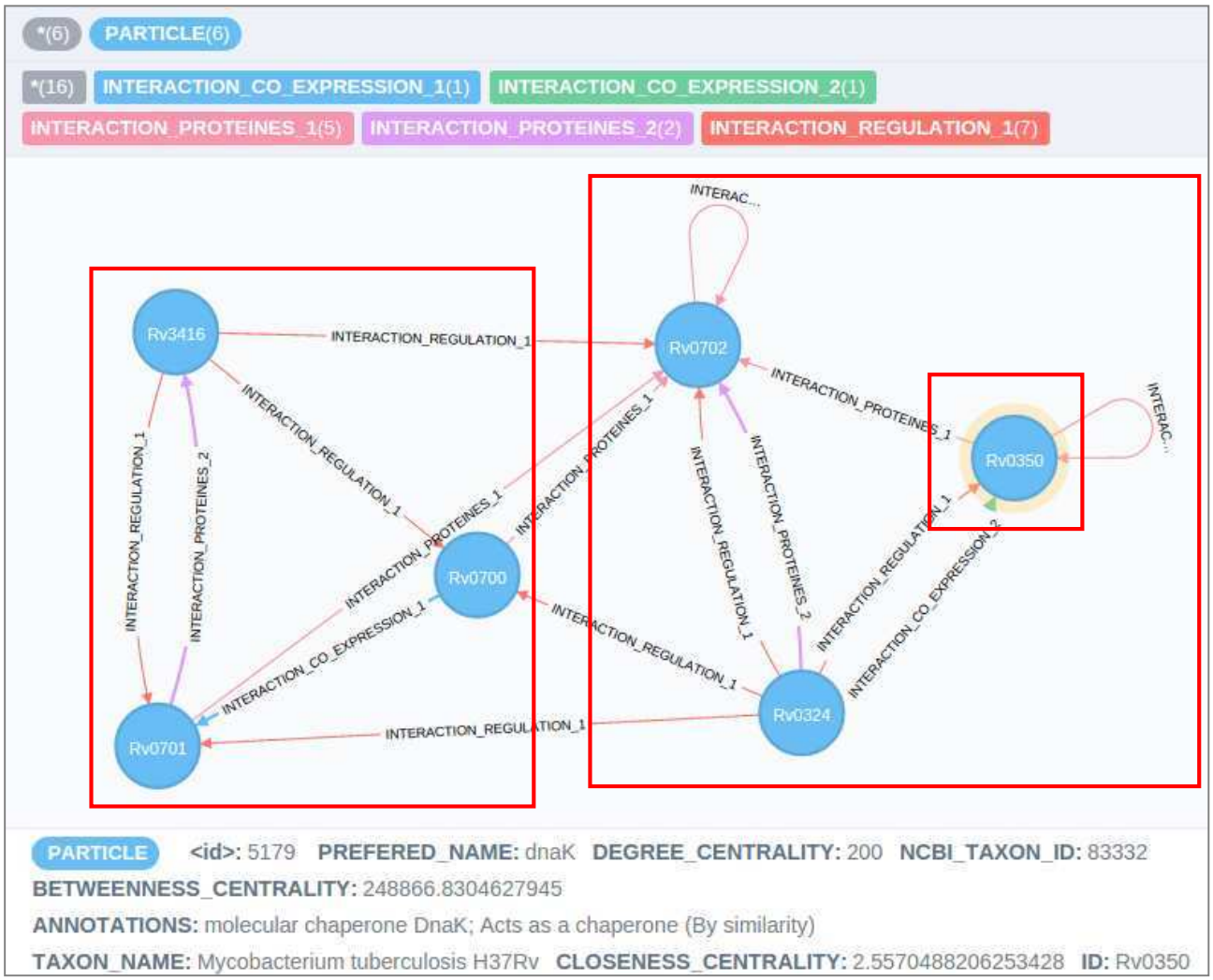

Ilustración 75: Prototipo de consulta 2 con 2 quasi-cliques basados en el grado de centralidad conjunto mayor.

Adicionalmente podemos observar en la **Ilustración 76**, como se presentan en los grafos de manera explícita quasi-cliques con restricciones no buscadas mediante relaciones complementarias, pero que pueden conllevar a un análisis posterior, por ejemplo quasi-cliques del tipo: (zona regulatoria) – (zona regulatoria) – (Interacción proteína-proteína) (ver recuadro marrón superior izquierdo), o quasi-cliques del tipo (zona regulatoria) – (zona regulatoria) – (co-expresión) (Ver recuadro verde inferior izquierdo). También es importante destacar aquí que la visualización de estas consultas no solo facilita entender las relaciones entre nodos, también es posible en la misma salida ver información más detallada de cada nodo. Ver en la parte inferior de la **Ilustración *75*** la información sobre el vértice seleccionado que corresponde a Rv0350. De la misma manera, se podría visualizar la información registrada para una arista del quasi-clique.

El gen Rv0350, o dnaK, codifica para la proteína DnaK, que es una chaperona involucrada en procesos de respuesta al estrés, en los que es común que participen diferentes proteínas y que requieren una coordinación de sus actividades. La proteína DnaK se encontró en fracciones extraídas del citosol, la membrana y la pared celular. Rv0324 codifica un regulador transcripcional de la familia arsR que se ubicaría en la membrana celular que aún no ha sido estudiado en detalle. Rv0702, o rplD, es un gen esencial que codifica una proteína ribosomal, que también se encontró en fracciones de membrana. En las bases de datos especializadas (String-DB, TBDB, MTM Network

Portal) estas interacciones no están registradas, aunque en la **Ilustración *75*** se muestra que el quasi-clique está soportado por más de una evidencia experimental.

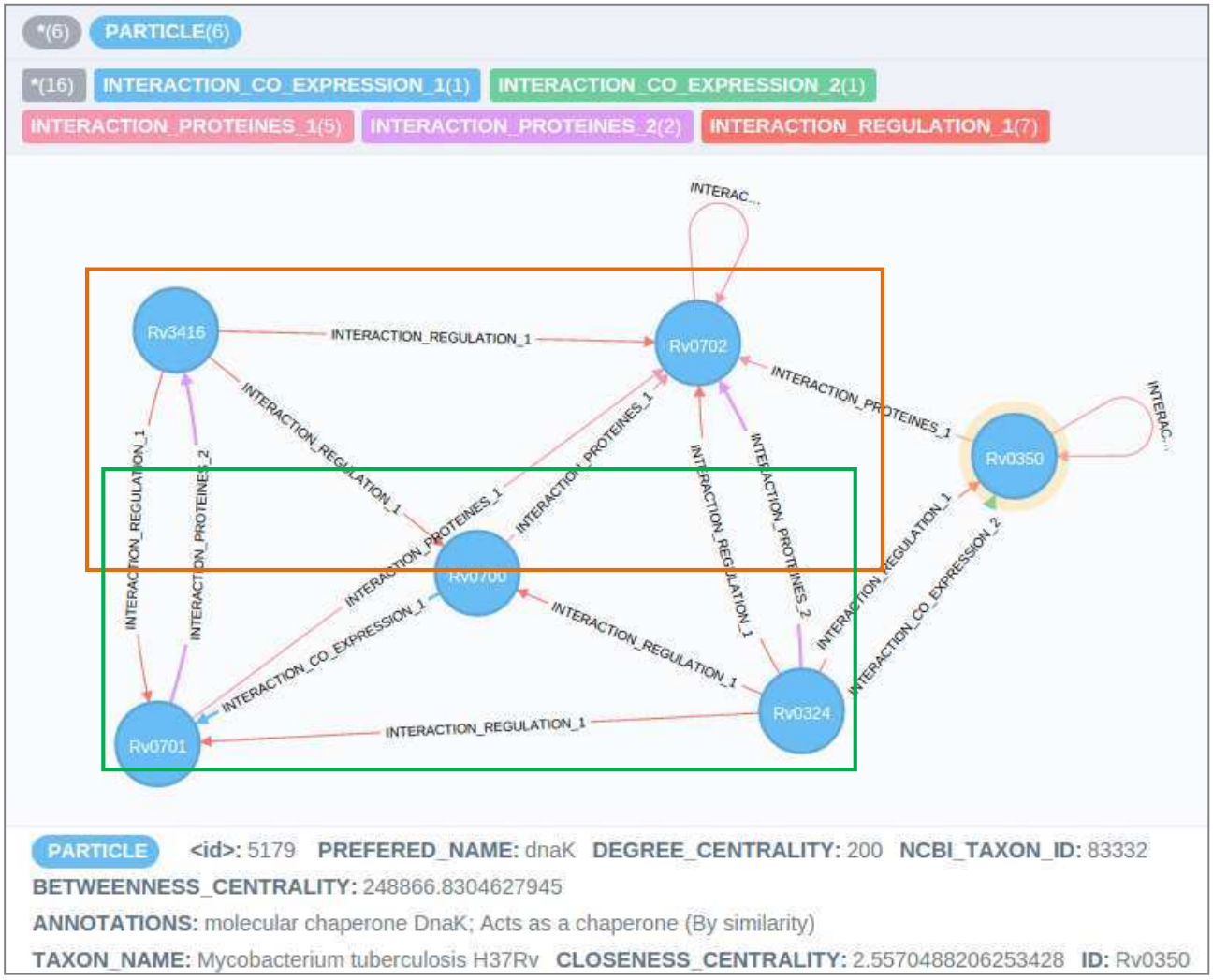

Ilustración 76: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de centralidad conjunta mayor con quasi-cliques no buscados.

2. **Grado de intermediación conjunto**: Se calcula el grado de intermediación conjunta y se toman los 1-quasi-cliques con los mayores dos grados conjuntos.

Dado que se comparten los filtros del prototipo de consultas 1, se tienen como base los mismos resultados, es decir 47 quasi-cliques compuestos por 86 nodos y 472 relaciones. En la **Ilustración 77**, para los dos quasi-cliques con mayor intermediación se pueden ver 6 nodos y 16 relaciones, de las cuales 10 son relaciones complementarias entre nodos y 6 corresponden a los dos quasi-cliques con mínimo grado, siendo el del triángulo verde de la derecha el quasi-clique con mayor grado conjunto igual a 533.273 y el de la izquierda el que le sigue con 383.153 de grado conjunto. Se muestran las características en la parte inferior del nodo Rv0702.

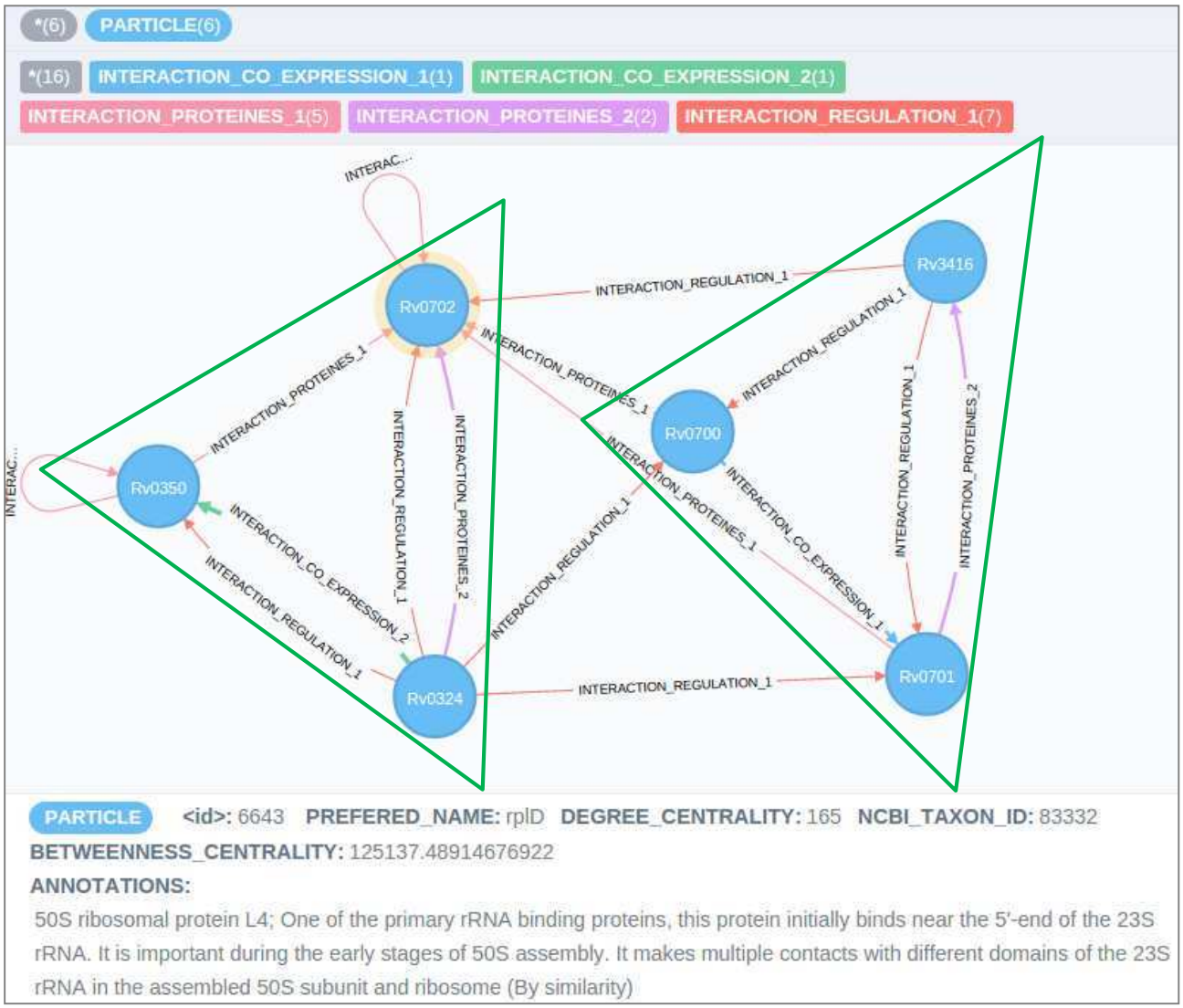

Ilustración 77: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de intermediación mayor.

3. **Grado de cercanía conjunto**: Se calcula el grado de cercanía conjunta y se toman los 1-quasi-cliques con los mayores dos grados conjuntos.

En la **Ilustración 78**, para los dos quasi-cliques con menor cercanía se pueden ver 6 nodos y 8 relaciones, de las cuales 2 son relaciones complementarias entre nodos que justamente son relaciones proteína-proteína de la proteína consigo misma y 6 corresponden a los dos quasi-cliques con mayor grado conjunto de cercanía. En éste caso, ambos quasi-cliques están separados y son fácilmente identificables visualmente, teniendo un valor de grado conjunto de cercanía igual a 10,05 y 9,99 de derecha a izquierda respectivamente. Se muestran las características en la parte inferior de la relación consigo mismo del nodo Rv0668.

El gen Rv0668 o rpoC, codifica para la proteína RpoC. Se ha determinado que este es un gen esencial, dado que no existen mutantes viables sin este gen. RpoC es una de las variantes de la subunidad beta de la RNA polimerasa una enzima fundamental para los procesos de transcripción génica. Esto explica su alto grado de intermediación en el grafo. Algunas mutaciones en este gen son responsables de la aparición de resistencia a ciertos antibióticos. El producto del gen Rv2839c (infB) es el factor IF-2 de la iniciación de la

traducción, Finalmente, el producto de Rv2199c es el polipéptido 4 de la citocromo coxidasa. Una enzima involucrada en procesos de generación de energía.

Los tres procesos, transcripción, traducción y formación de energía están acoplados y precisan estar regulados entre sí. La representación del quasi-clique en la **Ilustración *78*** sugiere que, aunque los registros indican evidencia de co-expresión entre dos de los tres genes, es probable que exista entre los tres.

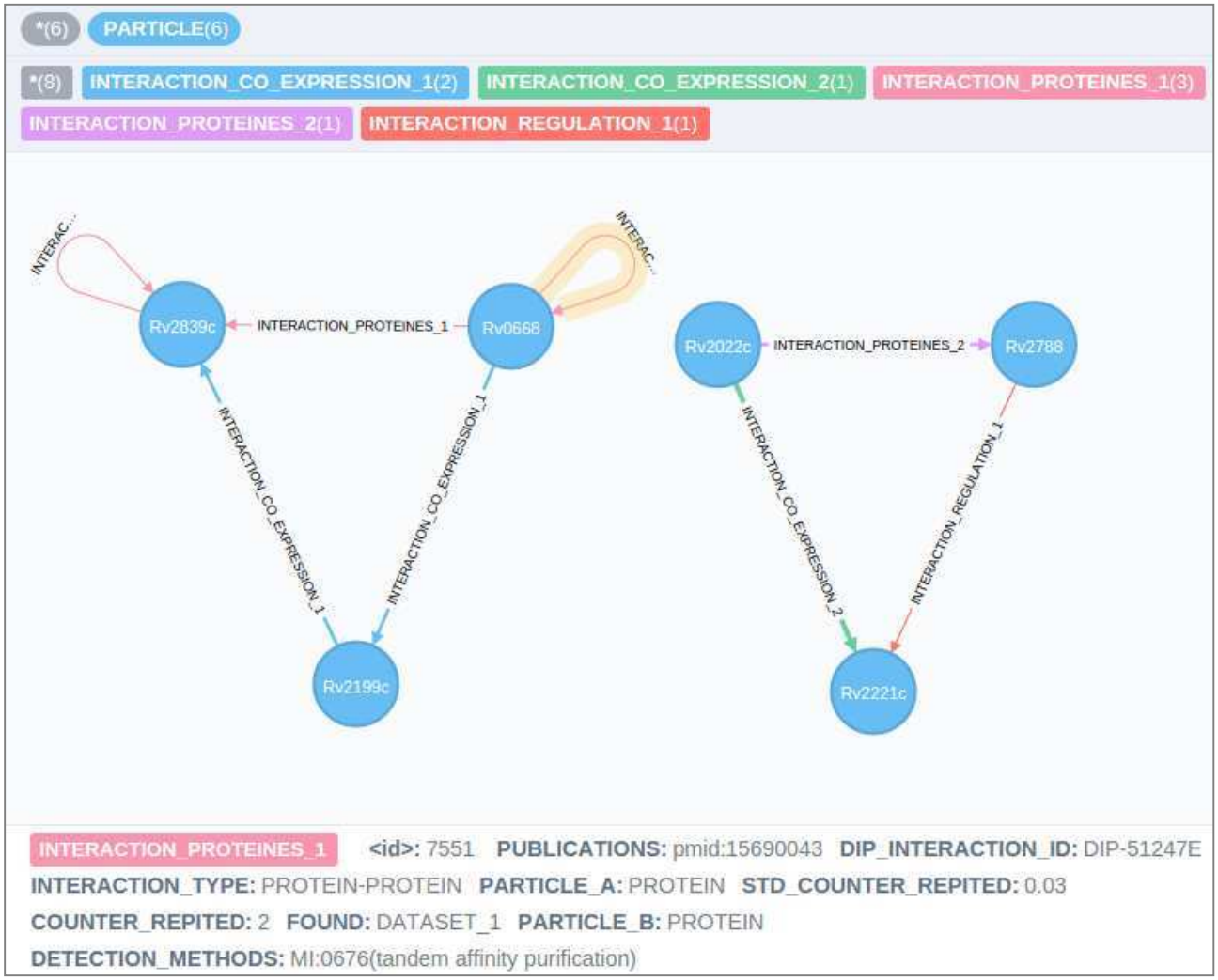

Ilustración 78: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de cercanía mayor.

4. **Número de repetición conjunta:** Se calcula el grado de centralidad conjunta y se toman los quasi-cliques con los mayores dos grados conjuntos.

En la **Ilustración 79**, para los dos quasi-cliques con mayor repetición se pueden ver tres nodos y seis relaciones, de las cuales tres son relaciones complementarias entre nodos. En éste caso, ambos quasi-cliques están sobre los mismos tres nodos, solo que uno inicia en Rv0665 y otro en Rv0685. Como se puede ver en la parte inferior, la relación Rv0665 y Rv0685 tiene una repetición igual a 14 y los cliques en conjunto tienen una repetición conjunta de 16.

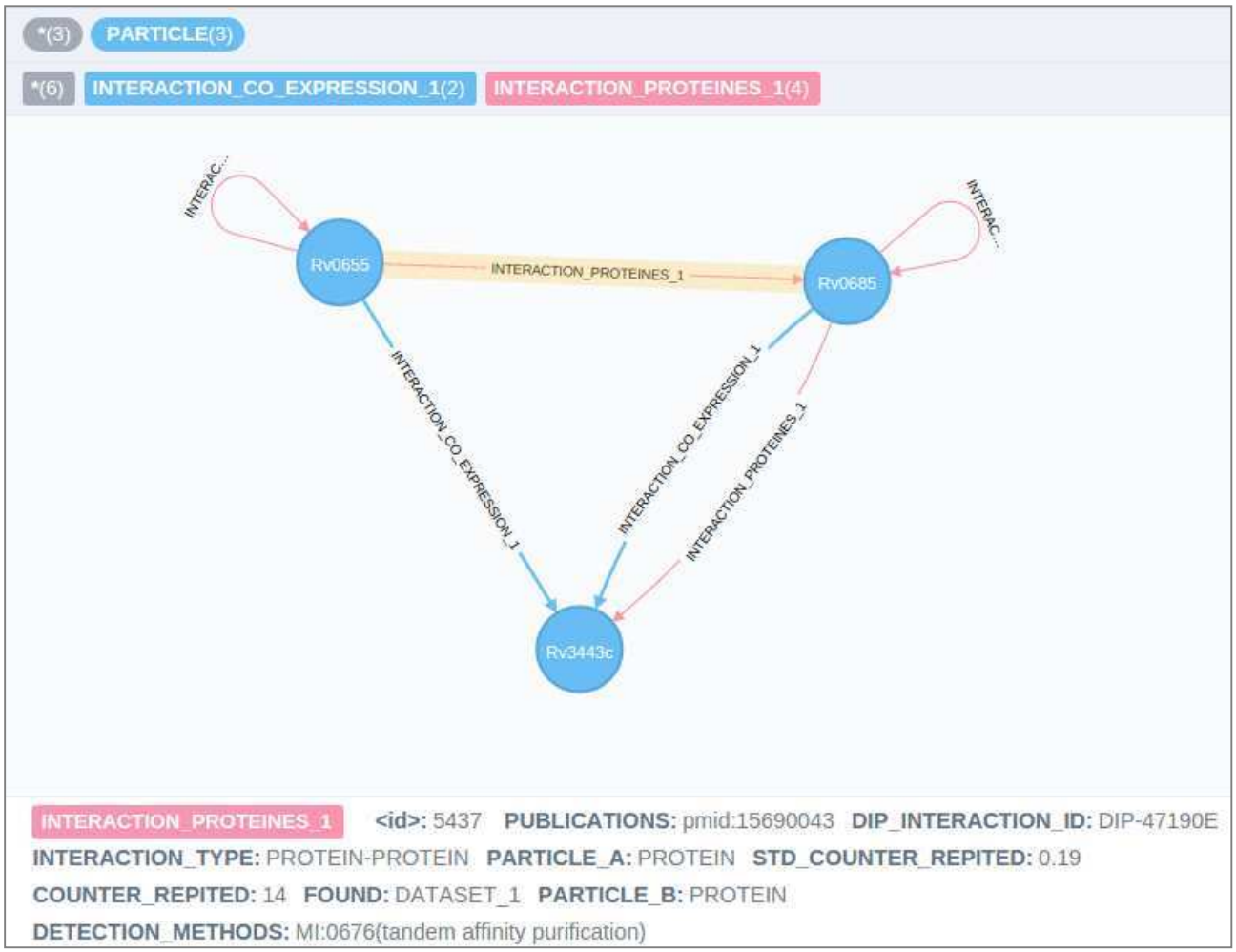

Ilustración 79: Prototipo de consulta 2 con 2 quasi-cliques basados en la medida de repetición conjunta mayor.

6.2.3 Prototipo de consulta 3 - Relaciones consecutivas y proteínas o genes en identificadores.

Pregunta biológica: ¿Cuál es el grupo de tres proteínas o genes que contiene alguna proteína o gen correspondiente a las cinco con mayor métrica SNA (grado de centralidad, grado de intermediación, grado de cercanía, número de repetición) y que se encuentran conectadas entre sí consecutivamente por alguna de las siguientes dos secuencias: una interacción proteína-proteína, una interacción de co-expresión y una interacción de regulación o una interacción proteína-proteína, una interacción de co-expresión y una interacción de co-expresión?

Resumen: 1-quasi-cliques k=3, con relaciones de interacción proteína-proteína, co-expresión y regulación proteína- zona regulatoria de un gen, tomando los quasi-cliques que contienen las cinco proteínas-genes con mayor grado en el multigrafo.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 1-quasi-clique con k=3.
2. La primera relación corresponde a datos de interacción proteína-proteína.
3. La segunda relación corresponde a datos de co-expresión.

4. La tercera relación corresponde a una co-expresión o a una región reguladora.
5. Se filtran los quasi-cliques que contienen nodos correspondientes a las 5 proteínas y genes con mayor grado de centralidad en el multígrafo.
6. Los tres nodos del quasi-clique deben ser distintos.
7. No se permite que en un quasi-clique se presente una relación de un nodo consigo mismo (con ésta limitación se pasa de 12 a 11 quasi-cliques).

Al igual que en el caso del prototipo de consulta 2, se puede realizar una variación de la consulta tomando los mayores o menores valores de cada medida y luego realizar la agregación de los ID de dichas proteínas-genes en la cláusula WHERE. Las medidas posibles podrían ser:

- Grado de centralidad (DEGREE_CENTRALITY).
- Grado de intermediación (BETWEENNESS_CENTRALITY).
- Grado de cercanía (CLOSENESS_CENTRALITY).
- Número de repetición (COUNTER_REPITED o STD_COUNTER_REPITED).

Para éste caso, solo se usará el grado de centralidad a modo de ejemplo.

En NEO4J/CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTION
_CO_EXPRESSION_2]-(a)
WHERE
 a<>b AND b<>c AND c<>a AND
        (a.ID in["Rv0350","Rv3416","Rv1219c","Rv1990c","Rv0702"] OR
        b.ID in["Rv0350","Rv3416","Rv1219c","Rv1990c","Rv0702"] OR
        c.ID in["Rv0350","Rv3416","Rv1219c","Rv1990c","Rv0702"])
RETURN a,b,c,r1,r2,r3
Limit 3
```

En Titan DB/Gremlin la consulta es:

```
g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').bothV().side
Effect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRES
SION_2').bothV().sideEffect{C = it}.filter{B!=C && A!=C &&
((A.attach().property('ID_').value().equals("Rv0350") ||
A.attach().property('ID_').value().equals("Rv3416") ||
A.attach().property('ID_').value().equals("Rv1219c") ||
A.attach().property('ID_').value().equals("Rv1990c") ||
A.attach().property('ID_').value().equals("Rv0702")) ||
(B.attach().property('ID_').value().equals("Rv0350") ||
B.attach().property('ID_').value().equals("Rv3416") ||
B.attach().property('ID_').value().equals("Rv1219c") ||
B.attach().property('ID_').value().equals("Rv1990c") ||
B.attach().property('ID_').value().equals("Rv0702")) ||
(C.attach().property('ID_').value().equals("Rv0350") ||
C.attach().property('ID_').value().equals("Rv3416") ||
C.attach().property('ID_').value().equals("Rv1219c") ||
C.attach().property('ID_').value().equals("Rv1990c") ||
C.attach().property('ID_').value().equals("Rv0702")))}.bothE('INTERACTION_REGUL
ATION_1','INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bo
thV().sideEffect{A_ = it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}-
>${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}-
>${it.get(4).property('ID_')}->${it.get(5).label()}"}0..2]
```

En total se encontraron 11 quasi-cliques que cumplen con la consulta sin ninguna limitación. Para permitir un mejor entendimiento de los resultados se toman los 3 primeros. En la **Ilustración 80**, se pueden ver siete nodos y 14 relaciones, de las cuales cinco son relaciones complementarias entre nodos y nueve corresponden a los tres quasi-cliques limitados en la consulta presentada, los cuales están encerrados en triángulos rojos en dicha ilustración. Vemos que con respecto al prototipo de consulta 2 se comparten dos de las proteínas o genes: Rv0350 y Rv0702, lo cual es esperable ya que éstas tienen muy alto grado en el multígrafo. Sin embargo, vemos que los patrones encontrados son distintos y proporcionan información diferente.

De igual forma podemos ver que se genera un quasi-clique no buscado del tipo (proteína-proteína) - (proteína-proteína) - (proteína-proteína) entre los nodos: Rv0702-Rv0260c-Rv2710. La información complementaría en la parte inferior de la ilustración muestra la información de la relación entre el nodo Rv0350 y Rv0702. Como se mencionó antes, esta funcionalidad es altamente útil cuando se quiere analizar rápidamente las propiedades de cualquier nodo o relación en cliques ya formados y hace parte de las funcionalidades que vienen incorporadas en el visualizador de grafos estándar de la consola web de NEO4J.

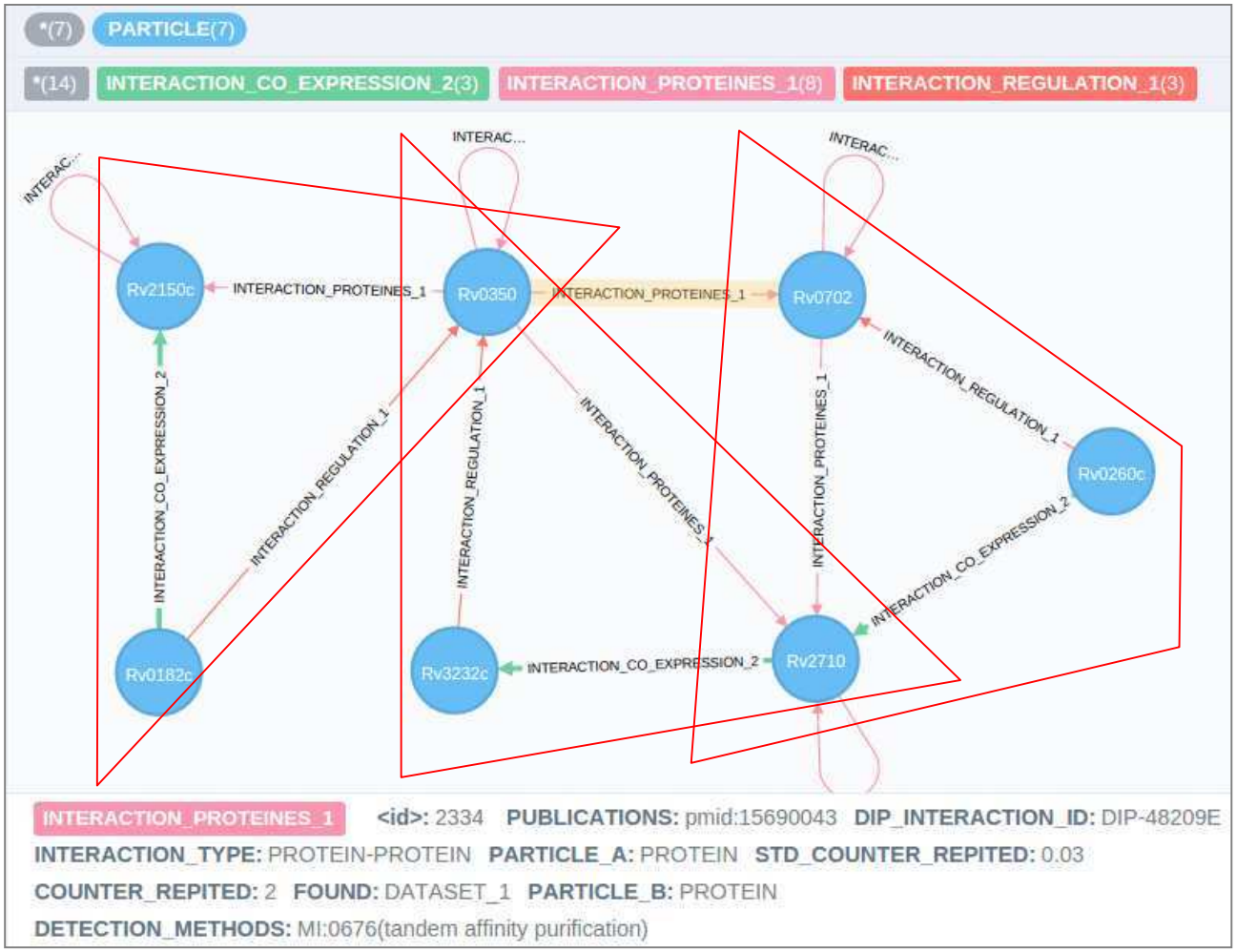

Ilustración 80: Prototipo de consulta 3 que muestra 3 de los 11 quasi-cliques encontrados filtrando el top 5 de las proteínas con más grado de multigrafo.

Por otro lado, los resultados además de poder ser visualizados, pueden ser vistos mediante datos, en general presentando la información de cada nodo y posteriormente las relaciones asociadas del quasi-clique. En la **Tabla 14** podemos ver los datos

correspondientes los 3 quasi-cliques con los datos de cada nodo y posteriormente de cada relación.

	quasi-clique 1	quasi-clique 2	quasi-clique 3
Nodo a	{"ANNOTATIONS":"molecular chaperone DnaK; Acts as a chaperone (By similarity)","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"200","ID":"Rv0350","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.55704882062 53428","BETWEENNESS_CENTRALITY":"248866.8304 627945","PREFERED_NAME":"dnaK"}	{"ANNOTATIONS":"50S ribosomal protein L4; One of the primary rRNA binding proteins, this protein initially binds near the 5'-end of the 23S rRNA. It is important during the early stages of 50S assembly. It makes multiple contacts with different domains of the 23S rRNA in the assembled 50S subunit and ribosome (By similarity)","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"165","ID":"Rv0702","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.592430060340 099","BETWEENNESS_CENTRALITY":"125137.489146 76922","PREFERED_NAME":"rplD"}	{"ANNOTATIONS":"molecular chaperone DnaK; Acts as a chaperone (By similarity)","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"200","ID":"Rv0350","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.5570488206253428","BETWEENNESS_CENTRALITY":"248866.8304627945","PREFERED_NAME":"dnaK"}
Nodo b	{"ANNOTATIONS":"RNA polymerase sigma factor SigB; Sigma factors are initiation factors that promote the attachment of RNA polymerase to specific initiation sites and are then released (By similarity)","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"110","ID":"Rv2710","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.74986286341 1958","BETWEENNESS_CENTRALITY":"114592.29828 027566","PREFERED_NAME":"mysB"}	{"ANNOTATIONS":"RNA polymerase sigma factor SigB; Sigma factors are initiation factors that promote the attachment of RNA polymerase to specific initiation sites and are then released (By similarity)","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"110","ID":"Rv2710","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.749862863411 958","BETWEENNESS_CENTRALITY":"114592.298280 27566","PREFERED_NAME":"mysB"}	{"ANNOTATIONS":"Cell division protein FtsZ; Essential cell division protein that forms a contractile ring structure (Z ring) at the future cell division site. The regulation of the ring assembly controls the timing and the location of cell division. One of the functions of the FtsZ ring is to recruit other cell division proteins to the septum to produce a new cell wall between the dividing cells (By similarity). Binds GTP and shows GTPase activity","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"50","ID":"Rv2150c","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"3.063631376851344","BETWEENNESS_CENTRALITY":"23327.02593509441","PREFERED_NAME":"ftsZ"}
Nodo c	{"ANNOTATIONS":"transcriptional regulatory protein PvdS","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"53","ID":"Rv3232c","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.91771804717 49864","BETWEENNESS_CENTRALITY":"14016.87714 9870683","PREFERED_NAME":"ppk2"}	{"ANNOTATIONS":"Possible transcriptional regulatory protein","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"50","ID":"Rv0260c","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","BETWEENNESS_CENTRALITY":"9875.348857 726289","CLOSENESS_CENTRALITY":"2.97888096544 15798","PREFERED_NAME":"MT0273"}	{"ANNOTATIONS":"RNA polymerase factor sigma-70","NCBI_TAXON_ID":"83332","DEGREE_CENTRALITY":"70","ID":"Rv0182c","TAXON_NAME":"Mycobacterium tuberculosis H37Rv","CLOSENESS_CENTRALITY":"2.9259462424574876","BETWEENNESS_CENTRALITY":"30644.332536540078","PREFERED_NAME":"sigG"}
Relación 1	{"DIP_INTERACTION_ID":"DIP-45065E","DETECTION_METHODS":"MI:0401(biochemical)","PARTICLE_A":"PROTEIN","STD_COUNTER_REPITED":"0.03","PARTICLE_B":"PROTEIN","COUNTER_REPITED":"2","PUBLICATIONS":"pmid:12059959","INTERACTION_TYPE":"PROTEIN-PROTEIN","FOUND":"DATASET_1"}	{"DIP_INTERACTION_ID":"DIP-51282E","DETECTION_METHODS":"MI:0676(tandem affinity purification)","PARTICLE_A":"PROTEIN","STD_COUNTER_REPITED":"0.03","PARTICLE_B":"PROTEIN","COUNTER_REPITED":"2","PUBLICATIONS":"pmid:15690043","INTERACTION_TYPE":"PROTEIN-PROTEIN","FOUND":"DATASET_1"}	{"DIP_INTERACTION_ID":"DIP-48572E","DETECTION_METHODS":"MI:0676(tandem affinity purification)","PARTICLE_A":"PROTEIN","STD_COUNTER_REPITED":"0.03","PARTICLE_B":"PROTEIN","COUNTER_REPITED":"2","PUBLICATIONS":"pmid:15690043","INTERACTION_TYPE":"PROTEIN-PROTEIN","FOUND":"DATASET_1"}
Relación 2	{"PARTICLE_A":"GEN","STD_COUNTER_REPITED":"1.00","PARTICLE_B":"GEN","COUNTER_REPITED":"1","INTERACTION_TYPE":"GEN-GEN_COEXPRESSION","FOUND":"DATASET_5"}	{"PARTICLE_A":"GEN","STD_COUNTER_REPITED":"1.00","PARTICLE_B":"GEN","COUNTER_REPITED":"1","INTERACTION_TYPE":"GEN-GEN_COEXPRESSION","FOUND":"DATASET_5"}	{"PARTICLE_A":"GEN","STD_COUNTER_REPITED":"1.00","PARTICLE_B":"GEN","COUNTER_REPITED":"1","INTERACTION_TYPE":"GEN-GEN_COEXPRESSION","FOUND":"DATASET_5"}
Relación 3	{"PROMOTER_SEQUENCE":"CTCCTGGCTCGATGAAAGAATCAGCGGCCGCACTCGGCGAGCGCTGCTCAGCGACCGTCAGCACGCCAGTTCAATACGGCGAGAAGCGCGTTCTCATCGAAAGTCGGTCAAGCTGGCGGACTGAGCACTGCTCTCCGGGCTTGACCGGGGCCTCTCCCAGCTACGCCCCGAGCGTGTGCCCTGCCGACACGCGGGAACAAGACCCGCACGACCAGCGTTAGCATGCTCAGTAAGTTGAGTGCATCAGGCTCAGCTCTGAATTGACAGCACACCGCCGTCGAGGCAAGCTTGAGCGGGGTGCACTCATCATAGTGCAGGAAAGAAGCTCTACATATTCAGGAGGATTCACC","PARTICLE_A":"PROTEIN","PARTICLE_B":"GEN","STD_COUNTER_REPITED":"0.50","PROMOTES_COORDINATES_GENOMA":"419485..419834","GENE_IN_OPERON":"Rv0350;Rv0351","TARGET_PROMOTER":"Rv0350_up","COUNTER_REPITED":"1","INTERACTION_TYPE":"PROTEIN-GEN","FOUND":"DATASET_2"}	{"PROMOTER_SEQUENCE":"CGCCCGCCGCTGCACCCGAGCCAGCGGCGATTTGGCCGAAGCGGTGATCTGGGGCATACTGTTCAGGTTGCCCTGCGCCAGGTTTGCGTCTGGCCGGGCATACGACCAGCGCCCACACGGGCGCGTCCGGTCCCACCCTTGAAGCGCGACGATTTCGGCCTTGAAATTGATCGCGACAAGGCTGTGTGCGGGCGACACGCCCGAGCGCGGGGCCGGTGGACCTACGACAGGTAAACAGCGGCGCAGTATTCGGCGCAACGCTAGATCGGTCCAGAAGGACCGGGTCGATCGGCGCGCCGGGGAGCACCGGACCCGGATACGGGCTCGAGTGGGAGTGAGGTAGGAGAAGC","PARTICLE_A":"PROTEIN","STD_COUNTER_REPITED":"0.50","PARTICLE_B":"GEN","PROMOTES_COORDINATES_GENOMA":"800137..800486","GENE_IN_OPERON":"Rv0700;Rv0701;Rv0702;Rv0703","TARGET_PROMOTER":"Rv0700_up","COUNTER_REPITED":"1","INTERACTION_TYPE":"PROTEIN-GEN","FOUND":"DATASET_2"}	{"PROMOTER_SEQUENCE":"CTCCTGGCTCGATGAAAGAATCAGCGGCCGCACTCGGCGAGCGCTGCTCAGCGACCGTCAGCACGCCAGTTCAATACGGCGAGAAGCGCGTTCTCATCGAAAGTCGGTCAAGCTGGCGGACTGAGCACTGCTCTCCGGGCTTGACCGGGGCCTCTCCCAGCTACGCCCCGAGCGTGTGCCCTGCCGACACGCGGGAACAAGACCCGCACGACCAGCGTTAGCATGCTCAGTAAGTTGAGTGCATCAGGCTCAGCTCTGAATTGACAGCACACCGCCGTCGAGGCAAGCTTGAGCGGGGTGCACTCATCATAGTGCAGGAAAGAAGCTCTACATATTCAGGAGGATTCACC","PARTICLE_A":"PROTEIN","PARTICLE_B":"GEN","STD_COUNTER_REPITED":"0.50","PROMOTES_COORDINATES_GENOMA":"419485..419834","GENE_IN_OPERON":"Rv0350;Rv0351","TARGET_PROMOTER":"Rv0350_up","COUNTER_REPITED":"1","INTERACTION_TYPE":"PROTEIN-GEN","FOUND":"DATASET_2"}

Tabla 14: Información de nodos y relaciones de los 3 quasi-cliques tomados del prototipo de consultas 3.

6.2.4 Prototipo de consulta 4 - Relaciones consecutivas y proteínas o genes en nombres y anotaciones.

Pregunta biológica: ¿Cuáles son los grupos de 3 proteínas o genes que contiene las proteínas o genes relevantes en los trabajos con referencias bibliográficas [7], [8], [9], [10], [11] y que tienen una interacción proteína-proteína, luego una interacción de co-expresión y finalmente tiene una interacción de regulación o de co-expresión?

Resumen: 1-quasi-cliques k=3, con variaciones PPIs hacia co-expresión y co-expresión o zona regulatoria, filtrando por las proteínas y genes de cada artículo asociado a los conjuntos de datos que componen el multígrafo.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 1-quasi-clique con k=3.
2. La primera relación corresponde a datos de interacción proteína-proteína.
3. La segunda relación corresponde a datos de co-expresión.
4. La tercera relación corresponde a una co-expresión o a una región reguladora.
5. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.
6. Se incluyen solo los quasi-cliques que tengan en alguno de sus componentes alguna proteína o gen en su ID, PREFERED_NAME o ANNOTATIONS según sea el caso. Así:
   - Cuando se usa el ID se está limitado, ya que en muchas publicaciones se usa lo que se conoce como nombre predeterminado (PREFERED_NAME) y por ende primero habría que realizar una búsqueda del ID equivalente y luego evaluarlo. Justamente una de los beneficios de realizar un enriquecimiento del multígrafo es evitar buscar información – en la mayoría de los casos – externamente.
   - En las anotaciones de algunos genes, existen posibles funciones regulatorias de una proteína o funciones hipotéticas de éstas y genes, por lo que resulta muy útil buscar dichos genes o proteínas en el texto de dichas anotaciones.
   - Al unir los tres atributos antes mencionados, se consiguen generar quasi-cliques con alta información mutua que puede resultar de interés, aunque existen ocasiones en que el uso de las anotaciones no trae ningún resultado positivo

A continuación, se muestra el análisis para cada uno de los 5 trabajos correspondientes a los datos usados para construir el multígrafo, sin embargo, este mismo procedimiento se puede realizar con cualquier conjunto de proteínas o genes que se quiera analizar no importando la fuente del cual se extrajeron. En la **Tabla 15** se pueden ver los genes y proteínas que se tuvieron en cuenta para cada trabajo y que los autores sugieren como relevantes en los mismos.

Número trabajo	# de genes y proteínas	Listado de genes y proteínas
1 [7]	3	v2752c, Rv1354c, pknk
2 [8]	3	mce1r, devr, pop
3 [9]	1	Whib
4 [10]	10	Rv0467, Rv1621c, Rv1622c, Rv1623c, Rv0251c, pdhb, groes, dnak, pdhb, dos
5 [11]	2	tgs1, tg

Tabla 15: Listado de genes y proteínas relevantes por cada trabajo relacionado con los datos que componen el multigrafo.

1. Genes y proteínas correspondientes al trabajo 1 [7]:

En NEO4J mediante el lenguaje de consultas CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND b<>c AND c<>a AND
        ((a.ID in["Rv2752c","Rv1354c"] OR lower(a.PREFERED_NAME) = "pknk") OR
        (b.ID in["Rv2752c","Rv1354c"] OR lower(b.PREFERED_NAME) = "pknk") OR
        (c.ID in["Rv2752c","Rv1354c"] OR lower(c.PREFERED_NAME) = "pknk"))
RETURN a,b,c,r1,r2,r3
```

En Titan DB mediante el lenguaje de consultas Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').bothV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffe
ct{C = it}.filter{B!=C && A!=C && ((A.attach().property('ID_').value().equals("Rv2752c")||
A.attach().property('ID_').value().equals("Rv1354c") ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "pknk") ||
(B.attach().property('ID_').value().equals("Rv2752c") || B.attach().property('ID_').value().equals("Rv1354c") ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "pknk") ||
(C.attach().property('ID_').value().equals("Rv2752c") || C.attach().property('ID_').value().equals("Rv1354c") ||
C.attach().property('PREFERED_NAME').value().toLowerCase() ==
"pknk"))}.bothE('INTERACTION_REGULATION_1','INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPR
ESSION_2').bothV().sideEffect{A_ = it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}-
>${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}->${it.get(4).property('ID_')}-
>${it.get(5).label()}"}
```

En la **Ilustración *81***, se pueden ver tres nodos y seis relaciones, de las cuales tres son relaciones complementarias entre nodos y tres corresponden al quasi-clique encontrado. En la parte inferior se pueden ver los atributos del nodo Rv1354c el cual resulta interesante ya que además de hacer parte del quasi-clique, tiene dos relaciones consigo mismo distintas: una de interacción proteína-proteína y otra de regulación. Vemos que, en éste caso, se hace uso tanto del filtro por ID como de PREFERED_NAME de manera combinada, sin embargo, el uso de ANNOTATIONS no mejora los resultados.

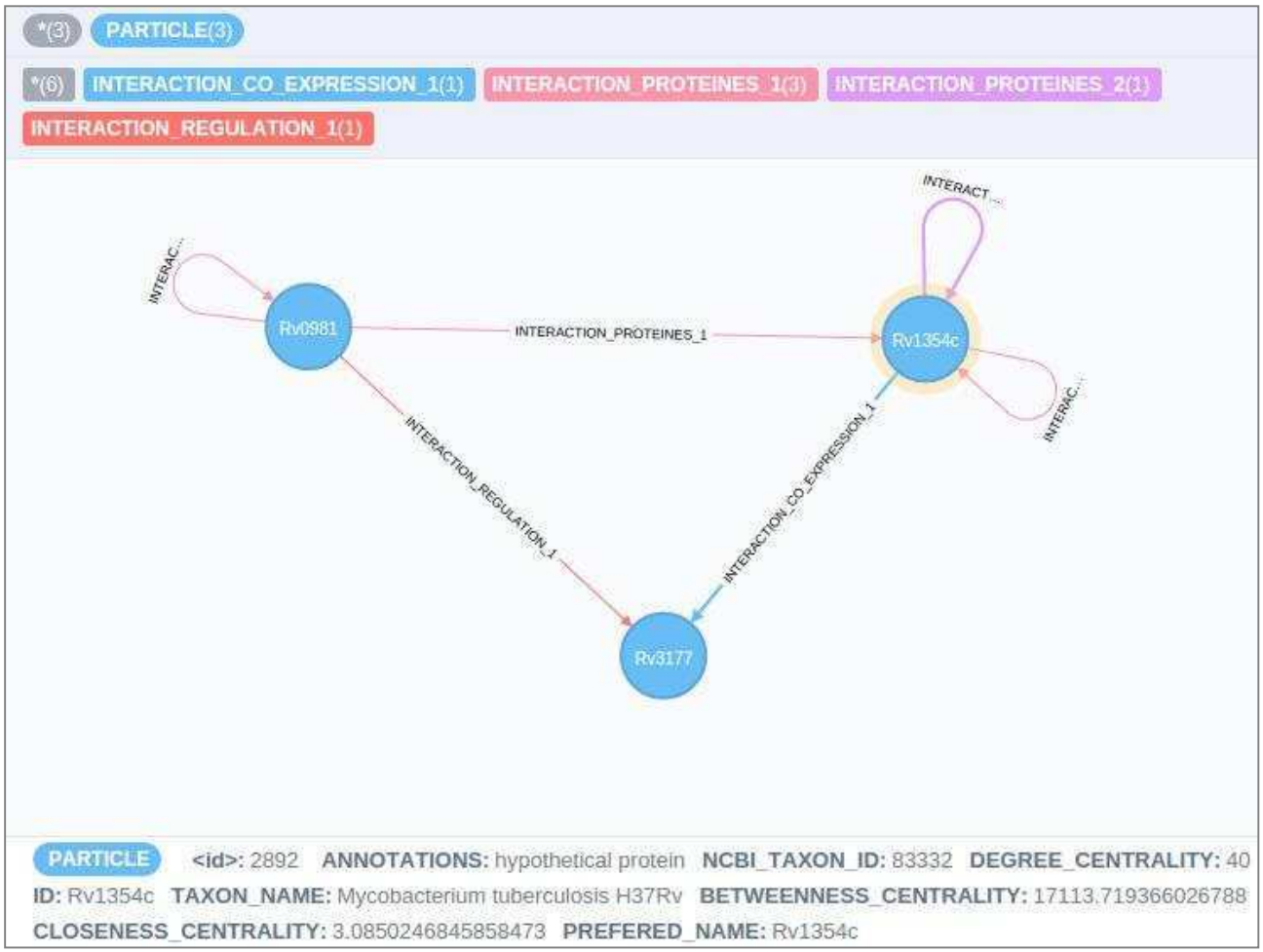

Ilustración 81: Prototipo de consulta 4 para el artículo 1, que muestra un quasi-clique encontrado filtrando las proteínas y genes relevantes en el artículo mencionado.

2. Genes y proteínas correspondientes al trabajo 2 [8]:

En NEO4J mediante el lenguaje de consultas CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND b<>c AND c<>a AND
        (lower(a.PREFERED_NAME) IN["mce1r","devr","phop"] OR
        lower(b.PREFERED_NAME) IN["mce1r","devr","phop"] OR
        lower(c.PREFERED_NAME) IN["mce1r","devr","phop"] OR
        (lower(a.ANNOTATIONS) CONTAINS "mce1r" OR lower(a.ANNOTATIONS) CONTAINS "devr" OR
lower(a.ANNOTATIONS) CONTAINS "phop") OR
        (lower(b.ANNOTATIONS) CONTAINS "mce1r" OR lower(b.ANNOTATIONS) CONTAINS "devr" OR
lower(b.ANNOTATIONS) CONTAINS "phop") OR
        (lower(c.ANNOTATIONS) CONTAINS "mce1r" OR lower(c.ANNOTATIONS) CONTAINS "devr" OR
lower(c.ANNOTATIONS) CONTAINS "phop"))
RETURN a,b,c,r1,r2,r3
```

En Titan DB mediante el lenguaje de consultas Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').bothV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffe
ct{C = it}.filter{B!=C && A!=C && ((A.attach().property('PREFERED_NAME').value().toLowerCase()=="mce1r" ||
A.attach().property('PREFERED_NAME').value().toLowerCase()=="devr" ||
A.attach().property('PREFERED_NAME').value().toLowerCase()=="phop") ||
(A.attach().property('ANNOTATIONS').value().toLowerCase().contains("mce1r") ||
A.attach().property('ANNOTATIONS').value().toLowerCase().contains("devr") ||
A.attach().property('ANNOTATIONS').value().toLowerCase().contains("phop")) ||
(B.attach().property('PREFERED_NAME').value().toLowerCase()=="mce1r" ||
B.attach().property('PREFERED_NAME').value().toLowerCase()=="devr" ||
B.attach().property('PREFERED_NAME').value().toLowerCase()=="phop") ||
(B.attach().property('ANNOTATIONS').value().toLowerCase().contains("mce1r") ||
B.attach().property('ANNOTATIONS').value().toLowerCase().contains("devr") ||
B.attach().property('ANNOTATIONS').value().toLowerCase().contains("phop")) ||
(C.attach().property('PREFERED_NAME').value().toLowerCase()=="mce1r" ||
C.attach().property('PREFERED_NAME').value().toLowerCase()=="devr" ||
C.attach().property('PREFERED_NAME').value().toLowerCase()=="phop") ||
(C.attach().property('ANNOTATIONS').value().toLowerCase().contains("mce1r") ||
C.attach().property('ANNOTATIONS').value().toLowerCase().contains("devr") ||
C.attach().property('ANNOTATIONS').value().toLowerCase().contains("phop"))))}.bothE('INTERACTION_REGUL
ATION_1','INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffect{A_ =
it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}->${it.get(1).label()}->${it.get(2).property('ID_')}-
>${it.get(3).label()}->${it.get(4).property('ID_')}->${it.get(5).label()}"}
```

En la **Ilustración 82**, se pueden ver 11 nodos y 15 relaciones, de las cuales tres son relaciones complementarias entre nodos y 12 corresponden a los 4 quasi-cliques encontrados. En la parte inferior se pueden ver los atributos del nodo Rv0757 que comparten dos quasi-cliques y cuya estructura es interesante para el análisis.

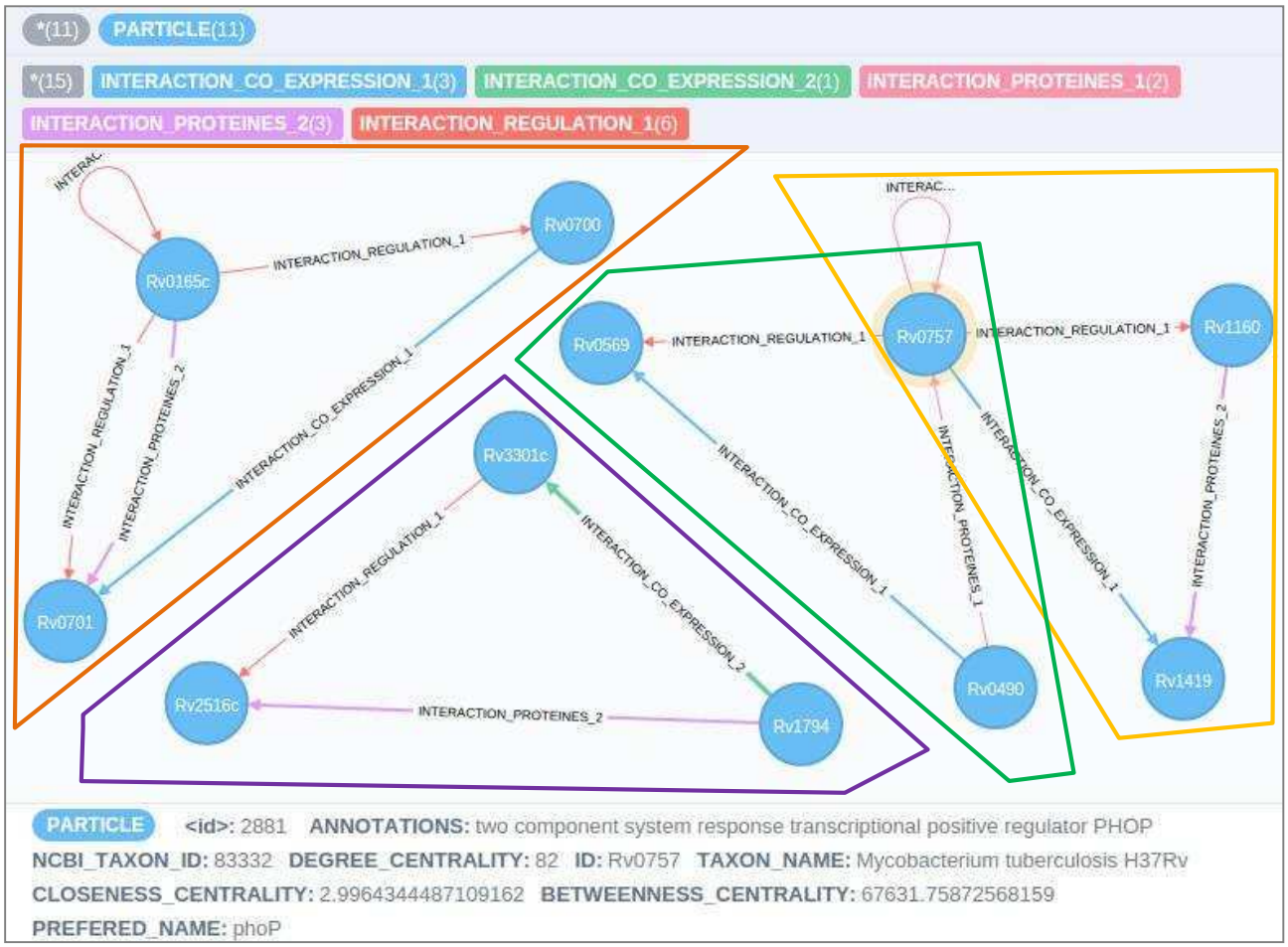

Ilustración 82: Prototipo de consulta 4 para el artículo 2, que muestra 4 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo.

Para el caso planteado, al realizar un análisis solo filtrando con ID y PREFERED_NAME, se obtienen 3 quasi-cliques, sin embargo, al agregar la búsqueda en las ANNOTATIONS se agrega un cliqué más al análisis correspondiente al encerrado en el polígono violeta de la **Ilustración 82**.

3. Genes y proteínas correspondientes al trabajo 3 [9]:

En NEO4J/CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND b<>c AND c<>a AND
        ((lower(a.ANNOTATIONS) CONTAINS "whib" ) OR
        (lower(b.ANNOTATIONS) CONTAINS "whib" ) OR
        (lower(c.ANNOTATIONS) CONTAINS "whib" ))
RETURN a,b,c,r1,r2,r3
```

En Titan DB/Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').bothV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffe
ct{C = it}.filter{B!=C && A!=C &&
((A.attach().property('ANNOTATIONS').value().toLowerCase().contains("whib")) ||
(B.attach().property('ANNOTATIONS').value().toLowerCase().contains("whib")) ||
(C.attach().property('ANNOTATIONS').value().toLowerCase().contains("whib")))}.bothE('INTERACTION_REGU
LATION_1','INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffect{A_ =
it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}->${it.get(1).label()}->${it.get(2).property('ID_')}-
>${it.get(3).label()}->${it.get(4).property('ID_')}->${it.get(5).label()}"}
```

En la **Ilustración 83**, se pueden ver seis nodos y siete relaciones, de las cuales una es complementaria entre nodos y 6 corresponden a los dos quasi-cliques encontrados. En la parte inferior se pueden ver los atributos del nodo Rv3416 el cual resulta interesante debido a que tiene una interacción proteína-proteína hacia el nodo Rv0701 y a su vez este tiene una interacción regulatoria hacía el primero. Vemos que, en éste caso, no se hace uso del filtro por ID y el PREFERED_NAME, como en las anteriores ocasiones, sin embargo, el uso de ANNOTATIONS genera algunos resultados.

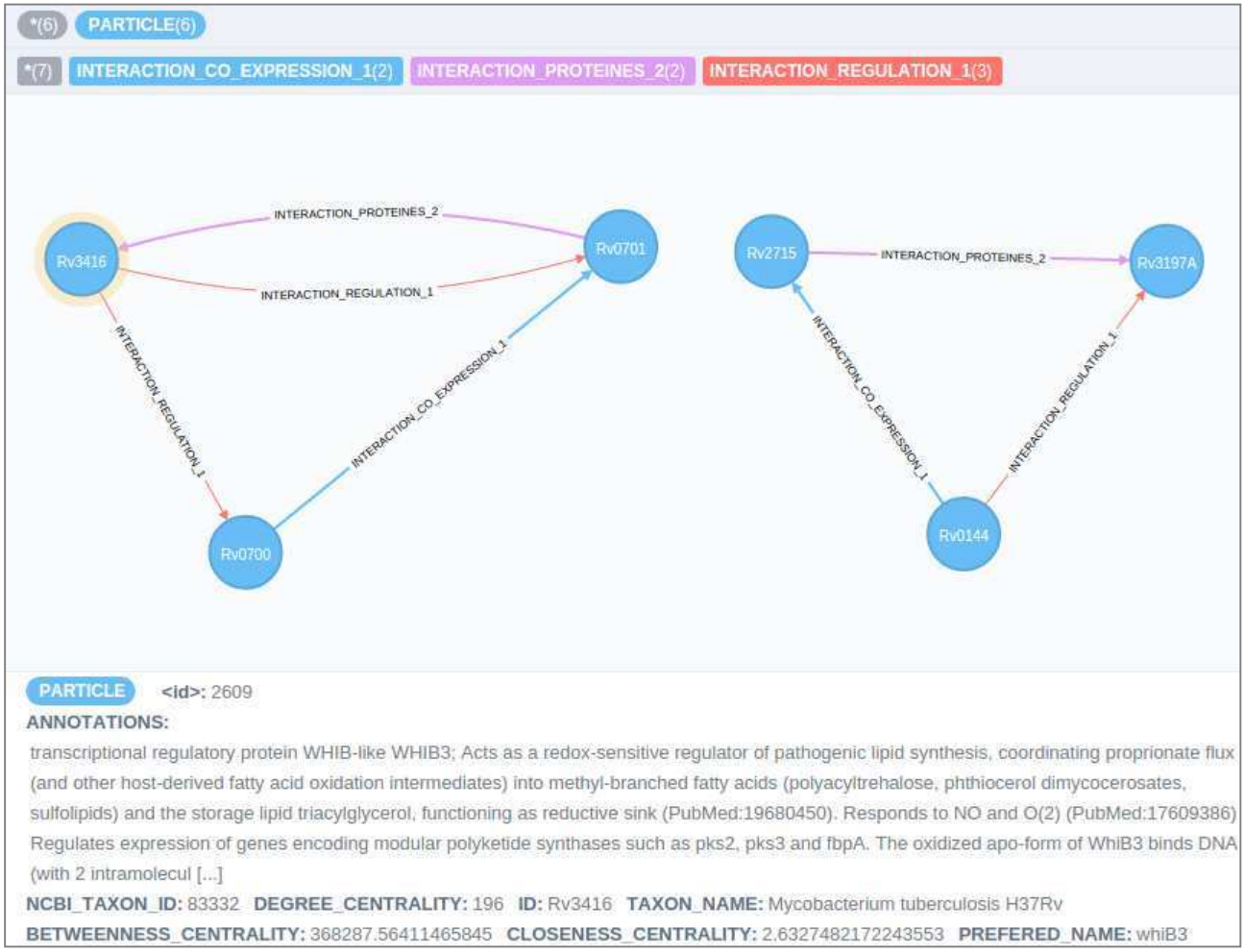

Ilustración 83: Prototipo de consulta 4 para el artículo 3, que muestra dos quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho el artículo.

4. Genes y proteínas correspondientes al trabajo 4 [10]:

En NEO4J CYPHER la consulta generada es la siguiente:

```
MATCH (a)-[r1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2]-(b)-
[r2:INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[r3:INTERACTION_REGULATION_1|INTERACTION_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND b<>c AND c<>a AND
        (a.ID in["Rv0467","Rv1621c","Rv1622c","Rv1623c","Rv0251c"] OR lower(a.PREFERED_NAME)
IN["pdhb","groes","dnak","pdhb","doss"] OR
        b.ID in["Rv0467","Rv1621c","Rv1622c","Rv1623c","Rv0251c"] OR lower(b.PREFERED_NAME)
IN["pdhb","groes","dnak","pdhb","doss"] OR
        c.ID in["Rv0467","Rv1621c","Rv1622c","Rv1623c","Rv0251c"] OR lower(c.PREFERED_NAME)
IN["pdhb","groes","dnak","pdhb","doss"])
RETURN a,b,c,r1,r2,r3
```

En Titan DB/Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A =
it}.bothE('INTERACTION_PROTEINES_1','INTERACTION_PROTEINES_2').bothV().sideEffect{B =
it}.filter{A!=B}.bothE('INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPRESSION_2').bothV().sideEffe
ct{C = it}.filter{B!=C && A!=C && ((A.attach().property('ID_').value()=="Rv0467" ||
A.attach().property('ID_').value()=="Rv1621c" || A.attach().property('ID_').value()=="Rv1622c" ||
A.attach().property('ID_').value()=="Rv0251c" ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "groes" ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "dnak" ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
A.attach().property('PREFERED_NAME').value().toLowerCase() == "doss") ||
(B.attach().property('ID_').value()=="Rv0467" || B.attach().property('ID_').value()=="Rv1621c" ||
B.attach().property('ID_').value()=="Rv1622c" || B.attach().property('ID_').value()=="Rv0251c" ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "groes" ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "dnak" ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
B.attach().property('PREFERED_NAME').value().toLowerCase() == "doss") ||
(C.attach().property('ID_').value()=="Rv0467" || A.attach().property('ID_').value()=="Rv1621c" ||
C.attach().property('ID_').value()=="Rv1622c" || C.attach().property('ID_').value()=="Rv0251c" ||
C.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
C.attach().property('PREFERED_NAME').value().toLowerCase() == "groes" ||
C.attach().property('PREFERED_NAME').value().toLowerCase() == "dnak" ||
C.attach().property('PREFERED_NAME').value().toLowerCase() == "pdhb" ||
C.attach().property('PREFERED_NAME').value().toLowerCase() ==
"doss"))}.bothE('INTERACTION_REGULATION_1','INTERACTION_CO_EXPRESSION_1','INTERACTION_CO_EXPR
ESSION_2').bothV().sideEffect{A_ = it}.filter{A==A_}.path().collect{"${it.get(0).property('ID_')}-
>${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}->${it.get(4).property('ID_')}-
>${it.get(5).label()}"}
```

Para la consulta sin filtros de límite, se obtienen 14 nodos y 44 relaciones, de las cuales 23 son relaciones complementarias; todo lo anterior para 8 quasi-cliques, como podemos observar en la **Ilustración 84**. Es interesante mencionar que al realizar los filtros únicamente por las ANNOTATIONS, se encuentran los mismo cliques que realizando la búsqueda por ID y PREFERED_NAME.

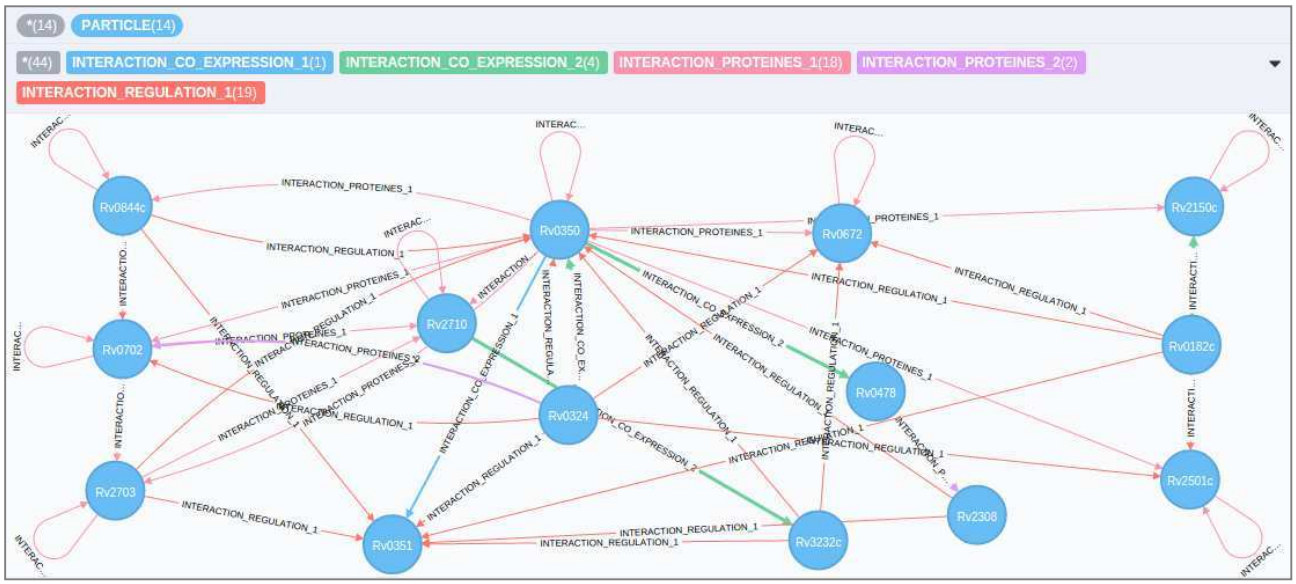

Ilustración 84: Prototipo de consulta 4 para el artículo 4, que muestra 8 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo.

Para realizar un análisis más profundo, tomaremos 3 de los 8 quasi-cliques hallados. En la **Ilustración 85**, se pueden ver 7 nodos y 12 relaciones, de las cuales 3 son complementarias entre nodos y 9 corresponden a los quasi-cliques encontrados. En la parte inferior se pueden ver los atributos del nodo Rv0350 el cual presenta el mayor grado entre los nodos y adicionalmente está en cada uno de los tres quasi-cliques.

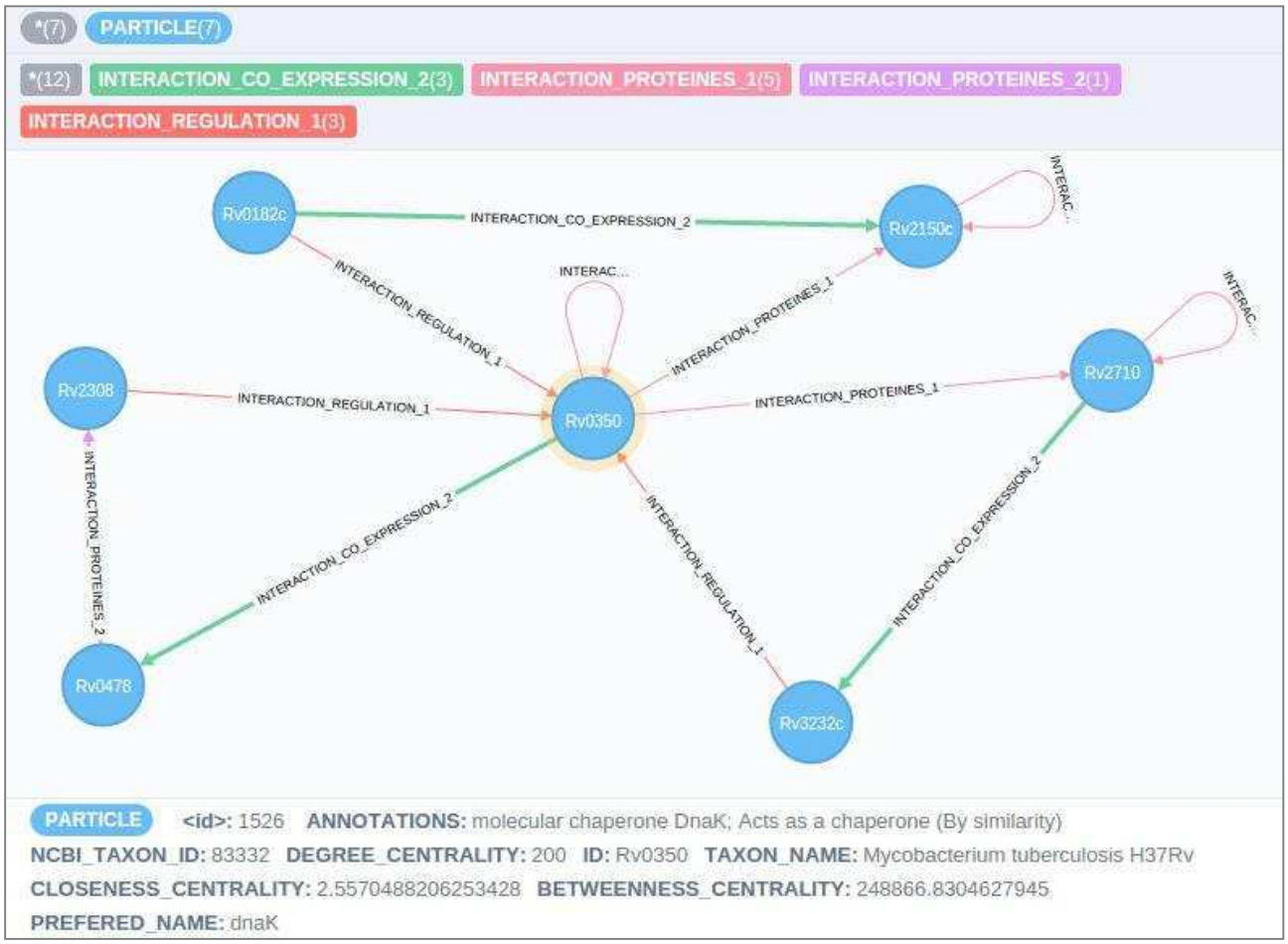

Ilustración 85: Prototipo de consulta 4 para el artículo 4, que muestra 3 de los 8 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho artículo.

5. Genes y proteínas correspondientes al trabajo 5 [11]**:**

Para las proteínas y genes correspondientes al trabajo 5, no se pudo encontrar ningún quasi-clique que siguiera las delimitaciones del prototipo de consulta 4.

6.2.5 Prototipo de consulta 5 - Número de relaciones de tipo distinto.

Pregunta biológica: ¿Cuáles son los grupos de tres proteínas o genes que tienen dos de sus tres relaciones con tipo distinto (Interacción proteína-proteína, interacción región promotora, interacción de co-expresión)?

Resumen: 1-quasi-cliques k=3, con dos relaciones distintas entre sí de las tres posibles.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 1-quasi-clique con k=3.
2. Es posible cualquier tipo de relación en cada parte del quasi-clique, solo que entre las tres, dos deben ser distintas.
3. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.

En NEO4J/CYPHER la consulta generada es la siguiente:

```
MATCH (a)-
[x1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(b)-
[x2:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[x3:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
 a<>b AND b<>c AND c<>a AND
 ((x1.INTERACTION_TYPE <> x2.INTERACTION_TYPE AND x2.INTERACTION_TYPE <> x3.INTERACTION_TYPE)
OR
 (x2.INTERACTION_TYPE <> x3.INTERACTION_TYPE AND x3.INTERACTION_TYPE <> x1.INTERACTION_TYPE)
OR
 (x3.INTERACTION_TYPE <> x1.INTERACTION_TYPE AND x1.INTERACTION_TYPE <> x2.INTERACTION_TYPE))
RETURN
 a,b,c,x1,x2,x3
```

En Titan DB/Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A = it}.bothE().sideEffect{x1 = it}.bothV().sideEffect{B = it}.filter{A!=B}.bothE().sideEffect{x2 =
it}.bothV().sideEffect{C = it}.filter{B!=C && A!=C}.bothE().sideEffect{x3 = it}.bothV().sideEffect{A_ =
it}.filter{A==A_}.filter{((x1.attach().property('INTERACTION_TYPE').value() !=
x2.attach().property('INTERACTION_TYPE').value() && x2.attach().property('INTERACTION_TYPE').value() !=
x3.attach().property('INTERACTION_TYPE').value()) || (x2.attach().property('INTERACTION_TYPE').value() !=
x3.attach().property('INTERACTION_TYPE').value() && x3.attach().property('INTERACTION_TYPE').value() !=
x1.attach().property('INTERACTION_TYPE').value()) || (x3.attach().property('INTERACTION_TYPE').value() !=
x1.attach().property('INTERACTION_TYPE').value() && x1.attach().property('INTERACTION_TYPE').value() !=
x2.attach().property('INTERACTION_TYPE').value()))}.path().collect{"${it.get(0).property('ID_')}-
>${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}->${it.get(4).property('ID_')}-
>${it.get(5).label()}"}.count()
```

Los filtros de la consulta para hallar quasi-cliques parecen ser muy pocos, ya que se generan 15.672, compuestos por 720 nodos y 10.753 relaciones, lo cual no permite un fácil análisis de los resultados en primera instancia. Para esto se filtran las proteínas y genes relevantes para el trabajo número 5 que conforma el multígrafo, específicamente en las ANNOTATIONS y se eligen los primeros 3 quasi-cliques de 126 encontrados.

En / CYPHER la consulta generada es la siguiente:

```
MATCH (a)-
[x1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(b)-
[x2:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[x3:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
 a<>b AND b<>c AND c<>a AND
 ((x1.INTERACTION_TYPE <> x2.INTERACTION_TYPE AND x2.INTERACTION_TYPE <> x3.INTERACTION_TYPE)
OR
 (x2.INTERACTION_TYPE <> x3.INTERACTION_TYPE AND x3.INTERACTION_TYPE <> x1.INTERACTION_TYPE)
OR
 (x3.INTERACTION_TYPE <> x1.INTERACTION_TYPE AND x1.INTERACTION_TYPE <> x2.INTERACTION_TYPE))
AND
(lower(a.ANNOTATIONS) CONTAINS "tgs1" OR lower(a.ANNOTATIONS) CONTAINS "tg" OR
        lower(b.ANNOTATIONS) CONTAINS "tgs1" OR lower(b.ANNOTATIONS) CONTAINS "tg" OR
        lower(c.ANNOTATIONS) CONTAINS "tgs1" OR lower(c.ANNOTATIONS) CONTAINS "tg" )
RETURN
 a,b,c,x1,x2,x3
limit 3
```

En Titan DB/ Gremlin la consulta generada es la siguiente:

```
g.V().sideEffect{A = it}.bothE().sideEffect{x1 = it}.bothV().sideEffect{B = it}.filter{A!=B}.bothE().sideEffect{x2 =
it}.bothV().sideEffect{C = it}.filter{B!=C && A!=C}.bothE().sideEffect{x3 = it}.bothV().sideEffect{A_ =
it}.filter{A==A_}.filter{((x1.attach().property('INTERACTION_TYPE').value() !=
x2.attach().property('INTERACTION_TYPE').value() && x2.attach().property('INTERACTION_TYPE').value() !=
x3.attach().property('INTERACTION_TYPE').value()) || (x2.attach().property('INTERACTION_TYPE').value() !=
x3.attach().property('INTERACTION_TYPE').value() && x3.attach().property('INTERACTION_TYPE').value() !=
x1.attach().property('INTERACTION_TYPE').value()) || (x3.attach().property('INTERACTION_TYPE').value() !=
x1.attach().property('INTERACTION_TYPE').value() && x1.attach().property('INTERACTION_TYPE').value() !=
x2.attach().property('INTERACTION_TYPE').value())) &&
(A.attach().property('ANNOTATIONS').value().toLowerCase().contains("tgs1") ||
A.attach().property('ANNOTATIONS').value().toLowerCase().contains("tg") ||
B.attach().property('ANNOTATIONS').value().toLowerCase().contains("tgs1") ||
B.attach().property('ANNOTATIONS').value().toLowerCase().contains("tg") ||
C.attach().property('ANNOTATIONS').value().toLowerCase().contains("tgs1") ||
C.attach().property('ANNOTATIONS').value().toLowerCase().contains("tg"))}.path().collect{"${it.get(0).propert
y('ID_')}->${it.get(1).label()}->${it.get(2).property('ID_')}->${it.get(3).label()}->${it.get(4).property('ID_')}-
>${it.get(5).label()}"} [0..2]
```

En la **Ilustración 86**, se pueden ver 4 nodos y 6 relaciones, de las cuales uno es complementario entre nodos y cinco corresponden a los 3 quasi-cliques analizados de los 126 encontrados; vemos que solo son 5 nodos principales ya que uno es compartido por dos de los cliques presentados. En la parte inferior se pueden ver los atributos de la relación de tipo CO-EXPRESION del nodo Rv0117 al Rv1994c el cual corresponde al conjunto de datos 5 [11] como se puede ver en el atributo FOUND.

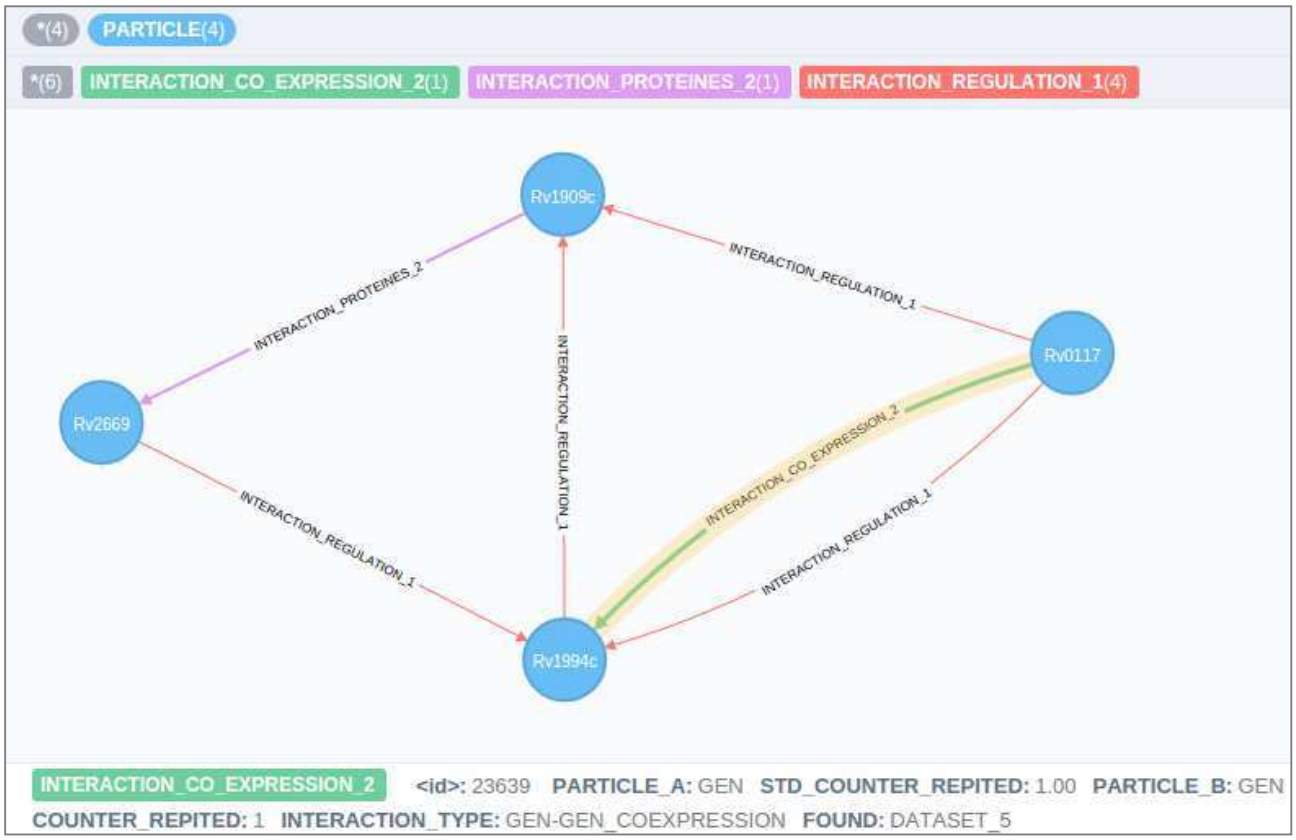

Ilustración 86: Prototipo de consulta 5 para el artículo 5, que muestra 3 de los 126 quasi-cliques encontrados filtrando las proteínas y genes relevantes en dicho el artículo.

6.2.6 Prototipo de consulta 6 - Relaciones de conjuntos de datos distintos.

Pregunta biológica: ¿Cuáles son los grupos de 4 proteínas o genes que tiene relaciones provenientes de conjuntos de datos o grafos distintos?

Resumen: 0.83-quasi-cliques k=4 con una no completitud de una relación y restricciones de diferencia entre todas las fuentes de las relaciones que lo componen.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 0.83-quasi-clique con k=4.
2. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.
3. Se limita a que las 5 relaciones deben ser distintas entre sí, teniendo en cuenta que hay 5 tipos de relaciones correspondientes a los 5 conjuntos de datos o grafos que componen el multigrafo mixto; de ésta forma se tienen quasi-cliques más homogéneos y limitados.

En NEO4j mediante el lenguaje de consultas CYPHER la consulta generada es la siguiente:

```
MATCH (a)-
[x1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(b)-
[x2:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[x3:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(d)-
[x4:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)-
[x5:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)
WHERE
a<>b AND b<>c AND c<>a AND c<>d AND d<>a AND b<>d AND
 type(x1) <> type(x2) AND
 type(x1) <> type(x3) AND
 type(x1) <> type(x4) AND
 type(x1) <> type(x5) AND
 type(x2) <> type(x3) AND
 type(x2) <> type(x4) AND
 type(x2) <> type(x5) AND
 type(x3) <> type(x4) AND
 type(x3) <> type(x5) AND
 type(x4) <> type(x5)
RETURN
        a.ID,type(x1),b.ID,type(x2),c.ID,type(x3),d.ID,type(x4),type(x5)
```

En Titan DB/Gremlin la consulta generada es la siguiente:

```
g.V().match(
                __.as('a').bothE().as('x1').map{it.get().label()}.as('label1'),
                __.as('x1').bothV().as('b'),
                __.as('b').bothE().as('x2').map{it.get().label()}.as('label2'),
                __.as('x2').bothV().as('c'),
                __.as('c').bothE().as('x3').map{it.get().label()}.as('label3'),
                __.as('x3').bothV().as('d'),
                __.as('d').bothE().as('x4').map{it.get().label()}.as('label4'),
                __.as('x4').bothV().as('a'),
                __.as('a').bothE().as('x5').map{it.get().label()}.as('label5'),
                __.as('x5').bothV().as('c')).
        where('label1', neq('label2')).where('label1', neq('label3')).where('label1',
neq('label4')).where('label1', neq('label5')).
        where('label2', neq('label3')).where('label2', neq('label4')).where('label2', neq('label5')).
        where('label3', neq('label4')).where('label3', neq('label5')).
        where('label4', neq('label5')).
        where('a', neq('b')).where('a', neq('c')).where('a', neq('d')).
        where('b', neq('c')).where('b', neq('d')).
        where('c', neq('d')).
        select('a','label1','b','label2','c','label3','d','label4','label5')
```

Se encuentran 4 quasi-cliques, compuestos por 4 nodos y 9 relaciones, presentándose una misma estructura central o núcleo, con variaciones de un quasi-clique a otro de una o dos relaciones distintas.

En la **Ilustración 87**, se pueden ver los cuatro quasi-cliques hallados. En la parte inferior se pueden observar los atributos del nodo Rv0702 (rpID, proteína ribosomal L4), el cual tiene un alto grado y forma parte de los cuatro quasi-cliques.

A partir de determinaciones de interacciones entre factores de transcripción y sitios promotores se observó que el promotor del gen Rv0350 (dnaK, hsp70) interactuaría con numerosos factores de transcripción. Y un alto porcentaje de casos (82%) pudo ser confirmado con otras técnicas de laboratorio [8]

Es notable que tanto el nodo Rv0350 como Rv0702 -mostrados en la ilustración en círculos rojos-, están presentes en los cuatro quasi-cliques y además estén entre los 5 nodos con más alto grado del multigrafo, lo cual puede sugerir el grado de importancia de los mismos y su posible influencia en uno o más mecanismo de regulación de tipo global.

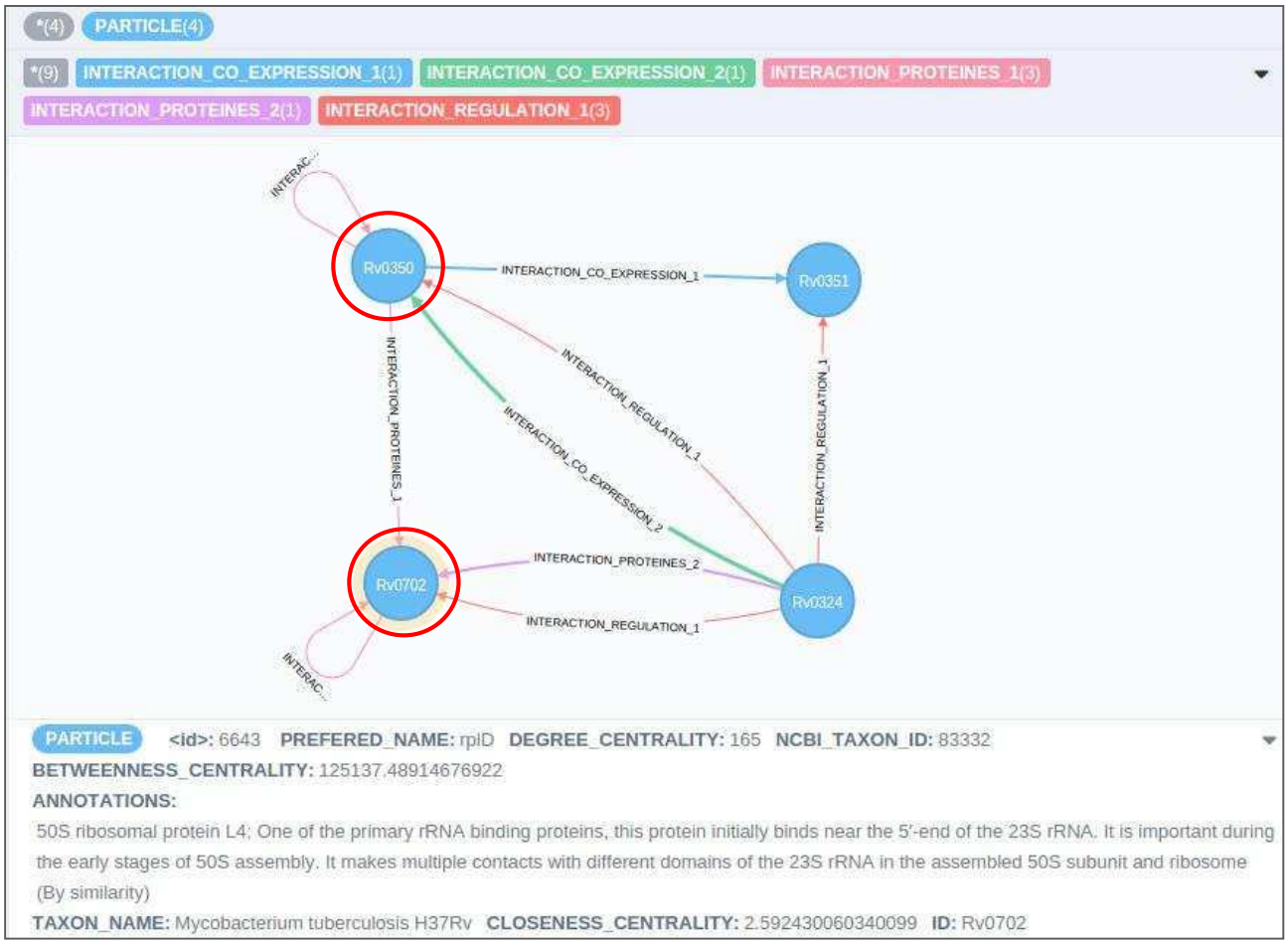

Ilustración 87: Prototipo de consulta 6, que muestra 4 quasi-cliques encontrados.

6.2.7 Prototipo de consulta 7 - Relaciones distintas consecutivas.

Pregunta biológica: ¿Cuáles son los grupos de 5 proteínas o genes que tiene relaciones consecutivas que a pares provienen de conjuntos de datos o grafos distintos?

Resumen: 1-quasi-cliques k=5, con restricciones de relaciones continuas diferenciadas.

Se desean encontrar estructuras de conexiones en el multigrafo representadas mediante un clique restringido o quasi-clique en donde se presenten las siguientes condiciones:

1. Es un 1-quasi-clique con k=5.
2. Se delimita con una sentencia no permitir que en un quasi-clique se presente una relación de un nodo consigo mismo.
3. Se limita que el tipo de relación r - sabiendo que existen cinco tipos de relaciones correspondientes a los cinco distintos conjuntos de datos - entre dos nodos A y B, sea distinta a la relación r-1 entre los nodos B y C, de ésta forma se tienen menos quasi-

cliques y más homogéneos.

En / NEO4J/CYPHER la consulta generada es la siguiente:

```
MATCH (a)-
[x1:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(d)-
[x2:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(e)-
[x3:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)-
[x4:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[x5:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(d)-
[x6:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(b)-
[x7:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(c)-
[x8:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(e)-
[x9:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTION
_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(b)-
[x10:INTERACTION_PROTEINES_1|INTERACTION_PROTEINES_2|INTERACTION_REGULATION_1|INTERACTIO
N_CO_EXPRESSION_1|INTERACTION_CO_EXPRESSION_2]-(a)
WHERE
        a<>b AND a<>c AND a<>d AND a<>d AND a<>e AND
        b<>c AND b<>d AND b<>e AND
        c<>d AND c<>e AND
        d<>e AND
        type(x1) <> type(x2) AND
        type(x2) <> type(x3) AND
        type(x3) <> type(x4) AND
        type(x4) <> type(x5) AND
        type(x5) <> type(x6) AND
        type(x6) <> type(x7) AND
        type(x7) <> type(x8) AND
        type(x8) <> type(x9) AND
        type(x9) <> type(x10)
RETURN
        a.ID,b.ID,c.ID,d.ID,e.ID,type(x1),type(x2),type(x3),type(x4),type(x5),type(x6),type(x7),type(x8),type(x
9),type(x10)
```

En Titan DB/Gremlin la consulta generada es la siguiente:

```
g.V().match(
                __.as('a').bothE().as('x1').map{it.get().label()}.as('label1'),
                __.as('x1').bothV().as('d'),
                __.as('d').bothE().as('x2').map{it.get().label()}.as('label2'),
                __.as('x2').bothV().as('e'),
                __.as('e').bothE().as('x3').map{it.get().label()}.as('label3'),
                __.as('x3').bothV().as('a'),
                __.as('a').bothE().as('x4').map{it.get().label()}.as('label4'),
                __.as('x4').bothV().as('c'),
                __.as('c').bothE().as('x5').map{it.get().label()}.as('label5'),
                __.as('x5').bothV().as('d'),
                __.as('d').bothE().as('x6').map{it.get().label()}.as('label6'),
                __.as('x6').bothV().as('b'),
                __.as('b').bothE().as('x7').map{it.get().label()}.as('label7'),
                __.as('x7').bothV().as('c'),
                __.as('c').bothE().as('x8').map{it.get().label()}.as('label8'),
                __.as('x8').bothV().as('e'),
                __.as('e').bothE().as('x9').map{it.get().label()}.as('label9'),
                __.as('x9').bothV().as('b'),
                __.as('b').bothE().as('x10').map{it.get().label()}.as('label10'),
                __.as('x10').bothV().as('a')).
        where('label1', neq('label2')).where('label2', neq('label3')).where('label3', neq('label4')).
        where('label4', neq('label5')).where('label5', neq('label6')).where('label6', neq('label7')).
        where('label7', neq('label8')).where('label8', neq('label9')).where('label9', neq('label10')).
        where('a', neq('b')).where('a', neq('c')).where('a', neq('d')).where('a', neq('e')).
        where('b', neq('c')).where('b', neq('d')).where('b', neq('e')).
        where('c', neq('d')).where('c', neq('e')).
        where('d', neq('e')).
        select('a','b','c','d','e','label1','label2','label3','label4','label5','label6','label7','label8','label9','label1')
```

En la **Ilustración 88**, se pueden ver 7 nodos y 26 relaciones. Vemos que el quasi-clique inicial marcado con verde en el polígono de la imagen, comparte con el segundo quasi-clique 3 nodos, y los otros nodos que lo componen son los correspondientes a las proteínas laterales Rv0001 y Rv0165c. Al igual que en el prototipo 6, vemos que las proteínas Rv3416, Rv0702 (marcadas en círculos rojos en la ilustración), hacen parte de las 5 proteínas con mayor grado, además de que la primera es la de mayor grado de intermediación del multigrafo, lo cual puede sugerir el grado de importancia de las mismas y las que componen ambos quasi-clique y su posible influencia en un mecanismo de regulación.

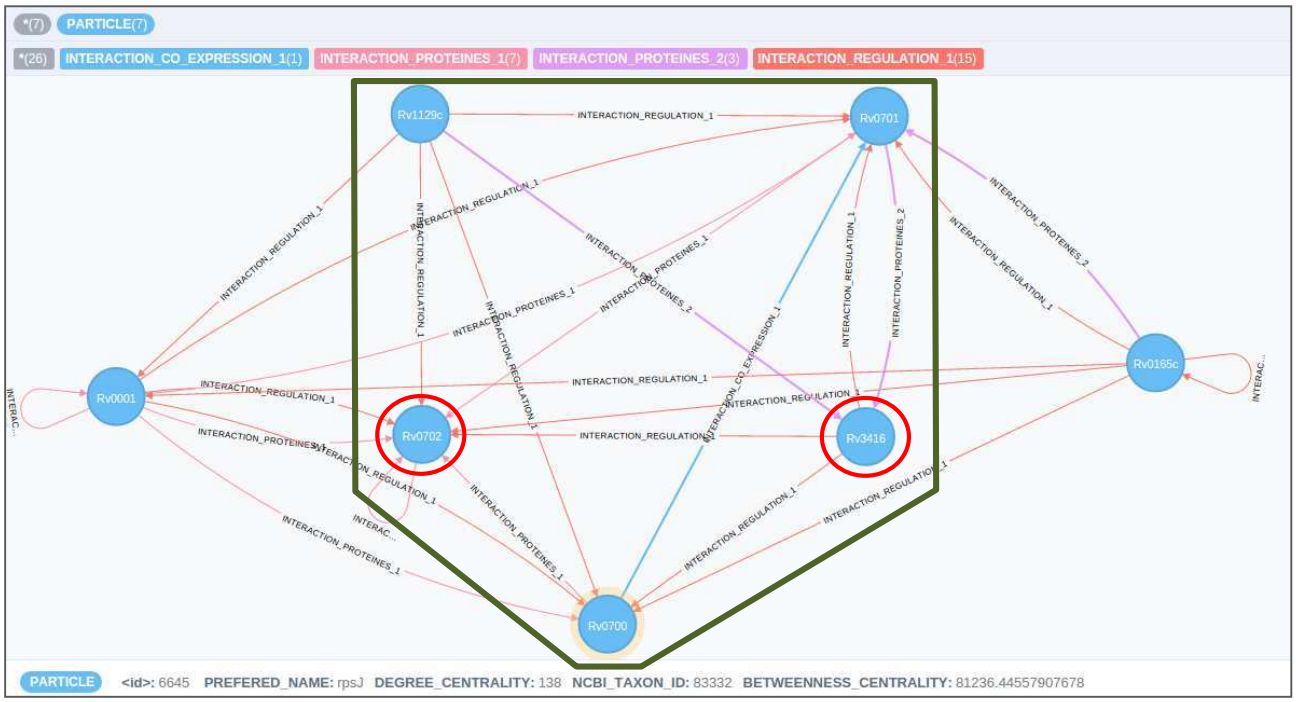

Ilustración 88: Prototipo de consulta 7, que muestra dos quasi-cliques con k=5.

6.3 ANÁLISIS BIOLÓGICO DE RESULTADOS

Para el análisis de los resultados de los prototipos presentados se eligieron los correspondientes al prototipo 6 para ejemplificar el proceso final que a nivel biológico se podría realizar con los patrones planteados sobre el multigrafo mixto. Debido a que el objetivo del presente trabajo no es desarrollar e interpretar todos los resultados de los patrones obtenidos, se dejarán el análisis de los demás resultados como un trabajo futuro.

A continuación, se presentará el análisis que plantea una configuración de funciones regulatoria de los resultados correspondientes al prototipo 6 presentado anteriormente (Ver **Ilustración *87***).

Según [8] RV0350 y Rv0351 forman parte de un operón, por lo cual es razonable que aparezcan relacionadas en alguno de los conjuntos de datos de co-expresión. Según las predicciones de la base de datos de operones *DOOR* (http://csbl.bmb.uga.edu/DOOR/) [69, 70, 71] el operón también incluiría a Rv0352 y Rv0353. Este operón de cuatro genes también fue descripto experimentalmente en [72].

Estos cuatro genes están caracterizados y sus productos son:

1. Rv0350, *DnaK*, con función de chaperona
2. Rv0351, *GrpE*, una proteína de respuesta al *"heat shock*"
3. Rv0352, *DnaJ*, otra chaperona
4. Rv0353, *HspR*, otra proteína de respuesta al "*heat shock*".

Según string-db [60] *DnaJ*, *DNAK* y *GrpE* participan activamente y en forma conjunta en las respuestas al shock hiperosmótico y al de calor, previniendo la agregación de proteínas desnaturalizadas por estrés y promueve su desagregación. Este sistema fue originalmente descripto en *Escherichai coli*, pero también se encontró en *M. tuberculosis*

[72].

El quasi-clique indica que hay una relación de regulación tipo proteína-gen entre Rv0324 (un regulador transcripcional de la familia *ArsR*) con Rv0350 y Rv0351. Esto sugiere que Rv0324 actuaría sobre la regulación del operón completo (Ver **Ilustración *89***, recuadro verde superior). Esta relación no está descripta en string-db. Además, el cuasi-clique también indica que existe una relación de co-expresión entre Rv0324 y Rv0350, que debería extenderse al operón. La familia *SmtB*/*ArsR* se describió originalmente como un represor del operón de resistencia al arsénico, pero posteriormente se le asignó una función más general en la resistencia a iones de otros metales pesados, además de arsénico, como plomo, cadmio y bismuto [73].

Rv0702 es la proteína L4 de la subunidad ribosomal de 50S. El quasi-clique mostrado revela interacciones proteína – proteína entre L4 y Rv0324, además de una actividad regulatoria del tipo proteína – gen de Rv0324 sobre el operón que codifica para las proteínas ribosomales S10, L3, L4 y L23.

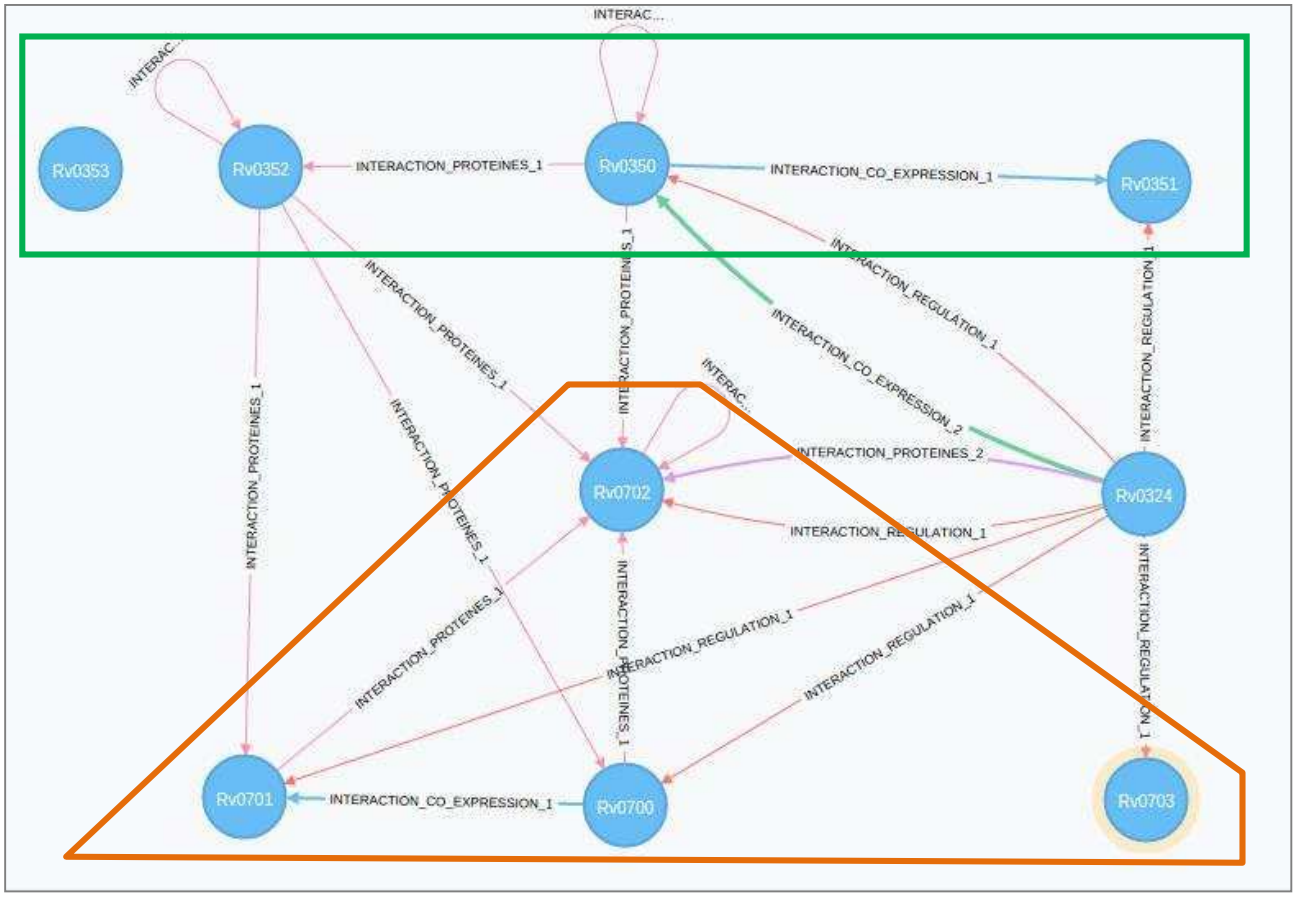

***Ilustración* 89:** Quasi-clique enriquecido con componentes adicionales de operones.

Se puede observar una interacción proteína proteína entre Rv0324 y solo un elemento (Rv0702) del operón S10, L3, L4 y L2 (Rv0700, Rv0701, Rv0702 y Rv0703). Ninguna de las relaciones ha sido documentada (con alguna parte del operón) por lo parecería despreciable para el análisis; sin embargo, serían necesario una prueba de laboratorio para descartar dicha relación.

Al agregar al quasi-clique los otros genes del operón *dnaKEJ*, se observa que también

hay interacciones proteína-proteína entre Rv0350 (*DnaK*) y Rv0352 (*DnaJ*) con Rv0702 (Ver **Ilustración *89***, polígono marrón inferior).

En síntesis, esta información apunta a un rol más importante del regulador transcripcional Rv0324 sobre los operón *dnaKJE* y el que contiene a los genes de las proteínas ribosomales S10, L3, L4, y L23. Por ejemplo, y en línea con estas observaciones [74] encontraron que Rv0324 activaría al represor *KstR* (Rv3574), un regulador transcripcional de la familia *TetR* que interviene en la degradación del colesterol [75].

De ésta forma, el modelo hipotético planteado luego del análisis sería el presentado en la **Ilustración *88***.

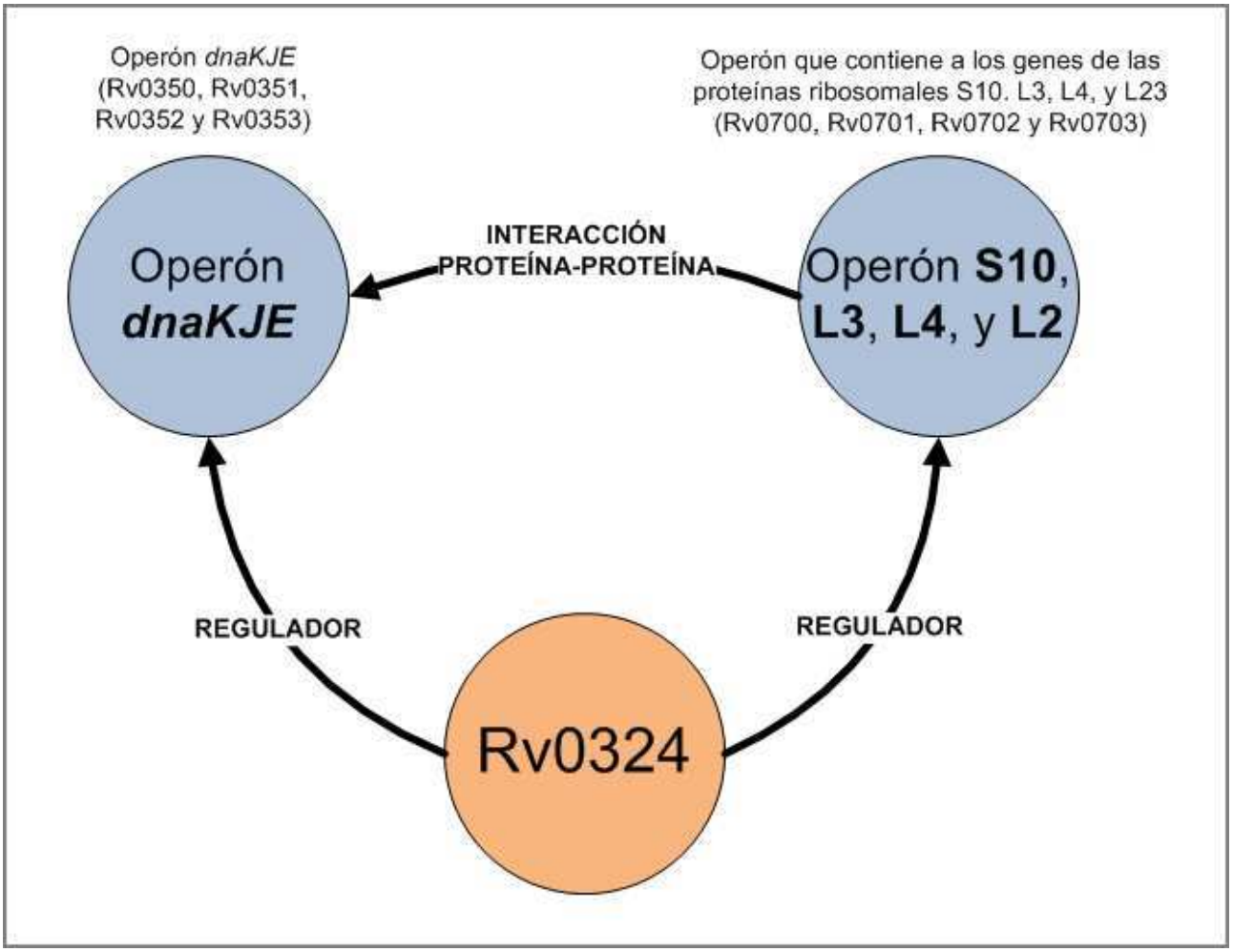

Ilustración 90: Modelo hipotético planteado para regulador transcripcional hallado en los quasi-cliques del prototipo 6.

6.4 RESUMEN GENERAL DE RESULTADOS EN LOS PROTOTIPOS DE CONSULTAS

Durante el proceso de construcción, ejecución y análisis de resultados de cada uno de los 7 prototipos de consultas planteados en el presente trabajo, surgieron los siguientes comentarios que apuntan – en general – a métodos para lograr encontrar patrones a nivel biológicos en multígrafos mixtos mediante estructuras como quasi-cliques:

- Los atributos de PUBLICATIONS no tuvieron un buen desempeño a la hora de buscar quasi-cliques, debido a que tan solo 1 conjunto de datos tiene dicho atributo, adicional a esto, pareciera que incluir además del código, el texto resumen pudiera generar un resultado muy parecido – y se presume mejorador – al obtenido al agregar las consultas en el atributo ANNOTATIONS. Adicionalmente, enriquecer los otros 4 grafos para que tengan ANNOTATIONS parece un camino orgánico para mejorar los resultados de las consultas.
- Los quasi-cliques con $k = 4, \gamma = 1$ y algunos otros, generan problemas al realizar las consultas en gremlin antes de la versión de Titan 1.0.0, en donde fue introducida la cláusula *MATCH.* El inconveniente surge al querer describir un clique siguiendo los caminos consecutivos de sus relaciones, lo que en algunos casos deja solo dos posibilidades para buscar el patrón*:*
    - No incluir las relaciones que no se pueden representar en un "solo trazo" del camino del clique consiguiendo quasi-cliques con un $0 < \gamma < 1$.
    - Agregar las relaciones adicionales sabiendo que el espacio de búsqueda va a ser limitado de ésta forma.

  Al respecto, en las versiones finales usadas tanto para Neo4j y Titan DB se encontraron capacidades para hallar relaciones en caminos no adyacentes. Así en Neo4j existe la cláusula MATCH que unida a OPTIONAL MATCH puede generar la consulta del patrón de un cliqué en caminos no consecutivos. Por otro lado, Titan DB posee la cláusula MATCH – con el mismo nombre que Neo4j – que recibe como parámetros patrones de relaciones traversas no consecutivas. Finalmente, si se quiere buscar cliques con k variable, se debe hacer uso de lenguajes más específicos y programar dichos comportamientos de forma personalizada.
- Para reducir el espacio de búsqueda de los prototipos de consultas 6 y 7 y prototipos semejantes con mayor número de k, se pueden usar las aproximaciones mostradas en los prototipos de consulta del 1 al 5. Algunos de ellos pueden ser:
  a) Limitar el orden de las relaciones en el quasi-clique ya sea por una búsqueda específica de un patrón biológico o por repetición de tipo de relaciones en las relaciones del quasi-clique.
  b) Filtrar por proteínas o genes que estén en los nodos componentes del quasi-clique ya sea en los ID, PREFERED_NAME o ANNOTATIONS.
  c) Filtrar por un valor de alguna de las métricas SNA que posee cada nodo del multígrafo.
  d) Filtrar por el número de repeticiones de las relaciones de un clique.
  e) Filtrar por alguno de las métricas conjuntas de los cliques (antes descritos).
  f) Analizar cada quasi-clique de forma individual o por subconjuntos pequeños de 2 o 3 de ellos.
  g) Buscar estructuras con delimitaciones en el tipo de relaciones que las componen (Ver prototipos 6 y 7).
- Un efecto adicional de los k-quasi-cliques con k mayor a 3, es que en los resultados se encontrarán k-1 quasi-cliques que heredan algunas restricciones y filtros adicionados en la consulta para hallar el quasi-clique original.
- A medida que el k de los quasi-cliques aumenta, la complejidad de adquisición a nivel computacional, aumenta; contrariamente la facilidad de su análisis disminuye, por lo que resulta más práctico la búsqueda y análisis de quasi-cliques con k igual a 3 o 4.
- Al plantear las consultas para hallar los quasi-cliques – dependiendo del lenguaje – existen consideraciones a tener en cuenta:
    - **Cypher**: En éste lenguaje de tipo declarativo, a nivel global se plantean

relaciones entre nodos consecutivos, por lo que para un k-clique específico, se debe obtener un camino continuo sobre las relaciones que se quieren encontrar entre los nodos siguiendo un único camino de principio a fin y que no repita una relación; esto último en algunos casos puede solucionarse usando múltiples clausulas OPTIONAL MATCH.

  - **Gremlin**: Al realizar filtros de atributos que están en la relación, TITAN no se comporta bien, ya que está optimizado para consultas centradas en nodos, esto sumado a que la forma para verificar la eficiencia del query no es tan sencilla y natural como en neo4j.

- En las dos bases de datos usadas, no se encontraron formas de realizar índices mixtos entre atributos de nodos y relaciones, aparentemente debido a la arquitectura propuesta por las bases de datos de grafos.
- Para las consultas 6 y 7 donde se planteaban patrones extensos con quasi-clique de k=4 y k=5 se vio una diferencia en el comportamiento de ambas bases de datos. Así es como para el caso de Titan DB se debió realizar un tipo de consulta distinto al que se venía aplicando en los patrones del 1 al 5, y se debió usar la cláusula MATCH optimizada para el tipo de consulta planteado. A pesar de haber planteado índices en *ElasticSearch*, las respuestas para las consultas mencionadas, incluso con la cláusula optimizada fueron finalizadas en tiempos muy superiores a los obtenidos en las demás consultas. Cabe anotar que Neo4j sin ningún tipo de adición de índices respondió para todas las consultas en tiempos menores a 10 segundos para todas las consultas.
- Al diseñar las consultas para encontrar los quasi-cliques, se pudo notar que gremlin es un lenguaje muy amplio, compuesto de muchos patrones orientados a consultas traversas propias de teoría de grafos tradicional, a diferencia de cypher que no posee tantas posibilidades expresivas.

CONCLUSIONES

1. El uso de multigrafos mixtos permite obtener un modelamiento más natural y realista de los comportamientos biológicos que ocurren en organismos como *M. tuberculosis*.

2. El uso de bases de datos de grafos no relacionales para modelar multigrafos mixtos, permite que para un investigador en biología, tanto las consultas como la manipulación de los datos sea más fácil y óptima.

3. El modelo de proceso propuesto para el enriquecimiento de datos para la construcción de multigrafos (Ver **Ilustración *63***) se presenta como una alternativa viable para la manipulación de datos biológicos en bases de datos orientadas a grafos y permite hallar patrones de funciones regulatorias de manera exitosa.

4. Las consideraciones sobre patrones de consultas de quasi-cliques planteadas en la sección 4.3 del presente trabajo, representan una base sólida para la construcción de consultas y filtros que permiten hallar patrones que mapean a funciones regulatorias a nivel biológico.

5. El uso de $\gamma - quasi - cliques$ con valores de $\gamma = 1$ (clique fuerte) y $0 < \gamma < 1$ (clique débil) en consultas basadas en Cypher o Gremlin permiten hallar de forma flexible patrones que corresponden a mecanismos de regulación biológicos de interés.

TRABAJO FUTURO

1. Desarrollo de análisis biológico de los resultados correspondientes a los prototipos de consultas 1-5 y 7, con el fin de confirmar funciones regulatorias ya descubiertas o plantear hipótesis de funciones regulatorias conjuntas nuevas.

2. Optimización de patrones de consultas planteados en el trabajo, aplicando índices adicionales especializados en motores de búsqueda de soporte a bases de datos de grafos no relacionales basados en *Apache Lucene*.

3. Desarrollar e implementar algoritmos de detección de quasi-cliques optimizados nativamente para bases de datos de grafos no relacionales.

4. Desarrollar e implementar un algoritmo de detección de comunidades basado en detección de quasi-cliques optimizado nativamente para bases de datos de grafos no relacionales y aplicados a predicción de conjuntos regulatorios en organismos biológicos.

5. Desarrollar e implementar un algoritmo para la detección de *k-cores* o k-núcleos de cliques que pueden resultar representativos biológicamente.

6. Agregar conjuntos de datos adicionales al multigrafo, siguiendo el modelo de proceso de enriquecimiento de datos planteado.

7. Agregar más grafos con datos y atributos de otro tipo al multigrafo siguiendo la metodología planteada.

8. Dado que existe poca literatura sobre multigrafos mixtos, se podría expandir el entendimiento de los mismos aportando medidas de centralidad y búsqueda de comunidades que tengan en cuenta pesos diferenciados de acuerdo a "el sector" del multigrafo donde se evalúe la arista y a su dirección, es decir, pesos diferenciados por grafo en el multigrafo.

9. Agregar atributos a nodos y relaciones, provenientes de la base de datos Bio4j, tal como se plantea en el apartado 5.6.1 Construcción del multigrafo y verificar nuevas posibilidades de consultas sobre el multigrafo.

BIBLIOGRAFIA

[1] B. H.Junker y F. Schreiber, Analysis of biological networks, New Jersey: John Wiley & Sons, Inc., 2008.

[2] K. Han, G. Cui y Y. Chen, «Identifying Functional Groups by Finding Cliques and Near-Cliques in Protein Interaction Networks,» *IEEE Frontiers in the Convergence of Bioscience and Information Technologies,* pp. 159-168, 2007.

[3] J. Wang, Z. Cai y M. Li, «An Improved Method Based on Maximal Clique for Predicting Interactions in Protein Interaction Networks,» *IEEE International Conference on BioMedical Engineering and Informatics,* pp. 62-66, 2008.

[4] M. Bhattacharyya y S. Bandyopadhyay, «Mining the Largest Quasi-clique in Human Protein Interactome,» *IEEE International Conference on Adaptive and Intelligent Systems,* pp. 194-199, 2009.

[5] World Health Organization, «Global Tuberculosis Report 2014,» Switzerland, 2014.

[6] D. G. Russell, C. E. Barry 3rd y J. L. Flynn, «Tuberculosis: What We Don't Know Can, and Does, Hurt Us,» *Science,* nº 328, pp. 852-856, May 2010.

[7] T. Cui, L. Zhang, X. Wang y Z.-G. He, «Uncovering new signaling proteins and potential drug targets through the interactome analysis of Mycobacterium tuberculosis,» *BMC Genomics,* vol. 10, nº 1, p. 118, March 2009.

[8] J. Zeng, T. Cui y Z.-G. He, «A Genome-Wide Regulator−DNA Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv,» *Journal of proteome,* vol. 10, pp. 4682-4692, 2012.

[9] Y. Wang, T. Cui, C. Zhang, M. Yang, Y. Huang, W. Li, L. Zhang, C. Gao, Y. He, Y. Li, F. Huang, J. Zeng, C. Huang, Q. Yang, Y. Tian, C. Zhao, H. Chen, H. Zhang y Z.-G. He, «Global Protein-Protein Interaction Network in the Human Pathogen Mycobacterium tuberculosis H37Rv,» *Journal of Proteome Research,* vol. 9, p. 6665–6677, 2010.

[10] A. Koul, L. Vranckx, N. Dhar, H. W.H. Göhlmann, E. Özdemir, J.-M. Neefs, M. Schulz, P. Lu, E. Mørtz, J. D. McKinney, K. Andries y D. Bald, «Delayed bactericidal response of Mycobacterium tuberculosis to bedaquiline involves remodelling of bacterial metabolism,» *Nature Communitations,* vol. 5, p. 3369, Febrary 2014.

[11] C. Deb, C.-M. Lee, V. S. Dubey, J. Daniel, B. Abomoelak, T. D. Sirakova, S. Pawar, L. Rogers y P. E. Kolattukudy, «A Novel In Vitro Multiple-Stress Dormancy Model for Mycobacterium tuberculosis Generates a Lipid-Loaded, Drug-Tolerant, Dormant Pathogen,» *PLoS ONE,* vol. 4, nº 6, June 2009.

[12] J. M. Lew, A. Kapopoulou, L. M. Jones y S. T. Cole, «TubercuList - 10 years after,» *Tuberculosis (Edinb),* vol. 1, nº 91, pp. 1-7, September 2010.

[13] J. E. Galagan, P. Sisk, C. Stolte, B. Weiner, M. Koehrsen, F. Wymore, T. Reddy, J. D. Zucker, R. Engels, M. Gellesch, J. Hubble, H. Jin, L. Larson, M. Mao, M. Nitzberg, J. White, Z. K. Zachariah, G. Sherlock, C. A. Ball y G. K. Schoolnik, «TB database 2010: Overview and update,» *Tuberculosis (Edinb),* vol. 90, nº 1, pp. 225-235, March 2010.

[14] S. Choi, Introduction to Systems biology, New Jersey: Humana Press Inc., 2007, p. 549.

[15] R. Diestel, Graph Theory, Electronic Edition 2000 ed., New York: Springer-Verlag, 2000, p. 322.

[16] R. J.Wilson, Introduction to Graph Theory, Fourth Edition ed., Harlow: Prentice Hall, 1998, p. 180.

[17] R. Zafarani, M. Ali Abbasi y H. Liu, Social Media Mining - An Introduction, Arizona:

Cambridge University Press, 2014.

[18] F. Harary, Graph Theory, Addison-Wesley, 1969, p. 274.

[19] J. Scott, Social Network Analysis: A Handbook, Second Edition ed., London: SAGE Publications, 2000, p. 208.

[20] R. D. Alba, «A graph-theoretic definition of a sociometric clique,» *Journal of Mathematical Sociology,* vol. 3, pp. 113-126, 1973.

[21] B. J.A. y U. Murty, Graph Theory with applications, Quinta impresion ed., Ontario: North-Holland, 1982, p. 270.

[22] X. Lei, X. Shang, M. Wang y J. Diao, «Mining Frequent Correlated-Quasi-Cliques From PPI Networks,» *IEEE: 2010 WASE International Conference on Information Engineering,* vol. 2, pp. 7-10, August 2010.

[23] A. Ibna Matin, S. Jahan y M. Rezwanul Huq, «Community Recommendation in Social Network Using Strong Friends and Quasi-Clique Approach,» *8th International Conference on Electrical and Computer Engineering,* pp. 453-456, December 2014.

[24] Y. Zhang, J. Wang, Z. Zeng y L. Zhou, «Parallel Mining of Closed Quasi-Cliques,» *Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on,* pp. 1-10, April 2008.

[25] J. Celko's, Complete Guide to NoSQL, Waltham: Elsevier Inc, 2014.

[26] R. Van Bruggen, Learning Neo4j, First ed., Birmingham: August 2014, 2014.

[27] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall y W. Vogels, «Dynamo: Amazon's Highly Available Key-value Store,» *SOSP '07 Proceedings of twenty-first ACM SIGOPS symposium on Operating systems principles,* vol. 41, nº 6, pp. 205-220, 2007.

[28] P. J. Sadalage y M. Fowler, NoSQL Distilled : A Brief Guide to the Emerging World of Polyglot Persistence, Indiana: Addison-Wesley, 2012.

[29] W. Mcknight, Information Management, Waltham: Elsevier Inc, 2014.

[30] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes y R. E. Gruber, «Bigtable: A Distributed Storage System,» *ACM Transactions on Computer Systems,* vol. 26, nº 2, p. 26, June 2008.

[31] J. Dean y S. Ghemawat, «MapReduce: Simplified Data Processing on Large Clusters,» *OSDI'04: Sixth Symposium on Operating System Design and Implementation,* p. 13, December 2004.

[32] D. Feinberg, A. Merv, N. Heudecker, A. M. Ronthal y T. Palanca, «Magic Quadrant for Operational Database,» United Stade of America , 2015.

[33] Solid IT, December 2015. [En línea]. Available: http://db-engines.com/en/ranking.

[34] N. Yuhanna, L. Owens y E. Cullen, «Market Overview: Graph Databases,» Cambridge, 2015.

[35] GraphX - Apache spark, December 2015. [En línea]. Available: http://spark.apache.org/graphx/.

[36] G. Malewicz, M. H. Austern, A. J. C. Bik, J. C. Dehnert, I. Horn, N. Leiser y G. Czajkowski, «Pregel: A System for Large-Scale Graph Processing,» *SIGMOD '10 Proceedings of the 2010 ACM SIGMOD International Conference on Management of data,* pp. 135-146 , 2010.

[37] M. A. Rodriguez y P. Neubauer, Graph Data Management: Techniques and Applications, chapter The Graph Traversal Pattern, Pardede, 2011.

[38] I. Robinson, J. Webber y E. Eifrem, Graph databases, Second Edition ed., Gravenstein Highway North: O'Reilly Media, Inc., 2015.

[39] V. Mishra, Beginning Apache Cassandra Development, 1 ed., Apress, 2014.

[40] S. Jouili y V. Vansteenberghe, «An empirical comparison of graph databases,» *Social*

Computing (SocialCom), 2013 International Conference. IEEE, pp. 708 - 715, Sept. 2013.

[41] Neo Technology, Inc., August 2015. [En línea]. Available: http://neo4j.com/.

[42] Apache TinkerPop, August 2015. [En línea]. Available: https://github.com/tinkerpop/gremlin/.

[43] N. Watt, J. Partner y A. Vukotic, Neo4j in Action, First Edition ed., Manning Publications Co., 2015.

[44] Thinkaurelius, December 2015. [En línea]. Available: http://thinkaurelius.com/.

[45] Aurelius - DataStax, August 2015. [En línea]. Available: http://thinkaurelius.github.io/titan/.

[46] ThinkAurelius, December 2015. [En línea]. Available: http://tinkerpop.apache.org/.

[47] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen y D. Wilkins, «A Comparison of a Graph Database and a Relational Database. A Data Provenance Perspective,» *ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference,* p. 42, 2010.

[48] F. Holzschuher y R. Peinl, «Performance of Graph Query Languages. Comparison of Cypher, Gremlin and Native Access in Neo4j,» *ACM EDBT '13 Proceedings of the Joint EDBT/ICDT 2013 Workshops,* pp. 195-204, March 2013.

[49] O. Panzarino, Learning Cypher, Mumbai: Packt Publishing, 2014.

[50] M. A. Rodriguez, «The Gremlin Graph Traversal Machine and Language,» *2015 Proceedings of the ACM Database Programming Languages Conference,* p. 10, August 2015.

[51] J. C. Palomino, S. Cardoso Leão y V. Ritacco, Tuberculosis 2007 - From basic science to patient care, First Edition ed., TuberculosisTextbook.com, 2007.

[52] M. J.Buck y J. D. Lieb, «ChIP-chip: considerations for the design, analysis, and application of genome-wide chromatin immunoprecipitation experiments,» *Genomics,* vol. 83, pp. 340-360, October 2003.

[53] M. Zheng, ,. L. O. Barrera, B. Ren y Y. Nian Wu, «ChIP-chip: Data, Model, and Analysis,» *Biometrics,* vol. 63, p. 787–796, September 2007.

[54] G. Altay, M. Asim, F. Markowetz y D. E. Neal, «Differential C3NET reveals disease networks of direct physical interactions,» *BMC Bioinformatics,* vol. 12, p. 296, 21 July 2011.

[55] Z. Kurt, N. Aydin y G. Altay, «A comprehensive comparison of association estimators for gene network inference algorithms,» *Bioinformatics,* vol. 30, nº 15, pp. 2142-2149, 07 2014.

[56] C. Olsen, P. E.Meyer y G. Bontempi, «On the Impact of Entropy Estimation on Transcriptional Regulatory Network Inference Based on Mutual Information,» *EURASIP Journal on Bioinformatics and Systems Biology,* vol. 2009, nº 1, 1 2009.

[57] E. B.S. y D. A. Skron, The Cambridge dictionary of statistics, Cambridge, UK: Cambridge University Press, 2010.

[58] R. TB, R. R, W. F, M. P, D. D, E. R, G. M, H. J, J. D, J. H, K. M, L. L, M. M, N. M, S. P, S. C, W. B, W. J, Z. ZK, S. G, G. JE, B. CA y S. GK, «TB database: an integrated platform for tuberculosis research,» *Nucleic Acids Research,* vol. 37, p. D499–D508, October 2009.

[59] P. Pareja Tobes, R. Tobes, M. Manrique, E. Pareja y E. Pareja Tobes, «Bio4j: a high-performance cloud-enabled graph-based data platform,» *bioRxiv,* p. 11, March 2015.

[60] SIB(Swiss Institute of Bioinformatics), CPR(NNF Center for Protein Research) y EMBL(European Molecular Biology Laboratory), September 2015. [En línea]. Available: http://string-db.org.

[61] UCLA (University of California, Los Angeles), September 2015. [En línea]. Available: http://dip.doe-mbi.ucla.edu.

[62] SIB Swiss Institute of Bioinformatics, ecole Polytechnique Federale de Lausanne, September 2015. [En línea]. Available: http://tuberculist.epfl.ch.

[63] National Institute of Health (NIH), European Molecular Biology Laboratory (EMBL), British Heart Foundation (BHF), Parkinson's Disease United Kingdom (PDUK),State Secretariat for

Education, Research and Innovation SERI, National Science Foundation (NSF), September 2015. [En línea]. Available: http://www.uniprot.org.

[64] K. J. Kiesera, C. Baranowskia, M. C. Chaob, J. E. Longc, C. M. Sassettic, M. K. Waldorb, J. C. Sacchettinif, T. R. Loergerg y E. J. Rubina, «Peptidoglycan synthesis in Mycobacterium tuberculosis is organized into networks with varying drug susceptibility,» *PNAS: Proceedings of the National Academy of Sciences of the United States of America.,* vol. 112, nº 42, p. 22, 2015.

[65] A. J. C. Steyn, D. M. Collins, M. K. Hondalus, W. R. Jacobs, R. P. Kawakami y B. R. Bloom, «Mycobacterium tuberculosis WhiB3 interacts with RpoV to affect host survival but is dispensable for in vivo growth,» *PNAS,* vol. 99, nº 5, p. 3147–3152, March 5, 2002.

[66] M. Kwasi G., C. V. Forst, K. M. Dobos, J. T. Belisle, J. Chen, E. M. Bradbury, A. R.M. Bradbury y X. Chen, «Mycobacterium tuberculosis functional network analysis by global subcellular protein profiling,» *Molecular Biology of the Cell,* vol. 16, p. 396–404, January 2005.

[67] J. E. Griffin, J. D. Gawronski, M. A. DeJesus, T. R. Ioerger, B. J. Akerley y C. M. Sassetti, «High-resolution phenotypic profiling defines genes essential for mycobacterial growth and cholesterol catabolism,» *PLoS Pathogens,* vol. 7, nº 9, September 2011.

[68] J. Dziadek, S. A. Rutherford, M. V. Madiraju, M. A. L. Atkinson y M. Rajagopalan, «Conditional expression of Mycobacterium smegmatis ftsZ, an essential cell division gene,» *Microbiology,* nº 149, p. 1593–1603, 2003.

[69] X. Mao, Q. Ma, C. Zhou, X. Chen, H. Zhang, J. Yang, F. Mao, W. Lai y Y. Xu, «DOOR 2.0: presenting operons and their functions through dynamic and integrated views,» *Nucleic Acids Research,* pp. 1-6, October 2013.

[70] F. Mao, P. Dam, J. Chou, V. Olman y Y. Xu, «DOOR: a database for prokaryotic operons,» *Nucleic Acids Research,* vol. 37, p. D459–D463, November 2009.

[71] P. Dam, V. Olman, K. Harris, Z. Su y Y. Xu, «Operon prediction using both genome-specific and general genomic information,» *Nucleic Acids Research,* vol. 35, nº 1, p. 288–298, December 2006.

[72] T. Das Gupta, B. Bandyopadhyay y S. K. Das Gupta, «Modulation of DNA-binding activity of Mycobacterium tuberculosis HspR by chaperones,» *Microbiology,* vol. 154, nº 2, p. 484–490, Febrary 2008.

[73] L. S. Busenlehner, M. A. Pennella y D. P. Giedroc, «The SmtB/ArsR family of metalloregulatory transcriptional repressors: structural insights into prokaryotic metal resistance,» *FEMS Microbiology Reviews,* vol. 27, pp. 131-143, May 2003.

[74] J. E. Galagan, K. Minch, M. Peterson, A. Lyubetskaya, E. Azizi, L. Sweet, A. Gomes y e. al, «The Mycobacterium tuberculosis regulatory network and hypoxia,» *Nature,* vol. 499, pp. 178-183, July 2013.

[75] E. G. Fernández, F. Javier Medrano, B. Galán y J. L. García, «Deciphering the Transcriptional Regulation of Cholesterol Catabolic Pathway in Mycobacteria. IDENTIFICATION OF THE INDUCER OF KstR REPRESSOR,» *THE JOURNAL OF BIOLOGICAL CHEMISTRY,* vol. 289, nº 25, p. 17576 –17588, June 2014.

ANEXOS

ANEXO 1: Scripts de importanción de multigrafo a Titan DB

A continuación, se detalla la inicialización general de la base de datos Titan DB y cada una de las importaciones de los diferentes archivos en la misma.

a) **Creación de atributos de nodo:** además de crear el tipo de nodo PARTICLE, se adicionan los atributos posibles de los nodos.

```
mgmt = graph.openManagement()
particle = mgmt.makeVertexLabel('particle').make();

mgmt.makePropertyKey("PREFERED_NAME").dataType(String.class).make();
mgmt.makePropertyKey("NCBI_TAXON_ID").dataType(String.class).make();
mgmt.makePropertyKey("TAXON_NAME").dataType(String.class).make();
mgmt.makePropertyKey("ANNOTATIONS").dataType(String.class).make();

mgmt.makePropertyKey("DEGREE_CENTRALITY").dataType(Double.class).make();
mgmt.makePropertyKey("CLOSENESS_CENTRALITY").dataType(Double.class).make();
mgmt.makePropertyKey("BETWEENNESS_CENTRALITY").dataType(Double.class).make();

mgmt.commit()
```

b) **Creación de atributos de relaciones:** Se crean los tipos de relaciones y se especifican los atributos de las mismas.

```
mgmt = graph.openManagement()
mgmt.makeEdgeLabel('INTERACTION_PROTEINES_1').multiplicity(MULTI).make()
mgmt.makeEdgeLabel('INTERACTION_REGULATION_1').multiplicity(MULTI).make()
mgmt.makeEdgeLabel('INTERACTION_PROTEINES_2').multiplicity(MULTI).make()
mgmt.makeEdgeLabel('INTERACTION_CO_EXPRESSION_1').multiplicity(MULTI).make()
mgmt.makeEdgeLabel('INTERACTION_CO_EXPRESSION_2').multiplicity(MULTI).make()
mgmt.getRelationTypes(EdgeLabel.class)

mgmt.makePropertyKey("INTERACTION_TYPE").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("PARTICLE_A").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("PARTICLE_B").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("DIP_INTERACTION_ID").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("DETECTION_METHODS").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("PUBLICATIONS").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("FOUND").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("COUNTER_REPITED").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("STD_COUNTER_REPITED").dataType(String.class).cardinality(Cardinality.SINGLE).make();

mgmt.makePropertyKey("TARGET_PROMOTER").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("PROMOTES_COORDINATES_GENOMA").dataType(String.class).cardinality(Cardinality.SINGLE).m
ake();
mgmt.makePropertyKey("GENE_IN_OPERON").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("PROMOTER_SEQUENCE").dataType(String.class).cardinality(Cardinality.SINGLE).make();

mgmt.makePropertyKey("GENE_INTERACTOR_A").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("GENE_INTERACTOR_B").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("ANNOTATION_INTERACTOR_A").dataType(String.class).cardinality(Cardinality.SINGLE).make();
mgmt.makePropertyKey("ANNOTATION_INTERACTOR_B").dataType(String.class).cardinality(Cardinality.SINGLE).make();

mgmt.commit()
```

c) **Creación de índices:** se crean dos índices externos en ElasticSearch para el ID de las PARTICLES y para el tipo de interacción.

```
mgmt = graph.openManagement()
IDProperty =
mgmt.makePropertyKey("ID_").dataType(String.class).cardinality(Cardinality.SET).make()
mgmt.buildIndex("search", Vertex.class).addKey(IDProperty,
Mapping.STRING.asParameter()).buildMixedIndex("search")

INTERACTION_TYPE_Property = mgmt.getPropertyKey('INTERACTION_TYPE')
mgmt.buildIndex("search_2", Edge.class).addKey(INTERACTION_TYPE_Property,
Mapping.STRING.asParameter()).buildMixedIndex("search")
mgmt.commit()
```

d) **Importación de nodos individuales con atributos básicos.** Dado que no se tiene ningún nodo cargado en la base, se adiciona uno por cada línea del archivo mediante el comando *addVertex.* El nodo adicionado es correspondiente a un nodo de tipo PARTICLE en el modelo propuesto. Al final del script se realiza un *commit* que corresponde a un modelo de transacciones manual definido en las dos primeras líneas de código, el cual permite que dichos *commit* se realicen al final de una transacción y para casos de importación – como el presentado – la eficiencia de guardado sea más alta.

```
graph.tx().onReadWrite(Transaction.READ_WRITE_BEHAVIOR.MANUAL)
graph.tx().open()

file = new File("[Ruta del archivo correspondiente]");
line_number=0
file.readLines().each({ line ->
        println line_number;
        line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");
        size = line_list.size();
        line_number++;
        def ID = line_list[0];
        def PREFERED_NAME = line_list[1];
        def NCBI_TAXON_ID = line_list[2];
        def TAXON_NAME = line_list[3];
        def ANNOTATIONS = line_list[4];

        particle = graph.addVertex('particle');
        particle.property("ID_", ID);
        particle.property("PREFERED_NAME", PREFERED_NAME);
        particle.property("NCBI_TAXON_ID", NCBI_TAXON_ID);
        particle.property("TAXON_NAME", TAXON_NAME);
        particle.property("ANNOTATIONS", ANNOTATIONS);
})

graph.tx().commit()
```

e) **Importación de atributos SNA del multígrafo para cada nodo**. Por cada línea se busca un nodo y se agregan tres atributos SNA actualizándolo; éste proceso se hace uno a uno y está optimizado para grandes conjuntos de datos mediante un índice ya mencionado por ID, y el uso de la estrategia de *commit* ya mencionada en el punto anterior.

```
graph.tx().open();
file_sna = new File("[Ruta del archivo correspondiente]");
notExist = 0;
counterLine = 0;
file_sna.readLines().each({ line ->
	line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

	def ID1 = line_list[0];
	def ROW1 = line_list[1];
	def ROW2 = line_list[2];
	def ROW3 = line_list[3];
	def DEGREE_CENTRALITY = line_list[4];
	def ROW5 = line_list[5];
	def ROW6 = line_list[6];
	def ROW7 = line_list[7];
	def ROW8 = line_list[8];
	def CLOSENESS_CENTRALITY = line_list[9];
	def BETWEENNESS_CENTRALITY = line_list[10];

	if(counterLine > 0){
		counter_protein_first = g.V().has('ID_',ID1);
		if(counter_protein_first == null || g.V().has('ID_',ID1).count()==0){
			notExist++;
			System.out.println("Particula no existente: " + ID1)
		}else{
			protein_first=g.V().has('ID_',ID1).next();

			protein_first.property("DEGREE_CENTRALITY",DEGREE_CENTRALITY as Double)
			protein_first.property("CLOSENESS_CENTRALITY",CLOSENESS_CENTRALITY as Double)
			protein_first.property("BETWEENNESS_CENTRALITY",BETWEENNESS_CENTRALITY as
Double)

			if(counterLine%100==0){
				System.out.println("counterLine: " + counterLine);
			};
		};

		counterLine++;
	}else{
		counterLine++
	};

})

System.out.println("Particulas no existentes: " + notExist);
System.out.println("lineas analizadas: " + counterLine);
```

f) **Importación de subgrafo número 1**.

```
graph.tx().open();
file_1 = new File("[Ruta del archivo correspondiente]");
line_number=0
file_1.readLines().each({ line ->
		println line_number;
		line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

		def ID1 = line_list[0];
		def ID2 = line_list[1];
		def DIP_INTERACTION_ID = line_list[2];
		def DETECTION_METHODS = line_list[3];
		def PUBLICATIONS = line_list[4];
		def COUNTER_REPITED = line_list[5];
		def STD_COUNTER_REPITED = line_list[6];

		v1 = g.V().has('ID_',ID1).next();
		v2 = g.V().has('ID_',ID2).next();

		edgeObject = v1.addEdge('INTERACTION_PROTEINES_1',v2);
		edgeObject.property("INTERACTION_TYPE", "PROTEIN-PROTEIN");
		edgeObject.property("PARTICLE_A", "PROTEIN");
		edgeObject.property("PARTICLE_B", "PROTEIN");
		edgeObject.property("DIP_INTERACTION_ID", DIP_INTERACTION_ID);

		edgeObject.property("DETECTION_METHODS", DETECTION_METHODS);
		edgeObject.property("PUBLICATIONS", PUBLICATIONS);
		edgeObject.property("FOUND", "DATASET_1");
		edgeObject.property("COUNTER_REPITED", COUNTER_REPITED);
		edgeObject.property("STD_COUNTER_REPITED", STD_COUNTER_REPITED);

		line_number++;
})

graph.tx().commit();
```

g) **Importación de subgrafo número 2.**

```
graph.tx().open();
file_2 = new File("[Ruta del archivo correspondiente]");
line_number=0
file_2.readLines().each({ line ->
		println line_number;
		line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

		def ID1 = line_list[0];
		def ID2 = line_list[1];
		def ID3 = line_list[2];
		def TARGET_PROMOTER = line_list[3];
		def PROMOTES_COORDINATES_GENOMA = line_list[4];
		def GENE_IN_OPERON = line_list[5];
		def PROMOTER_SEQUENCE = line_list[6];
		def COUNTER_REPITED = line_list[7];
		def STD_COUNTER_REPITED = line_list[8];

		v1 = g.V().has('ID_',ID1).next();
		v2 = g.V().has('ID_',ID2).next();

		edgeObject = v1.addEdge('INTERACTION_REGULATION_1',v2);
		edgeObject.property("INTERACTION_TYPE", "PROTEIN-GEN");
		edgeObject.property("PARTICLE_A", "PROTEIN");
		edgeObject.property("PARTICLE_B", "GEN");

		edgeObject.property("TARGET_PROMOTER", TARGET_PROMOTER);
		edgeObject.property("PROMOTES_COORDINATES_GENOMA", PROMOTES_COORDINATES_GENOMA);
		edgeObject.property("GENE_IN_OPERON", GENE_IN_OPERON);
		edgeObject.property("PROMOTER_SEQUENCE", PROMOTER_SEQUENCE);

		edgeObject.property("FOUND", "DATASET_2");
		edgeObject.property("COUNTER_REPITED", COUNTER_REPITED);
		edgeObject.property("STD_COUNTER_REPITED", STD_COUNTER_REPITED);

		line_number++;
})

graph.tx().commit();
```

h) Importación de subgrafo número 3.

```
graph.tx().open();
file_3 = new File("("[Ruta del archivo correspondiente]");
line_number=0
file_3.readLines().each({ line ->
		println line_number;
		line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

		def ID1 = line_list[0];
		def ID2 = line_list[1];

		def GENE_INTERACTOR_A = line_list[2];
		def GENE_INTERACTOR_B = line_list[3];
		def ANNOTATION_INTERACTOR_A = line_list[4];
		def ANNOTATION_INTERACTOR_B = line_list[5];

		def COUNTER_REPITED = line_list[6];
		def STD_COUNTER_REPITED = line_list[7];

		v1 = g.V().has('ID_',ID1).next();
		v2 = g.V().has('ID_',ID2).next();

		edgeObject = v1.addEdge('INTERACTION_PROTEINES_2',v2);
		edgeObject.property("INTERACTION_TYPE", "PROTEIN-PROTEIN");
		edgeObject.property("PARTICLE_A", "PROTEIN");
		edgeObject.property("PARTICLE_B", "PROTEIN");

		edgeObject.property("GENE_INTERACTOR_A", GENE_INTERACTOR_A);
		edgeObject.property("GENE_INTERACTOR_B", GENE_INTERACTOR_B);
		edgeObject.property("ANNOTATION_INTERACTOR_A", ANNOTATION_INTERACTOR_A);
		edgeObject.property("ANNOTATION_INTERACTOR_B", ANNOTATION_INTERACTOR_B);

		edgeObject.property("FOUND", "DATASET_3");
		edgeObject.property("COUNTER_REPITED", COUNTER_REPITED);
		edgeObject.property("STD_COUNTER_REPITED", STD_COUNTER_REPITED);

		line_number++;
})

graph.tx().commit();
```

i) Importación de subgrafo número 4.

```
graph.tx().open();
file_4 = new File("[Ruta del archivo correspondiente]");
line_number=0
file_4.readLines().each({ line ->
		println line_number;
		line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

		def ID1 = line_list[0];
		def ID2 = line_list[1];

		def OTHER = line_list[2];

		def COUNTER_REPITED = line_list[3];
		def STD_COUNTER_REPITED = line_list[4];

		v1 = g.V().has('ID_',ID1).next();
		v2 = g.V().has('ID_',ID2).next();

		edgeObject = v1.addEdge('INTERACTION_CO_EXPRESSION_1',v2);
		edgeObject.property("INTERACTION_TYPE", "GEN-GEN_COEXPRESSION");
		edgeObject.property("PARTICLE_A", "GEN");
		edgeObject.property("PARTICLE_B", "GEN");

		edgeObject.property("FOUND", "DATASET_4");
		edgeObject.property("COUNTER_REPITED", COUNTER_REPITED);
		edgeObject.property("STD_COUNTER_REPITED", STD_COUNTER_REPITED);

		line_number++;
})

graph.tx().commit();
```

j) Importación de subgrafo número 5.

```
graph.tx().open();
file_5 = new File("[Ruta del archivo correspondiente]");
line_number=0
file_5.readLines().each({ line ->
        println line_number;
        line_list=line.split(",(?=(?:[^\"]*\"[^\"]*\")*(?![^\"]*\"))");

        def ID1 = line_list[0];
        def ID2 = line_list[1];

        def OTHER = line_list[2];

        def COUNTER_REPITED = line_list[3];
        def STD_COUNTER_REPITED = line_list[4];

        v1 = g.V().has('ID_',ID1).next();
        v2 = g.V().has('ID_',ID2).next();

        edgeObject = v1.addEdge('INTERACTION_CO_EXPRESSION_2',v2);
        edgeObject.property("INTERACTION_TYPE", "GEN-GEN_COEXPRESSION");
        edgeObject.property("PARTICLE_A", "GEN");
        edgeObject.property("PARTICLE_B", "GEN");

        edgeObject.property("FOUND", "DATASET_5");
        edgeObject.property("COUNTER_REPITED", COUNTER_REPITED);
        edgeObject.property("STD_COUNTER_REPITED", STD_COUNTER_REPITED);

        line_number++;
})

graph.tx().commit();
```