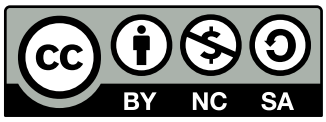
Aalto University
School of Science

Master's Programme in Mathematics and Operations Research

Identification of vital vertices within protein-protein interaction networks

Joona Lindell

Master's Thesis
2026



Aalto University
School of Science

Author Joona Lindell

Title Identification of vital vertices within protein-protein interaction networks

Degree programme Mathematics and Operations Research

Major Operations Research

Supervisor Asst. Prof. Philine Schiewe

Advisor Dr. Fernando Dias

Date 6 March 2026 **Number of pages** 61+5 **Language** English

Abstract

In this thesis we consider the biological problem of identifying what are the most important, or vital, proteins within protein interaction networks. We consider the notable signs of importance to be functional modules of proteins corresponding to some biological processes and being part of a minimum set which is required for all of the predicted protein interactions to exist. This connects the biological problem to graph theory and allows us to consider the maximum clique problem and the minimum vertex cover problem.

Both of the aforementioned problems are NP-complete combinatorial optimisation problems which have no known polynomial time algorithms for exact solutions. Due to this property and large scale of the studied networks, solving the problems exactly becomes infeasible quickly considering both the optimisation runtime and memory requirements. Attempting to find the most important proteins can be done through a centrality measurement which quantifies how important a vertex is within a network with respect to some metric. Most graph theoretical approaches on identifying vital proteins from protein-protein interaction networks have utilised either NP-complete problems or centrality methods separately. As a more novel part of the research, these centrality values are used to construct heuristic algorithms for the clique and vertex cover problems. Additionally, methods which simplify the process of finding exact solutions to the NP-complete problems in the context of the interaction networks are considered.

The methods are applied to the proteomes of parasitic species under genus *Leishmania* spreading a disease called leishmaniasis which is considered to be a major global health problem. These proteomes are analysed to find significant matches among proteins which predict the protein-protein interactions. Metrics on the alignment indicate that a majority of the found interactions are significant which are used as the basis for formulating the interaction networks. The methods applied produced good results in improving integer linear programming procedures and finding heuristic solutions on the interaction networks. The only notable exception of the network of *Leishmania donovani* proving to be relatively difficult to analyse with the considered tools due to the scale and underlying structures in the network.

Keywords Protein-protein interaction network, Leishmania, NP-complete, optimisation, clique, vertex cover, centrality, heuristic

Aalto University
School of Science

Tekijä Joona Lindell

Työn nimi Tärkeiden solmujen tunnistaminen proteiinien vuorovaikutusverkoista

Koulutusohjelma Mathematics and operations research

Pääaine Operations research

Työn valvoja Apulaisprofessori Philine Schiewe

Työn ohjaaja Tohtori Fernando Dias

Päivämäärä 6.3.2026 **Sivumäärä** 61+5 **Kieli** englanti

Tiivistelmä

Tässä diplomityössä tutkitaan tärkeimpien tai elintärkeiden proteiinien tunnistamista proteiinien vuorovaikutusverkoista. Tutkimusen kohteena ovat biologisia prosesseja ylläpitävät proteiinikompleksit sekä minimaaliset proteiinijoukot, jonka proteiinit vaaditaan proteiiniverkon vuorovaikutusten ylläpitämiseksi. Nämä biologiset ongelmat voidaan tulkita verkkoteorian avulla yhdistämällä proteiinien vuorovaikutusverkot klikkiongelmaan ja solmupeiteongelmaan.

Molemmat edellä mainitut ongelmat ovat NP-täydellisiä kombinatorisia optimointiongelmia, joille ei tunneta polynomisessa ajassa toimivia ratkaisualgoritmeja. Laskennan vaikeuden ja verkkojen suuren koon takia tarkkojen ratkaisujen muodostaminen on mahdotonta suoritusajan ja käyttömuistin vaatimusten takia. Tärkeitä proteiineja voidaan myös tunnistaa keskeisyyden avulla, joka mittaa solmun tärkeyttä verkossa. Tärkeitä proteiineja on aikaisemmassa tutkimuksessa tutkittu verkkoteorian menetelmillä, mutta menetelmät ovat usein hyödyntäneet NP-täydellisiä ongelmia tai keskeisyyttä erikseen. Tämän diplomityön tutkimus pyrkii yhdistämään lähestymistavat hyödyntämällä keskeisyyttä klikkiongelman ja solmupeiteongelman ratkaisevien heuristiikoiden pohjana. Tämän lisäksi diplomityössä tutkitaan menetelmiä, jotka yksinkertaistavat NP-täydellisten ongelmien tarkkojen ratkaisujen muodostamista vuorovaikutusverkkojen kontekstissa.

Menetelmillä tutkitaan merkittävänä maailmanlaajuisena terveysongelmana pidettyä leishmanioosia levittävien *Leishmania*-parasiittien proteiinikantoja. Parasiittien proteiinien sekvensseja verrataan toisiinsa paikallisen kohdistamisen avulla, jolla voidaan löytää proteiinisekvensseistä samanlaisuuksia. Samanlaisuudet ennustavat proteiinien välisiä vuorovaikutuksia, joita käytetään vuorovaikutusverkkojen luomiseen. Kohdistuksen tuloksena saatiin suuri joukko merkittävänä pidettäviä proteiinipareja, joiden perusteella tutkittavat verkot muodostettiin. Menetelmät tuottivat hyviä tuloksia lineaaristen optimointiongelmien menettelyjen parantamisessa sekä heuristisen ratkaisujen löytämisessä. Merkittävä poikkeus oli lajin *Leishmania donovani* proteiinien vuorovaikutuksia kuvaava verkko, jossa edellä käsiteltyjen ongelmien ratkaisu oli huomattavasti vaikeampaa. Haasteiden syynä pidettiin verkon suurta kokoa ja rakenteita.

Avainsanat Proteiinien vuorovaikutusverkko, Leishmania, NP-täydellinen, optimointi, klikki, solmupeite, keskisyys, heuristiikka

Preface

I want to thank my supervisor Professor Philine Schiewe and my advisor Dr. Fernando Dias for their support and guidance throughout the process of working on the corresponding research and writing the thesis. Working on this topic has definitely not been easy by any means, but it has fully rekindled my spirit to research altogether and I truly felt like I could have a place within the field of research.

I also want to thank my immediate family along with Guild of Physics and its' boards 2023 and 2024 for their support throughout the years. You really made my time studying what it ended up being: not only studying. I know there is an alternative universe where I would be another lost name after Kiljava who could be spotted on the campus only during exams. Working towards fun things and learning about culture and life was a lovely highlight of my studies that I will never forget.

Finally, I want to thank Atte, Dylan, Henri N., Ilmari, Jaakko and especially Joonas for allowing me to grow as a person to heights I would have never known were possible. You made these past years the best and most important time of my life so far and I cannot put it into words how much you all have meant to me. *"Present company, the best that you can find"*, thank you so much for being around for me and supporting me in the ebb and flow of life.

Espoo, March 6, 2026

Joona Matias Aleksanteri Lindell

Contents

Symbols and abbreviations

Symbols and operators

G	An undirected and unweighted simple graph with the set of vertices V and edges E. Shorthand for $G(V, E)$.
$V(G)$	Vertex set of graph G
$E(G)$	Edge set of graph G
$A(G)$	Adjacency matrix of graph G.
$l_G(v)$	Adjacency list of vertex v in graph G.
$\rho(G)$	Density of graph G.
$P_G(k)$	Degree distribution of graph G.
$\mathbb{Z}_{\geq 0}$	Set of nonnegative integers.
$\deg(v)$	Degree of vertex v.
$\Delta(G)$	Maximum vertex degree of a graph G.
$\bar{d}(G)$	Average vertex degree in graph G.
$S_1 \backslash S_2$	Set of all elements in S_1 which are not in S_2.
$G[S]$	Subgraph of G induced by $S \subset V$.
$\mathrm{d}(u, v)$	Geodesic distance between vertices u and v in a graph.
σ_{st}	Amount of unique shortest paths between distinct vertices s and t.
$\sigma_{st}(v)$	Amount of unique shortest paths between distinct vertices s and t which are passing through v.
C	Connected component of a graph.
$\mathbf{e}_n$	n-dimensional vector of ones.

Abbreviations

ILP	integer linear programming
MISP	maximum independent set problem
MCP	maximum clique problem
MVCP	minimum vertex cover problem
NTD	neglected tropical disease
L.	*Leishmania*
PPIN	protein-protein interaction network
s.t.	subject to

1 Introduction

Proteins do not appear in isolation in organisms but rather amongst other proteins. In addition, many biological mechanisms heavily rely on interactions between proteins. The networks built from these interactions essentially define life as it is known. Identifying vital proteins within these systems of interactions is a long standing problem with many biological applications as these proteins can be a key factor in new treatment methods [5, 47] and vaccines [55]. The work in this thesis focuses on different species of *Leishmania*, which are sub-tropical parasitic organisms that cause the disease called leishmaniasis [4]. The protein databases of these species are studied and interactions between proteins are predicted with basic local alignment search tools which are widely used within biological applications [3]. These search tools predict interactions from protein databases based on similarity of the proteins [53]. With the predicted connections, networks are formed where the proteins are used as vertices and predicted interactions are used as edges.

For the identification of vital proteins in these interaction networks, two well known NP-complete problems are considered: the maximum clique problem and the minimum vertex cover problem. Cliques can be used as a predictor for functional modules of proteins [59] and vertex covers offer a way to construct a set of proteins which are necessary for all protein interactions within the proteome of the organism [34]. These optimisation problems can be used to predict vital sets of proteins of the species. However, since these problems belong to the NP-complete class of problems [38], computational complexity of solving the problems is a concern. Finding exact solutions or even formulating these problems can be inefficient considering runtime or memory usage. This inefficiency has led to the consideration of a number of alternative computational methods to identify vital proteins. For example, centrality methods offer algorithms that quantify how important the vertices of a network are according to some metric [25].

Previous research on vital proteins has focused on the NP-complete problems [1, 20] and centrality methods [46, 56] separately. The more novel part of our research combines the methods where the centrality methods are utilised as a base for heuristic algorithms to generate solution candidates for the NP-complete problems. The goal of this thesis is to find a way to represent predicted interactions of proteins as networks and analyse them with the methods that attempt to identify important proteins from the interaction networks. Additionally, more standard ILP formulations are also formulated which attempt to solve the corresponding NP-complete problems exactly. The heuristic algorithms are compared to the ILP formulations in the quality of the result and the runtime. For the problem solution procedures, memory efficiency and ease of execution on a lower end computer are additionally prioritised as a proof of concept. Everything in this thesis is run on an Intel Core i5-10210U 1.6 GHz CPU with 4 GB of available RAM.

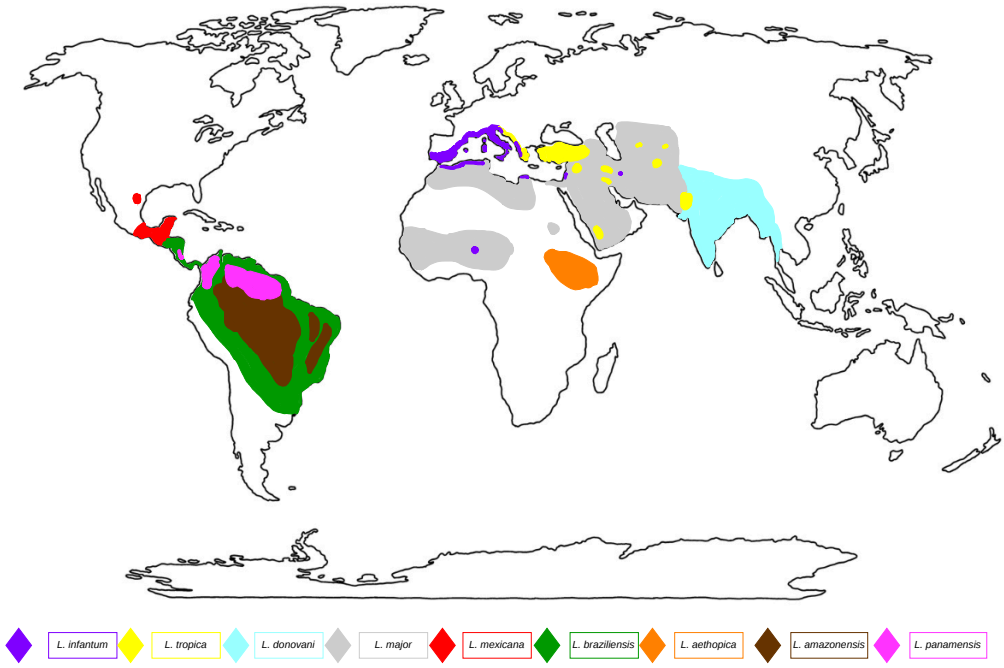


Figure 1: World distribution of certain *Leishmania* (*L.*) species. In this thesis, the species *infantum*, *tropica*, *donovani*, *major*, *mexicana*, *braziliensis* and *amazonensis* are considered for the protein-protein interaction analysis.

2 Background

Leishmaniasis is a disease that belongs to the class of neglected tropical diseases (NTD) classified by the World Health Organization (WHO). The disease is spread by parasites of the genus *Leishmania* (abbreviated as *L.*) that are distributed in sub-tropical and tropical areas of the world according to Figure 1. More than a billion people live in areas endemic to the diseases [67] and more than 1 million new cases of leishmaniasis are diagnosed each year [4, 66]. Currently, treatment and prevention of the disease is difficult since there are no effective and durable vaccines for humans [6], treatments have problems with long treatment times and toxicity, and the diseases have a significant risk of developing resistances over time [17].

In the fight against leishmaniasis, a significant part of the difficulty comes from difficulties of modelling protein interactions and reliably finding vital proteins of organisms. Interactions between proteins can be physically identified with *in vitro* and *in vivo* methods [54]. Physical experiments give more concrete results, but these experiments can also produce inaccurate results while being slow and expensive. This has led to the implementation of different computational methods for identification of protein interactions. These computational methods are constantly being developed, but they currently have problems with accuracy [41] and may require pairing with experimentally defined interactions to produce stronger results [47]. Regardless of these issues, these methods still offer necessary tools to perform research on interactions between proteins.

Protein-protein interaction networks (PPIN) have been used in the analysis of protein functions [63], diseases [64], vector species such as *Leishmania* parasites [27], and in the development of new drugs [5, 47] and vaccines [55]. They offer an effective way to represent proteomes of species while considering the biological processes between proteins. With appropriate analysis, these networks can be further considered to find parts of the proteome which construct functional modules [61] that are the building blocks of biological processes or vital sets of proteins necessary for the network to exist. These protein interaction structures can be identified and studied using graph theory.

2.1 Graph theory

In this thesis, protein-protein interaction networks for *Leishmania* species are the point of study. These networks and their properties can be studied by representing interactions of proteins through graphs. In mathematics, graphs are synonymously called networks. For this thesis, the theoretical applications on graphs will be called graphs, while the experimentally formed representations of protein-protein interactions will be called networks. This is done to be consistent with existing research on these networks.

Definition 2.1 (Graph, undirected graph, simple graph)**.** A graph $G(V, E)$ is a tuple of sets where V is the set of vertices and E is a set of edges. Each edge $e \in E$ connects a pair of vertices for which the vertices are called the end points of the edge. Edges can also be denoted as uv where u connects to v. Edges can be directed where $uv \in E$ means u only connects to v, or undirected where $uv \in E$ means u connects to v, and v connects to u by the same edge. Vertex sets $V(G)$ and edge sets $E(G)$ of a graph include all the corresponding elements of the graph G. A simple graph is an undirected graph which contains no edges that connect vertices back to themselves, called loops, and no sets of multiple edges that would connect the same pairs of vertices, called parallel edges. The notation $|V|$ and $|E|$ is used to refer to the number of vertices and edges within a graph.

Definition 2.2 (Adjacent vertices)**.** For a graph $G(V, E)$, two vertices $u, v \in V$ are said to be *adjacent* if $uv \in E$. *Adjacent* vertices can also be synonymously called *neighbours*.

The end goal is to construct protein-protein interaction networks (PPIN) by representing the proteins of a certain organism as the vertices of a graph, while the edges represent interactions. As per Definition 2.1, all following mentions of graphs have the implicit assumption that the graph is undirected and simple. We also define different kinds of properties for graphs to help with the formulation and analysis of the graph theoretical side of the representation. For computational applications of graphs, adjacency matrices and adjacency lists are often used as data structures for graph representation.

Definition 2.3 (Adjacency matrix)**.** The *adjacency matrix* for a graph $G(V, E)$ is a $|V| \times |V|$ matrix, denoted as $A(G)$, which is defined as

$$A_{i,j} = \begin{cases} 1, & \text{if} \quad ij \in E \\ 0, & \text{if} \quad ij \notin E \end{cases}$$

where it is assumed that the set V is indexed to $\{1, \ldots, |V|\}$ and $i, j \in \{1, \ldots, |V|\}$.

Definition 2.4 (Adjacency list)**.** The *adjacency list* for a vertex $v \in V$ in a graph $G(V, E)$, denoted as $l_G(v)$, consists of all vertices $u \in V$ such that u and v are adjacent in G. The adjacency list of a vertex v can also be called the *neighbourhood* of v.

Adjacency lists are used as the data structure to represent graphs within this thesis since they offer fast lookups of neighbours of vertices which are heavily utilised in different algorithms that will be considered later on. The concept of graph density is introduced next, where density is a measurement of the number of edges normalised to the maximum possible number of edges.

Definition 2.5 (Graph density)**.** The *graph density*, or *density*, of $G(V, E)$, denoted as $\rho : G \to [0, 1]$, is defined as the ratio of edges and the maximum number of edges a graph with the same number of vertices could have. Graph density is denoted as

$$\rho(G) = \frac{2|E|}{|V|(|V| - 1)}.$$

Density affects the efficiency of approaches in the methods for analysis that are discussed later on in the thesis. Graphs can be called sparse or dense graphs depending on their density value.

Definition 2.6 (Sparse graph, dense graph)**.** A *sparse graph* is a graph which has low density or a significantly smaller number of edges than the largest possible number of edges. A graph G is sparse if

$$0 \leq \rho(G) \ll 1.$$

A *dense graph* is a graph which has high density or the number of edges is close to the largest possible number of edges. A graph G is dense if

$$1 \geq \rho(G) \gg 0.$$

The special cases considering density are the cases where $\rho(G) = \{0, 1\}$. A graph G with $\rho(G) = 0$ has no edges and is called an *empty graph*. On the other hand, a graph G with $\rho(G) = 1$ is called a *complete graph*.

Definition 2.7 (Complete graph)**.** A *complete graph* G is a graph in which each pair of distinct vertices is joined by an edge.

A more direct representation of the size of the neighbourhood of a vertex is also defined. These finer details near a vertex determine the structure of the graph more carefully.

Definition 2.8 (Vertex degree)**.** The *degree* of vertex $v \in V$ in a graph $G(V, E)$, denoted as $\deg(v)$, is defined as the number of distinct vertices adjacent to vertex v. The degree of the vertex equals to the size of the adjacency list $l_G(v)$ or neighbourhood of v.

The degrees of vertices can be considered for the whole graph to give metrics on the graph. The important metrics considered within this thesis are the *average vertex degree* of a graph, denoted as $\bar{d}(G)$, and the *maximum vertex degree* of a graph, denoted as $\Delta(G)$. The distribution of vertex degrees in a graph is also introduced.

Definition 2.9 (Degree distribution)**.** The number of vertices in a graph G with degree k scaled by the total number of vertices for all $k \in \mathbb{Z}_{\geq 0}$ produces the *degree distribution* of the graph G, which is denoted as $P_G(k)$.

Definition 2.10 (Power-law degree distribution, scale-free network)**.** A graph G has a *power-law degree distribution* if the vertex degree distribution $P_G(k)$ is proportional to $k^{-\gamma}$ with $\gamma > 1$. In practical applications, a network is said to have a power-law degree distribution if the tail of the degree distribution follows a power law [22, 60] and γ is often limited to $2 < \gamma < 3$. A graph, or a network, with a power-law degree distribution is synonymously called a *scale-free network*.

Graphs have other special kinds of sets of vertices with important structural features. The ones considered further in this thesis are independent sets, vertex covers and cliques.

Definition 2.11 (Independent set, maximal independent set, maximum independent set)**.** A subset S of vertices $V(G)$ is called an *independent set* if no two vertices of S are adjacent in G. An independent set is called *maximal* if there exists no $v \in V(G) \backslash S$ such that $S \cup \{v\}$ would be an independent set. An independent set is called *maximum* if G has no independent set S' with $|S'| > |S|$.

Definition 2.12 (Vertex cover, minimal vertex cover, minimum vertex cover)**.** A subset K of vertices $V(G)$ is called a *vertex covering set* or a *vertex cover* of G if every edge of G has at least one end point in K. A vertex cover is called *minimal* if there is no $v \in K$ such that $K \backslash \{v\}$ is a vertex cover. A vertex cover is called *minimum* if G has no vertex cover K' with $|K'| < |K|$.

For the definition of cliques, it is convenient to first define induced subgraphs.

Definition 2.13 (Induced subgraph)**.** *Induced subgraph* of G over a subset of vertices $S \subset V(G)$, denoted as $G[S]$, is a graph formulated by taking a subset of vertices $S = V(G[S])$ and edges such that $E(G[S]) = \{uv \in E \mid u, v \in S\}$.

Definition 2.14 (Clique, maximal clique, maximum clique)**.** A subset C of vertices $V(G)$ is called a *clique* if the induced subgraph $G[C]$ is a complete graph. A clique C is called maximal if there exists no vertex $v \in V(G) \backslash C$ such that $C \cup \{v\}$ would be a clique. A clique C is called maximum if G has no clique C' with $|C'| > |C|$.

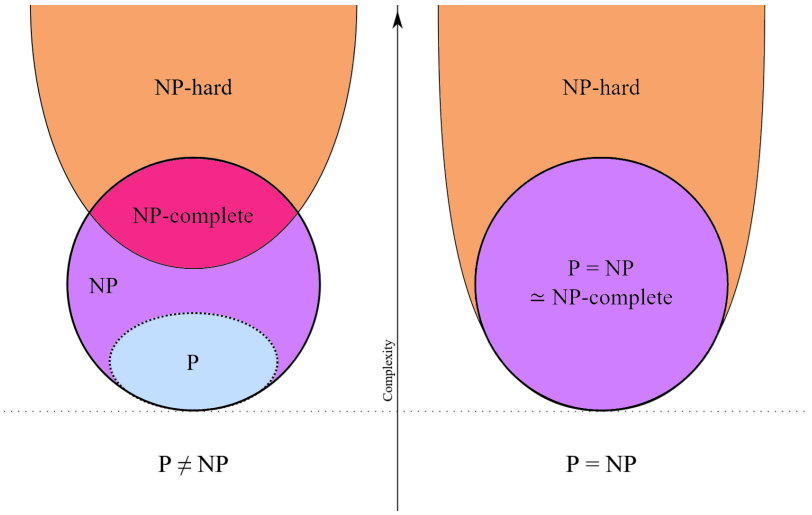


Figure 2: Venn diagram of problem complexity classes P, NP, NP-hard and NP-complete with assumptions $P \neq NP$ and $P = NP$. Image source: https://commons.wikimedia.org/wiki/File:P_np_np-complete_np-hard.svg licensed under the Creative Commons Attribution-Share Alike and edited to include colours for different classes.

Definitions 2.12 and 2.14 are directly connected to two complex network optimisation problems with biological applications: the maximum clique problem and the minimum vertex cover problem. These optimisation problems belong to a category called NP-complete problems [38] (Figure 2) which are unlikely to have solution methods that run in polynomial amount computation time, or polynomial time. The question of whether any NP-complete problem can be solved in polynomial time is one of the unsolved Millennium Prize Problems [23] and the solution to the question has a million dollar prize attached to it. As these problems are associated with finding vital proteins within protein interaction networks, they will be studied in Section 3. Finally, some additional definitions are introduced to help with the analysis of the graphs. First, we consider the definitions of walks and paths.

Definition 2.15 (Walk)**.** A *walk* in a graph $G(V, E)$ is a sequence of vertices $\{v_1, v_2, \ldots, v_n\}$ where $v_i \in V$ such that there exists an edge in the graph which joins each consecutive pair of vertices, or $v_i v_{i+1} \in E$ for $i \in \{1, \ldots, n-1\}$.

Definition 2.16 (Path)**.** A *path* in a graph G is a walk in which the vertices in the sequence $\{v_1, v_2, \ldots, v_n\}$ are unique. This can be stated as $v_i \neq v_j$ for any $i, j \in \{1, \ldots, n\}$ and $i \neq j$.

If there exists a path between two distinct vertices, the vertices are said to be *connected*. If such a path does not exist, the vertices are said to be *disconnected*. Additionally, a graph is said to be connected if every pair of vertices is connected, and disconnected otherwise. These definitions give us a way to break graphs into distinct components.

Definition 2.17 (Pairwise disjoint sets)**.** Sets S_1 and S_2 are called *pairwise disjoint* or *disjoint* if $S_1 \cap S_2 = \emptyset$.

Definition 2.18 (Connected component)**.** A *connected component* C, which is sometimes called a *maximally connected subgraph*, is an induced subgraph of G such that there exists no vertex $u \in V(C)$ which is disconnected from any other vertex $v \in V(C)$ and that there exists no vertex $w \notin V(C)$ such that $uw \in E(G)$. $C(v)$ is used as the notation for the connected component that includes the vertex v.

With the notion of paths, there also exists a notion of a shortest path between a source vertex $s \in V(G)$ and some target vertex $t \in V(G)$. The shortest path corresponds to the shortest sequence $\{v_i\}_{i=1}^{n}$ of distinct vertices such that $s = v_1$ and $t = v_n$. Calculating shortest paths between vertices offers two tools which are used within the scope of this thesis. First, the number of unique shortest paths between two vertices $s, t \in V(G)$ is denoted as σ_{st} and the number of unique shortest paths between two vertices $s, t \in V(G)$ that pass through some other vertex $u \in V(G)$, which is distinct from s and v, is denoted as $\sigma_{st}(u)$. Second, the geodesic distance is considered as the definition of the distance between vertices in graphs.

Definition 2.19 (Graph geodesic distance)**.** The *geodesic distance* in a graph G between vertices $u, v \in V(G)$ with $u \neq v$, denoted as $\mathrm{d}(u, v)$, is defined as the number of edges traversed on the shortest path between u and v. For the case $u = v$, it is defined separately that $\mathrm{d}(u, u) = 0$. If u and v are disconnected, the distance is set to be infinite.

2.2 Formulations for the optimisation problems

To study the optimisation problems related to cliques and vertex covers, exact formulations are formed which attempt to find exact solutions to the problems.

Definition 2.20 (Maximum clique problem)**.** The *maximum clique problem* (MCP) on a graph G is the maximisation problem of finding a maximum clique in G.

The MCP for a graph $G(V, E)$ may be formatted as an ILP such that

$$\begin{aligned} \max \quad & \sum_{i \in V} x_i, \\ \text{s.t.} \quad & x_i + x_j \leq 1, \quad i, j \in V,\ ij \notin E \\ & x_i \in \{0, 1\}, \quad i \in V \end{aligned} \tag{1}$$

where x can be represented by a binary vector corresponding to vertices formulated from V where the values indicate whether vertex i is chosen in the clique.

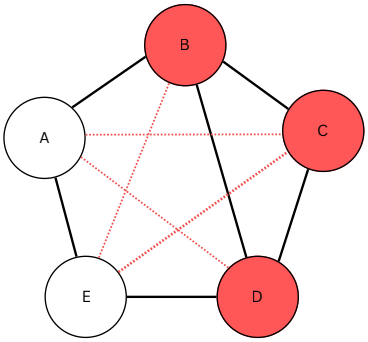


Figure 3: A maximum clique of an example graph is represented with red vertices. The clique is valid if each of the vertices in the clique is adjacent to the others. Edges of the graph are represented as black solid lines, and pairs of vertices corresponding to the constraints of the ILP formulation of the MCP are represented as red dashed lines.

In the ILP formulation (1), the conditions required to find a clique are constructed by setting a constraint for each pair of vertices that are not adjacent (Figure 3). The constraint restricts that only one of the vertices in such a pair can be chosen in the clique.

Remark 2.21. Any clique of a graph G is fully contained in one of the connected components of the graph.

With Definitions 2.14 and 2.18 it is relatively easy to see that Remark 2.21 holds and it allows us to consider parts of a graph separately when trying to solve the MCP. The main benefit comes from noticing that any combination of two vertices in different connected components requires a constraint in the ILP formulation, which is inefficient memory usage. After running an algorithm for the exact solution separately on the connected components, the best solution found between the components gives the largest clique within the whole graph.

Definition 2.22 (Minimum vertex cover problem)**.** The *minimum vertex cover problem* (MVCP) on a graph G is the minimisation problem of finding a minimum vertex cover in G.

The MVCP for a graph $G(V, E)$ may be formatted through ILP such that

$$\begin{aligned} \min \quad & \sum_{i \in V} x_i, \\ \text{s.t.} \quad & x_i + x_j \geq 1, \quad ij \in E \\ & x_i \in \{0, 1\}, \quad i \in V \end{aligned} \tag{2}$$

where x can be represented by a binary vector corresponding to vertices formulated from V where the values indicate whether vertex i is chosen in the vertex cover.

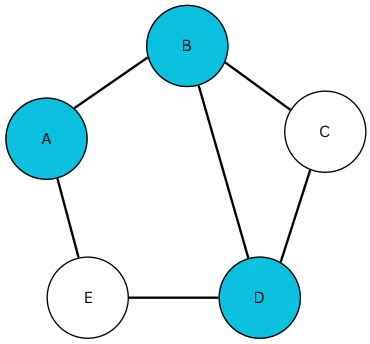


Figure 4: A minimum vertex cover of an example graph is represented with blue vertices. Each edge in the graph must have one endpoint in the vertex cover. Additionally, each edge corresponds to a single constraint in the ILP formulation.

In ILP formulation (2), the MVCP is formulated by creating a constraint for each edge (Figure 4). The definition of a vertex cover is fulfilled by choosing at least one endpoint of every edge. Similarly to cliques, this can also be considered separately for the connected components of the graph.

Remark 2.23. A vertex cover of a graph G can be expressed as the union of vertex covers over the connected components of the graph.

This can be useful considering some of the methods may have undefined behaviour for vertices that are disconnected. To conclude the section, the maximum independent set problem is also defined.

Definition 2.24 (Maximum independent set problem)**.** The *maximum independent set problem* (MISP) on a graph G is the maximisation problem of finding a maximum independent set in G.

Even though the maximum independent set problem is not directly studied for biological networks, it is still useful to define the problem since the MISP is inherently connected to the MVCP, which is utilised further when the heuristic methods are discussed in Section 3.2.2.

2.3 Centrality measures

Centrality measures attempt to answer the question of how important a vertex is in a graph. Previous research on PPINs has utilised centrality measures as a direct approach to find the most vital proteins [35] where the vertices with highest centrality values predict importance in PPINs. Degree centrality follows the idea that vertices which have more neighbours are considered to be more central or important in the graph.

Definition 2.25 (Degree centrality)**.** The *degree centrality* of a vertex u in a graph G is equivalent to the degree of the vertex. This is defined as

$$D_C(u) = \deg(u).$$

Closeness centrality measures how close a vertex is to the other vertices in the graph. A vertex is considered to be more central if the shorter paths to other vertices are short.

Definition 2.26 (Closeness centrality)**.** The *closeness centrality* [8, 57] of a vertex $v \in V$ in a graph G is often defined as

$$\frac{1}{\sum_{u \in S(v)} \mathrm{d}(u, v)}$$

where $\mathrm{d}(u, v)$ is the geodesic distance between the vertices and $S(v) \subset V(G)$ is the set of vertices that are connected to v. If the graph is disconnected, the centrality is considered separately for the connected components C of the graph [65]. In such cases, the centrality can be normalised over the whole graph, giving us the more general definition

$$C_C(v) = \frac{|V(C(v))| - 1}{|V(G)| - 1} \frac{1}{\sum_{u \in C(v) \setminus \{v\}} \mathrm{d}(u, v)}.$$

Remark 2.27. Definition 2.26 assumes that the graph G has at least two vertices in each connected component.

Betweenness centrality measures how many shortest paths pass through a specific vertex. A vertex is considered to be more central if many shortest paths within a graph pass through the vertex.

Definition 2.28 (Betweenness centrality)**.** The *betweenness centrality* [28] of a vertex $v \in V$ in a graph $G(V, E)$ is defined as

$$C_B(v) = \sum_{s,t \in V : s \neq v \neq t} \frac{\sigma_{st}(v)}{\max(\sigma_{st}, 1)}.$$

Eigenvector centrality of a vertex is proportional to the sum of the centralities within the neighbourhood of the vertex. Essentially, a vertex is considered to have high centrality if other vertices of high centrality are adjacent to it. In practice, eigenvector centrality is calculated iteratively through power iteration, where in each iteration a candidate vector is multiplied by the adjacency matrix and then normalised.

Definition 2.29 (Eigenvector centrality)**.** The *eigenvector centrality* [12] of a vertex $v \in V(G)$ in a graph G is defined as the corresponding index of the principal left eigenvector of the adjacency matrix A corresponding to the graph G.

The algorithm for eigenvector centrality may be altered to handle issues within the calculation. A simple reconstruction is the *Katz-Bonacich centrality* which gives each vertex some baseline value of centrality [12, 39]. An additional alteration to eigenvector centrality is normalising the adjacency matrix, which gives us the *PageRank* algorithm [13, 32] which was famously developed for Google to solve the problem of web-indexing.

Definition 2.30 (PageRank)**.** The *PageRank* value of a vertex $v \in V$ in a graph $G(V, E)$ is the corresponding index of the principal left eigenvector of matrix M defined as

$$M = (1 - q)Ap^\top + \frac{q}{|V|}\mathbf{e}_{|V|}\mathbf{e}_{|V|}^\top$$

where q is the jump probability, A is the adjacency matrix of the graph G, p is a $|V|$-dimensional vector defined corresponding to the indices for vertices in A as

$$p_v = \begin{cases} \frac{1}{|V|}, & \text{if } \deg(v) = 0, \\ \frac{1}{\deg(v)}, & \text{otherwise} \end{cases}$$

and $\mathbf{e}_{|V|}$ is the $|V|$-dimensional vector of ones.

Calculating the PageRank values is similar to the power iteration for eigenvector centrality. Finally, we consider how vertices preserve their connections while vertices with lower degrees are removed. A notation of k-cores and shells is given to construct the last centrality method considered within this thesis [44, 58].

Definition 2.31 (k-core)**.** The *k-core* of a graph G with $k \geq 1$ is defined as the subgraph of G which is formulated by iteratively removing vertices of degree $\{0, \ldots, k-1\}$ until the graph stays unchanged. The original graph can be referred to as the 0-core.

Definition 2.32 (Shell index)**.** The *shell index* of a vertex $v \in V(G)$, denoted as $S(v)$, is defined as the largest k value where v is included in the corresponding k-core of the graph G.

The shell indices of vertices can be used as a metric of centrality (Figure 5). However, this is not a good way to approach creating a centrality method, as only considering shell indices is guaranteed to produce a large set of vertices with equal centrality values. This is not preferred when the goal is to identify the most important vertices of the graph.

Remark 2.33. A non-empty k-core in a graph has at least $k + 1$ vertices.

Producing large sets of equivalently important nodes can be avoided by considering the shell-indices of the neighbourhood of a specific vertex. This construction gives the following formulation of coreness centrality.

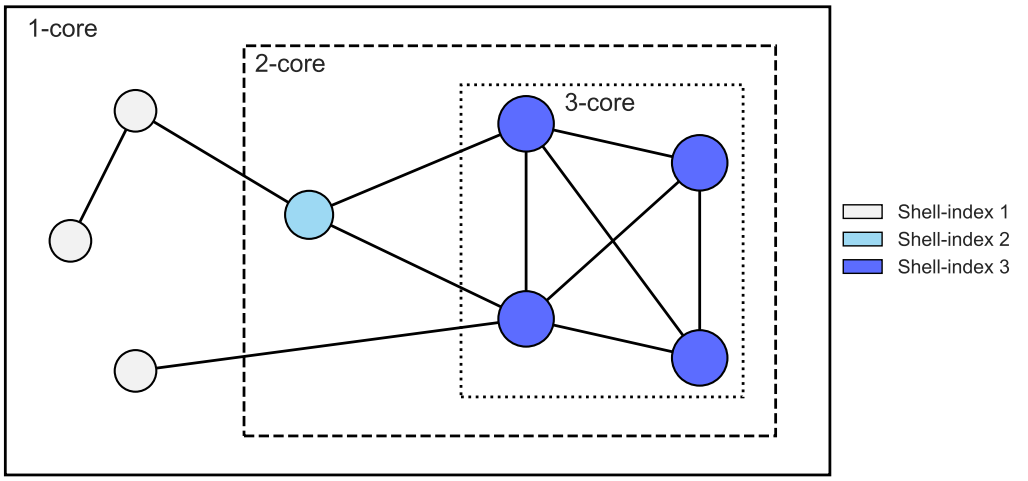


Figure 5: Representation of k-cores and vertex shell indices. k-cores are formed by iteratively removing vertices of degree $k-1$ or less until the graph stays unchanged. The cores are represented as parts of the graph enclosed within the lines and vertex shell indices are represented by colouring.

Definition 2.34 (Coreness centrality)**.** The *coreness centrality* of a vertex v in a graph G is defined by summing the shell indices of all the neighbours of v. This can be represented as

$$C_K(v) = \sum_{u \in l_G(v)} S(u).$$

All centrality methods are visualised on a small sparse example graph in Figure 6. Most centrality definitions include normalisation multipliers, but in this thesis they are left out unless they are necessary for the definition, like in the case of Definition 2.26. Normalisation parameters based on number of vertices in the graph are often used since it enables comparisons between different graphs.

Remark 2.35. Due to varying time complexity between the centrality methods, the time complexity for an arbitrary centrality method is noted as $O(C_M)$ in the relevant parts of the thesis.

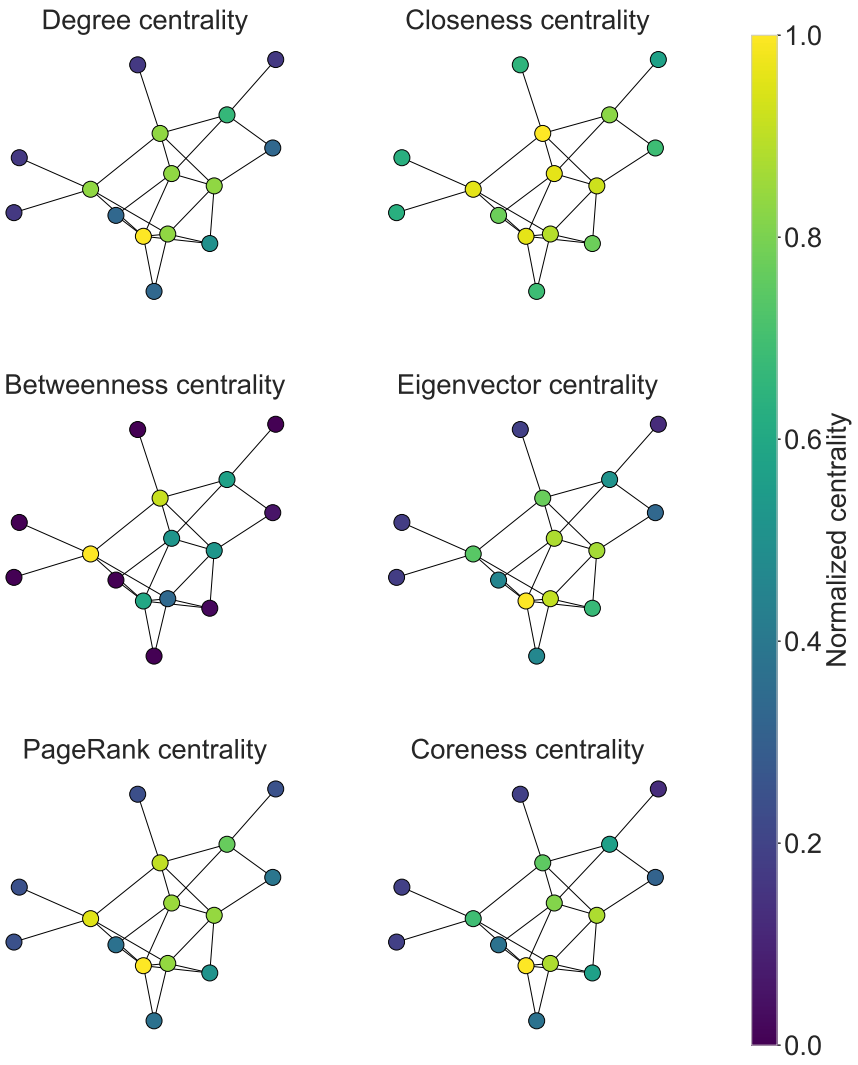


Figure 6: Centrality measures were calculated for the vertices of a small random sparse graph. Each method assigns importance to vertices, which is denoted with the colours. Value of 1 (in yellow) is given to the most important vertex, and other vertices are normalised to the interval $[0, 1]$. The centrality methods often produce similar results, as most of the centrality values seem to correlate with the degrees of the vertices. Betweenness centrality stands out as the vertex with the highest centrality, as the method gives higher values to vertices that are connected to other important vertices. Thus, a different vertex compared to the other methods is given the highest importance. Closeness centrality is also distinct, as the geodesic distances from each vertex are similar on average, resulting in more similar centrality values across vertices.

3 Solution methods

In previous research, graph theoretical methods of identifying vital proteins have studied centrality measures [46, 56] or NP-complete problems like MCP [1, 69] and MVCP [20]. The novel part of our research focuses on combining these methods where centrality is used as the base for finding cliques and vertex covers within PPINs. In this section, concrete simplifications to ILP formulations and heuristic methods are discussed which are used to find solution candidates to problems described in Definitions 2.20 and 2.22.

There are no deterministic polynomial-time algorithms that exactly solve the optimisation problems in Definitions 2.20, 2.22 and 2.24. With large enough graphs, the computation of exact solutions with in-memory procedures, such as ILP formulations on Gurobi and SCIP, easily run out of memory either while constructing the constraints or while attempting to solve the problem exactly. This becomes evident quickly with the MCP ILP formulations for a sparse graph $G(V, E)$ since there are $|V|(|V|-1)/2-|E|$ constraints and $|V|$ binary decision variables. With the definition for graph density, the constraint count may be rewritten as $|E|\left(\rho(G)^{-1}-1\right)$. The number of constraints is scaled both by the number of edges and the inverse of the density, which is much larger than 1 for sparse graphs. This significantly increases the memory requirement for the full ILP formulation. Even if the memory requirement would not be a problem, combinatorial optimisation problems in general are not easy to solve exactly due to large number of vertices and NP-completeness. This highlights the importance of reducing the optimisation problem size and finding heuristic approaches that find maximal cliques and minimal vertex covers that are relatively close to the corresponding problems' global optima.

Vertices with high centrality have been observed to more likely belong to large cliques [45] and small vertex covers [37]. Thus, a natural idea would be to construct valid solutions to the optimisation problems by picking nodes to solution candidates based on centrality. The following methods are designed to be used on sparse graphs with properties similar to those of scale-free graphs, but they can be applied on any graph.

3.1 Cliques

In the context of the biological application on PPINs, a clique represents a set of proteins that all interact pairwise with each other. Large cliques are used as a predictive tool for functional modules in organisms [68]. These sets of proteins are clearly of interest, as such functional modules can be vital for an organism. Thus, affecting the vital groups of proteins can offer approaches for new treatments and vaccines against diseases. Methods of finding large cliques from PPINs is a relevant question and one of the main points of study within the thesis.

3.1.1 MCP ILP formulation simplifications

This section discusses methods to simplify ILP formulations for the MCP. These methods are built around removing vertices from the MCP consideration which can not be part of cliques above a certain size. This process of removing vertices from consideration is referred to as *pruning* of graphs in this thesis. The main approach studied within this thesis uses k-cores and the definition of cliques.

Theorem 3.1. A clique of size k is fully included in $(k' - 1)$-core of graph G, if $k \geq k'$.

Proof. This follows immediately from Definitions 2.7, 2.14 and 2.31. When the $(k' - 1)$-core is built by iteratively removing vertices with degree $\{0, \ldots, k' - 2\}$, all vertices within the clique of size k' have a vertex degree of at least $k' - 1$ within the subgraph and thus no vertex within the clique is removed within the process. □

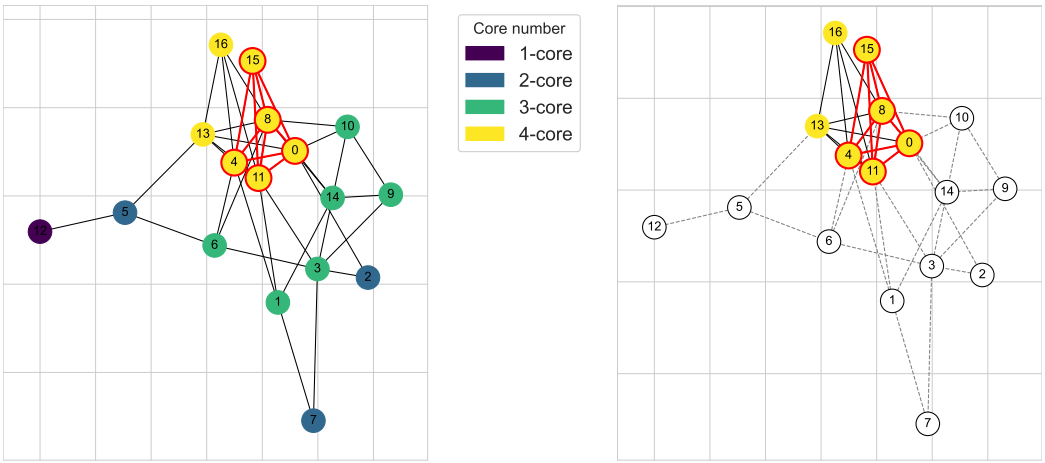


Figure 7: Illustration of taking a 4-core of a graph with a maximum clique of size 5 which is highlighted with red outlining. The maximum clique is kept in the 4-core as per Theorem 3.1. Vertices removed within the k-core iteration are displayed with white vertices and dashed edges.

Preserving cliques while taking k-cores is illustrated in Figure 7. In sparse graphs with low average degree, this can increase the graph density by a notable margin, which, on the other hand, significantly reduces the number of constraints for the optimisation problem. There exist other more aggressive pruning methods, such as the *k-nub* algorithm introduced by Chan, Morgan and Ugon [18]. However, the k-nub algorithm is based on counting the number of cliques of certain sizes. This can be inefficient in large, sparse graphs when compared to taking k-cores with fast methods for creating lower bounds for the maximum clique. Additionally, to obtain exact solutions to the maximum clique problem with less memory usage, the graphs can be partitioned into connected components according to Remark 2.21.

3.1.2 Clique heuristics

In this section, we introduce heuristic algorithms for the MCP. First, we consider a straightforward approach to constructing cliques in graphs through the Definition 2.14. The construction is done by finding a starting vertex and then selecting new vertices to the clique with the requirement that any added vertex must be joined by an edge to all of the previously selected vertices. This selection process of the starting vertex, or seed vertex, and the subsequent vertices is proposed as Algorithm 1. In this case, the selection uses centrality such that the vertex with highest centrality is chosen as the seed vertex and the rest of the vertices are considered as possible additions to the clique in the order of the centrality values.

Algorithm 1: CENTRALITY-ORDER CLIQUE HEURISTIC

Input: graph G, centrality method C_M

Calculate centrality values of vertices in G with C_M
Sort vertices of G by their centrality in decreasing order, save the list to L_0
Pick vertex with largest centrality as the seed s
Initialise $currClique = \{s\}$
Filter L_0 such that only nodes in $l_G(v)$ are included, save filtered list to L
for $v \in L$ **do**
    if v is adjacent with all vertices in $currClique$ **then**
        $currClique = currClique \cup \{v\}$
Return $currClique$

Algorithm 1 runs a check for all neighbours of the vertex with the highest centrality. An alternative algorithm is formulated by calculating vertex centralities in the induced subgraph of some seed vertex s and the neighbourhood of the seed vertex. This approach is proposed since the centrality values calculated for all of the vertices in a graph has information from vertices which can not be included in the formulated clique after deciding on a seed vertex. After pruning vertices from consideration, a procedure similar to 1 is performed, giving us Algorithm 2.

Theorem 3.2. Algorithms 1 and 2 produce a maximal clique of $G(V, E)$ in $O(O(C_M)+|V|\log|V|+|V|\Delta(G)+\Delta(G)^3)$ and $O(O(C_M(G))+|V|+\Delta(G)^3)$ time, respectively, where $O(C_M)$ is the time complexity of the centrality method C_M and $\Delta(G)$ is the maximum vertex degree of graph G. The produced clique is a lower bound for the maximum clique.

Proof. The part of producing a valid clique is straightforward, as vertices can only be accepted if adding the vertex fulfils the clique definition. As each vertex is considered as a possible addition to the clique, the found clique is also maximal. Since this is a valid clique, it is always at most equal in size to the maximum clique by Definition 2.14 which makes it a lower bound for the maximum clique.

Algorithm 2: SUBGRAPH CENTRALITY-ORDER CLIQUE HEURISTIC

Input: graph G, centrality method C_M

1 Calculate centrality values of vertices in G with C_M
2 Find and pick vertex with largest centrality as the seed s
3 Initialise $currClique = \{s\}$
4 Formulate the induced subgraph $G[l_G(s) \cup \{s\}]$ and save to SG
5 Calculate centrality values of vertices in SG with C_M
6 Sort vertices of SG by their centrality in decreasing order, save the list as L
7 **for** $v \in L \backslash \{s\}$ **do**
8 **if** v is adjacent with all vertices in $currClique$ **then**
9 Add v to $currClique$
10 Return $currClique$

For time complexity of Algorithm 1, the centrality calculation and sorting the vertices based on centrality values on lines 1 and 2 are calculated in $O(C_M)+|V|\log|V|$ time. Lines 3 and 4 are trivial operations after sorting, and line 5 requires $|V|$ linear searches from a set of $\Delta(G)$ elements in the worst case scenario. For the clique formulation, each neighbouring vertex is processed once, leading to $\Delta(G)$ iterations of the for-loop in lines 6-8. On line 7, each vertex is linearly searched from at most $\Delta(G)$ lists which each have at most $\Delta(G)$ elements each. This leads to the total time complexity of $O(O(C_M) + |V|\log|V| + |V|\Delta(G) + \Delta(G)^3)$ in the worst case.

Algorithm 2 does not require sorting after the initial calculation of centrality and thus finding the vertex with the largest centrality only requires $O(|V|)$ time. Building the subgraph on line 4 requires $O(\Delta(G)^3)$ time. The subgraph has up to $\Delta(G)$ vertices for which the new adjacency lists have to be formulated. The formulation of each adjacency list requires that up to $\Delta(G)$ nodes are linearly searched from the original adjacency list which has up to $\Delta(G)$ elements. In the context of this thesis, the calculation of centrality values for the subgraph on line 5 is bounded from above by the time complexity of the calculation of centrality values for the original graph, and the sorting of the vertices on line 6 takes $\Delta(G)\log\Delta(G)$ time. The rest of the algorithm is equivalent to lines 6-8 of Algorithm 1. This leads to the total time complexity of $O(2O(C_M) + |V| + \Delta(G)\log\Delta(G) + 2\Delta(G)^3)$ which is simplified to time complexity of $O(O(C_M) + |V| + \Delta(G)^3)$. □

3.2 Vertex covers

In the process of identifying vital proteins, formulating a vertex cover of a PPIN provides a way to identify which proteins collectively account for all protein interactions in the organism [11, 34]. This means that for each interaction, at least one of the proteins in the set of vital proteins must be included in the interaction. This differs from the question we have for cliques where the goal is to find protein modules. Thus, vertex covers in the context of PPINs are a more conservative method of finding vital proteins, since a vertex cover typically consists of a significantly larger set of proteins.

3.2.1 MVCP ILP formulation simplifications

In this section, we discuss possible simplifications to the ILP formulations for the MVCP. Similarly to the MCP, which can be reduced in the optimisation problem size by pruning via k-cores, the MVCP has some approaches that quantify which vertices must be included in the minimum vertex cover. However, the reduction in the size of the ILP formulation is much smaller than with cliques. This can be seen, for example, in the MVCP ILP formulation for a graph $G = (V, E)$ that only has $|E|$ constraints. Additionally, vertex covers in scale-free networks often have properties where the minimum vertex covers include most of the vertices [10]. Thus, a large number of vertices and edges can be included in the optimal solution to the MVCP. However, it should be noted that metrics like low graph density do not universally correlate with a large minimum vertex cover size, although the two metrics are more correlated in scale-free networks. For example, a star graph of $n > 3$ vertices can be considered sparse, while the vertex cover size is always 1.

One straightforward method of pruning is to identify all vertices with degree 0 or 1. Vertices of degree 0, called isolated vertices, are removed from consideration as they are not required to be in the vertex cover by Definition 2.12. For the consideration of vertices of degree 1, called leaf vertices, the vertex adjacent to the leaf vertex is always chosen for the minimum vertex cover unless the leaf vertex itself was already selected before. However, this method can be inefficient for graphs with power-law degree distributions, as a notable number of vertices often have at least two neighbours. A more significant alteration in the approach can be made by considering the problem of finding a maximum independent set based on the following theorem.

Theorem 3.3. For a graph $G(V, E)$, a subset of vertices $S \subset V$ is a vertex cover if and only if the subset $V \backslash S$ is an independent set.

Proof. For the proof we first consider the fact that the sets S and $V \backslash S$ are by definition disjoint. This means there exists no element $v \in V$ which would be included in both of the sets S and $V \backslash S$. Additionally, we have $S \cup (V \backslash S) = V$.

$\Leftarrow$: Let $G(V, E)$ be an arbitrary graph. Let $S \subset V$ be an independent set of the graph G. By Definition 2.11, for any edge in the graph $uv \in E$, at most one of the cases $u \in S$ or $v \in S$ holds. This implies that for every edge $uv \in E$ at least one endpoint is in the set $V \backslash S$ which is by Definition 2.12 a vertex cover.

$\Rightarrow$: Let $G(V, E)$ be an arbitrary graph. Let $S \subset V$ be a vertex cover and assume that the set $V \backslash S$ is not an independent set. Then by Definition 2.11 there exists an edge $uv \in E$ for which $u \in V \backslash S$ and $v \in V \backslash S$. However, this implies that we must have $u, v \notin S$ which is a contradiction to the assumption that S is a vertex cover, proving the claim. □

The transformation between MISP and MVCP visualised in Figure 8 is notable, but the ILP formulations for both problems are similar. Thus, Theorem 3.3 does not offer significant improvements to the size of the ILP formulation. However, this connection is still utilised in the MVCP heuristics.

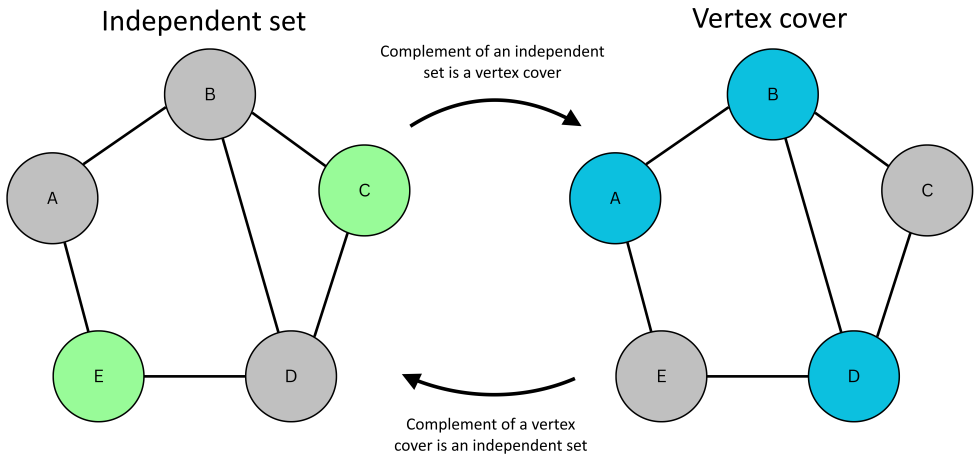


Figure 8: Visualisation of the connection between vertex covers and independent sets described in Theorem 3.3 on an example graph. The left graph represents the maximum independent set with the green vertices while the right graph represents the minimum vertex cover with the blue vertices. The grey vertices represent the complement for each of the previously mentioned sets.

3.2.2 Vertex cover heuristics

In this section, we introduce heuristic algorithms for the MVCP. As vertex covers require each edge to be covered by one of the endpoint vertices, finding heuristic approaches for valid vertex covers have a few straightforward approaches. One simple way to find nontrivial vertex covers is through checking all vertices in decreasing order of centrality and adding each newly covered edge into a list of checked edges. If any edges are added, the vertex is added to the vertex cover. This approach is modified to construct Algorithm 3 where the vertices and covered edges are removed when the vertices are added to the vertex cover, and the graph is reduced to an empty graph over the algorithm.

Algorithm 3: CENTRALITY-ORDER VERTEX COVER HEURISTIC

Input: graph G, centrality method C_M

Calculate centrality values of vertices in G with C_M
Sort vertices by their centrality values in decreasing order, save the list as L
Initialise $vertexCover = \{\}$
Initialise $remainingEdges = E(G)$
for $v \in L$ **do**
    Calculate $removed = remainingEdges \cap \{uv \in E(G) \mid u \in l_G(v)\}$
    if $|removed|>0$ **then**
        Add v to $vertexCover$
        $remainingEdges = remainingEdges \backslash removed$
Return $vertexCover$

Alternatively, as each of the edges needs to be covered, a natural idea for an algorithm would be to go through every edge of the graph and either picking the vertex with higher centrality or skipping the edge if one of the endpoints of the edge is already chosen. This procedure is constructed as Algorithm 4.

Algorithm 4: CENTRALITY-BASED VERTEX COVER HEURISTIC WITH EDGE INSPECTION

```
Input: graph G, centrality method C_M
Calculate centrality values of vertices in G with C_M
Save centrality values as a dictionary C
Initialise vertexCover = {}
for uv ∈ E do
    if u ∈ vertexCover or v ∈ vertexCover then
        Continue
    if C(u) > C(v) then
        Add u to vertexCover
        Continue
    Add v to vertexCover
Return vertexCover
```

The procedure for Algorithm 4 could be altered to account for the most central nodes first. This would require making a centrality based ordering for the edges before the for-loop. Ordering leads to a longer runtime as calculating the edge weights and sorting them takes at least additional $O(|E| \log |E|)$ time for the sorting. For the third vertex cover algorithm, we consider the idea that vertex covers and independent sets are connected through Theorem 3.3.

Algorithm 5: CENTRALITY-ORDER VERTEX COVER HEURISTIC USING INDEPENDENT SETS

```
Input: graph G, centrality method C_M
Calculate centrality values of vertices in G with C_M
Sort vertices by their centrality values in increasing order, save the list as L
Initialise independentSet = {}
Initialise setBoundary = {}
for v ∈ L do
    if v ∉ setBoundary then
        Add v to independentSet
        Add l_G(v) to setBoundary
Return V\independentSet
```

Unlike the other algorithms, Algorithm 5 prioritises vertices with low centrality. This comes from the observation that vertices with high centrality tend to belong to small vertex covers, and thus by Theorem 3.3 vertices with low centrality should belong to large independent set. Independent sets can be built by considering a *boundary* which includes the nodes that are neighbours to at least one of the nodes in the independent set. The boundary formulation enables the for-loop to only consider sets of vertices, which can make the algorithm more efficient when compared to the other algorithms that require looping over lists of edges. It should be noted that there is additional complexity when taking the set difference to return the vertex cover, but the additional complexity is also only dependent on the number of vertices.

Theorem 3.4. Algorithms 3, 4 and 5 produce a vertex cover for $G(V, E)$ in $O(O(C_M)+|V|\log|V|+|V||E|^2)$, $O(O(C_M)+|V||E|)$ and $O(O(C_M)+|V|^2\Delta(G))$ time respectively. The produced vertex cover is an upper bound for the minimum vertex cover.

Proof. Algorithm 3 checks that all edges are added to the set $checkedEdges$ and the only cases where vertices are not added are the cases where all edges have been checked by some other vertex. Algorithm 4 directly checks all edges and that at least one end point of each edge is chosen to the vertex cover. Algorithm 5 applies Theorem 3.3 in the return step and thus it is required to consider whether the independent set produced within the algorithm is valid. In the independent set construction, the vertices are picked for the set if they are not adjacent to any other picked vertices in the independent set. This ensures that no two adjacent vertices are picked, and thus the resulting independent set and the returned vertex cover are valid. For time complexity consideration, centrality calculation takes $O(C_M)$ time and sorting of vertices takes $|V|\log|V|$ time in Algorithms 3 and 5.

Algorithm 3 runs the for-loop in lines 5-9 once for each vertex, leading to $|V|$ iterations. For each iteration, the edges that need to be removed are identified. This takes at most $\Delta(G)$ lookups from a list of at most $|E|$ elements on line 6. Removing the edges takes $|E|$ linear searches on a list of $|E|$ elements in the worst case. Considering this, each for-loop takes at most $O(\Delta(G)|E|+|E|^2)$ time which is equivalent to $O(|E|^2)$ time since $\Delta(G)\leq|E|$. This gives the total time complexity of $O(O(C_M)+|V|\log|V|+|V||E|^2)$.

Algorithm 4 runs the for-loop in lines 4-9 once for each edge, leading to $|E|$ iterations. For each iteration, the algorithm performs a linear search of two vertices from a list of up to $|V|$ elements. Since each of the remaining operations, including dictionary lookups, value comparisons and added vertices to the vertex cover, are performed in constant time, this gives the total time complexity of $O(O(C_M)+|V||E|)$.

Algorithm 5 runs the for-loop in lines 5-8 once for each vertex, leading to $|V|$ iterations. For each iteration, a linear search is performed on a list of up to $|V|$ vertices to check whether the vertex is in the boundary. For each v not in the boundary, up to $\Delta(G)$ elements are added to the boundary. After the for-loop, the algorithm performs a set difference before returning the vertex cover, which takes at most $|V|$ linear searches on a list of up to $|V|$ elements. This gives the total time complexity of $O(O(C_M)+|V|\log|V|+|V|^2\Delta(G)+|V|^2)$ which is simplified to $O(O(C_M)+|V|^2\Delta(G))$ time. □

4 Method benchmarking

In this section, the previously discussed exact methods and heuristic algorithms are compared on benchmarking graphs for clique and vertex cover problems along with sets of randomly generated sparse graphs. Two ILP solvers, a commercial solver Gurobi [30], and an open-source integer-programming solver SCIP [70], are run on each of the ILPs formulated for the benchmark graphs. For the clique problem ILP formulations, Algorithm 1 is run with degree centrality to create a fast lower bound for the maximum clique size. This lower bound is used with Theorem 3.1 to take the k-core of the original graph, which is done to simplify the ILP formulation. For the MVCP, the ILP formulations are solved directly for the whole graph.

The heuristic solutions are computed by first calculating the centrality for the whole graph and then applying Algorithms 1-5. For the sake of consistency, all centrality values are calculated once for each graph. This ensures that the contribution of centrality value computation to the runtime is consistent across the different algorithms. The runtime of the heuristic method is considered to be the combined runtime of the centrality calculation and the runtime of the algorithm. The whole outline of the benchmarking process is described in Figure 9.

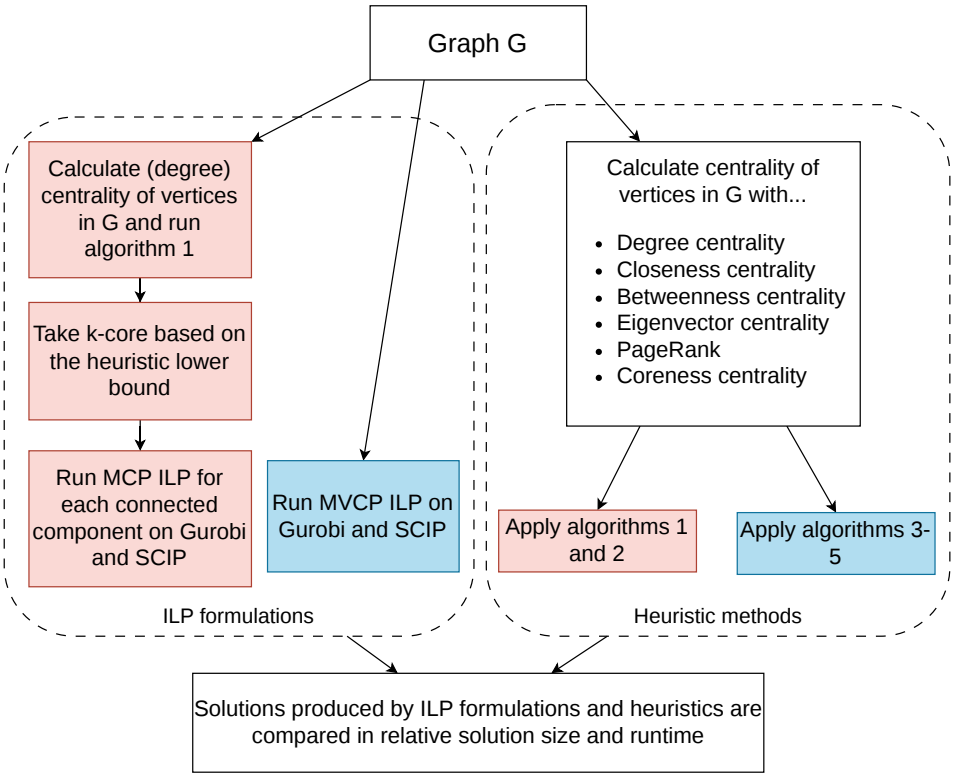


Figure 9: Flowchart describing the benchmark procedure for an arbitrary graph G. Steps related to cliques are highlighted in red and steps related to vertex covers are highlighted in blue.

Table 1: DIMACS benchmarking set on the maximum clique and vertex cover problems. The ω column has the expected size of the maximum clique. The VC column has the best upper and lower bounds for the vertex cover size found using Gurobi and SCIP. The values marked with an asterisk means that the maximum clique or minimum vertex cover size has been proven optimal.

Instance	|V|	|E|	$\omega(G)$	VC(G)	
				Upper-bound	Lower-bound
brock200_2	200	9 876	12	*189	–
brock200_4	200	13 089	17	*192	–
brock400_2	400	59 786	29	*392	–
brock400_4	400	59 765	33	*393	–
brock800_2	800	208 166	24	791	752
brock800_4	800	207 643	26	791	752
C125.9	125	6 963	*34	121	–
C250.9	250	27 984	*44	*245	–
C500.9	500	112 332	57	*495	–
C1000.9	1 000	450 079	68	994	979
C2000.5	2 000	999 836	*16	1 987	1 817
C2000.9	2 000	1 799 532	80	1 995	1 952
C4000.5	4 000	4 000 268	*18	3 988	2 000
DSJC500.5	500	125 248	13	488	461
DSJC1000.5	1 000	499 652	15	987	912
gen200_p0.9_44	200	17 910	44	*195	–
gen200_p0.9_55	200	17 910	55	*195	–
gen400_p0.9_55	400	71 820	55	*392	–
gen400_p0.9_65	400	71 820	65	*393	–
gen400_p0.9_75	400	71 820	75	*394	–
hamming8-4	256	20 864	16	*240	–
hamming10-4	1 024	434 176	40	1 004	985
keller4	171	9 435	11	*156	–
keller5	776	225 990	27	745	738
keller6	3 361	4 619 898	59	3 305	–
MANN_a27	378	70 551	126	*375	–
MANN_a45	1 035	533 115	345	*1 032	–
MANN_a81	3 321	5 506 380	*1 100	*3 318	–
p_hat300-1	300	10 933	8	261	256
p_hat300-2	300	21 928	25	*273	–
p_hat300-3	300	33 390	36	*291	–
p_hat700-1	700	60 999	11	635	590
p_hat700-2	700	121 728	44	651	633
p_hat700-3	700	183 010	62	690	669
p_hat1500-1	1 500	284 923	12	1 418	1 268
p_hat1500-2	1 500	568 960	*65	1 439	1 348
p_hat1500-3	1 500	847 244	*94	1 490	1 423

The set of preconfigured graphs used for the second DIMACS challenge [36] is intended for initial benchmarking. At 37 graphs (Table 1), the benchmarking set is relatively small but well-established for the MCP. As this set of graphs is typically used for the MCP, it does not have established best-known solutions for the MVCP. However, other researchers have also benchmarked MVCP heuristics on the DIMACS graphs [51, 62] and thus it was chosen to be used for both optimisation problems. To ensure reasonable runtimes with the ILP solver procedures and avoid out-of-memory crashes, a 600-second time limit is imposed on the optimisation step. With these limitations in mind, random scale-free graphs with a closer nature to the PPINs are used as an alternative benchmarking option.

For the formulation of random scale-free graphs, similar sparsity and a similar degree distribution to the studied networks were prioritised as properties for the benchmark graph formulation. To simulate the graphs, we consider the degree distributions of the networks which will be considered further in Section 5.1. Here, a power-law distribution is fitted using the Powerlaw-package for Python [2], which gives the parameters presented in Table 2.

Table 2: Results of power-law distribution fitting for the degree distributions formed for the PPINs of the *Leishmania* species. The formulation of the networks is described in detail in Section 5.1. $|V|$ represents the number of proteins within the network, k_{min} represents the lower bound such that the vertex degrees above the lower bound are assumed to follow a power-law distribution and γ describes the exponent of the fitted distribution proportional to $k^{-\gamma}$ where $k \geq k_{min}$.

Network	$\lvert V\rvert$	$\bar{d}$	k_{min}	γ
L. amazonensis	229	9.17	6	2.554
L. braziliensis	16451	31.35	13	2.084
L. donovani	51791	67.03	20	1.899
L. infantum	16915	30.81	15	2.156
L. major	8527	21.19	16	2.340
L. mexicana	8254	21.91	8	2.278
L. tropica	424	27.71	8	1.817

Considering the values k_{min} are relatively small when compared to the average degree of the corresponding network, the networks seem to have similar properties to scale-free networks. However, it is noteworthy that a power-law distribution is not a good fit for any of the networks which is consistent with the studies that PPINs may not follow a power-law degree distribution [14, 33, 40] even if they have similar properties [31]. Barabási–Albert models are often used as a graph generator for scale-free networks [7]. However, in this application the $\gamma \approx 3$ does not correspond well with the distribution of the actual graphs as seen in Table 2. A hyperbolic graph generator within the Python package NetworKit [50] is utilised as it allows arbitrary average degree and γ values for $\gamma > 2$. To account for different types of PPINs, different combinations of values are chosen for the generators by considering

$$n \in \{400, 8000, 25000\}, \quad \bar{d} \in \{10, 20, 30, 60\}, \quad \gamma \in \{2.05, 2.3, 2.45\}$$

with all combinations of $n \times \bar{d} \times \gamma$ being benchmarked with the procedure represented in Figure 9. A common seed was set for the graph generator to ensure reproducibility of the random graphs.

4.1 MCP benchmarking summary

This section discusses the main results of benchmarking for the MCP. The results are summarised in two ways. First, the results are categorised based on the different solvers used where the compared categories are Gurobi, SCIP and Algorithms 1-5 proposed throughout Section 3. Second, the results are categorised based on the different methods used by the solvers where the compared categories are Gurobi ILP, SCIP ILP, and all centrality methods defined in Section 2.3.

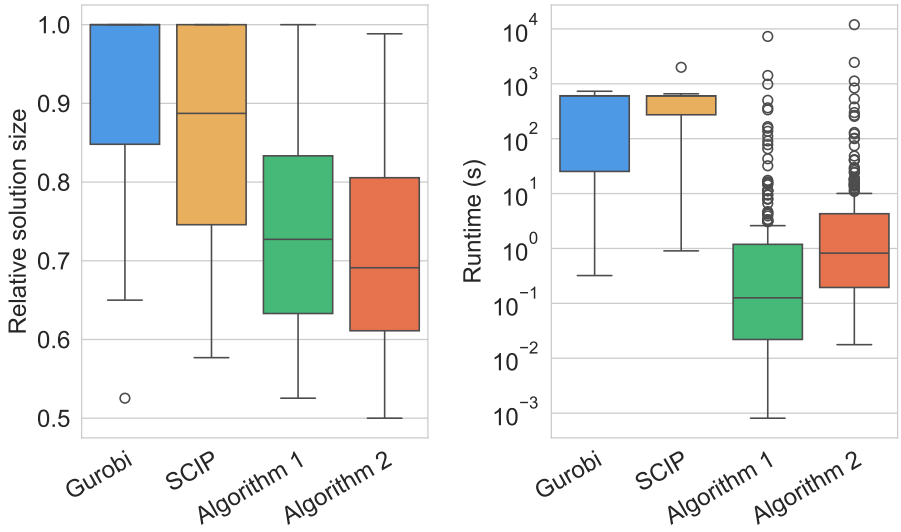


Figure 10: Boxplot summary of MCP benchmarking on DIMACS graphs categorised based on the solver type between the ILP formulation solvers (Gurobi and SCIP), and Algorithms 1 and 2. The different sets of solvers are compared according to normalised relative solution size (left graph) and runtime (right graph). For the solvers, higher relative solution size and lower runtime are considered to be better.

The results of the MCP solvers on the DIMACS set are presented in Figures 10-12. In the relative solution size, Gurobi performs the best on average, while SCIP performs slightly worse. The heuristic algorithms generally perform worse than the ILP formulations in terms of solution size, while they are much faster. However, both heuristic algorithms exhibit large sets of outliers in the runtime boxplot. This is clearly caused one of the centrality methods, as can be seen in Figure 12, where betweenness centrality is much slower than the other methods, reaching runtimes of almost 3 hours in some cases. The specific reason for this is unknown, but the large scale nature of the graphs and the requirement of finding all possible shortest paths between all vertices are likely the reasons to the long runtimes.

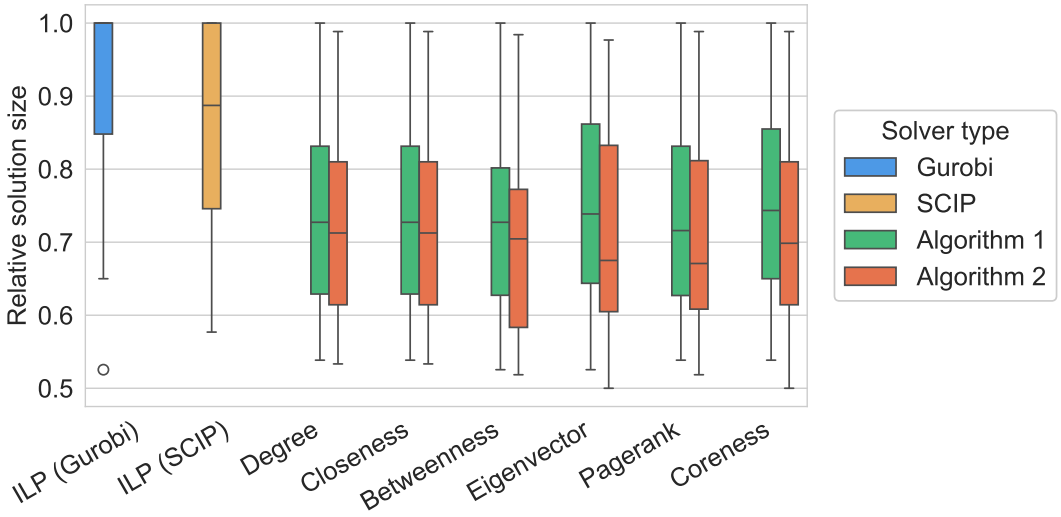


Figure 11: Boxplot of relative solution sizes obtained in MCP benchmarking on DIMACS graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 1 and 2 are separated for each of the centrality methods. For the methods, higher relative solution size is considered to be better.

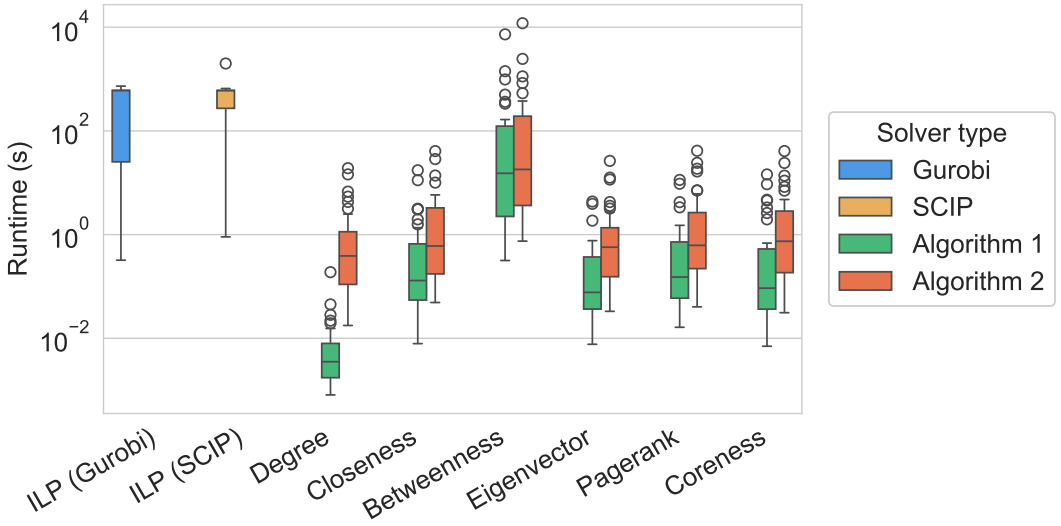


Figure 12: Boxplot of runtimes obtained in MCP benchmarking on DIMACS graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 1 and 2 are separated for each of the centrality methods. For the methods, lower runtime is considered to be better.

Considering Figures 11 and 12 more generally, Algorithm 1 seems to narrowly outperform Algorithm 2 since the relative solution size is better for each quartile of the boxplots. At the same time, the runtimes are on average slightly faster for Algorithm 1. While betweenness centrality is a significant exception in terms of runtime, the other centrality methods are consistent across both algorithms, producing solutions of similar size in a similar time. Degree centrality is slightly faster than the other centrality algorithms.

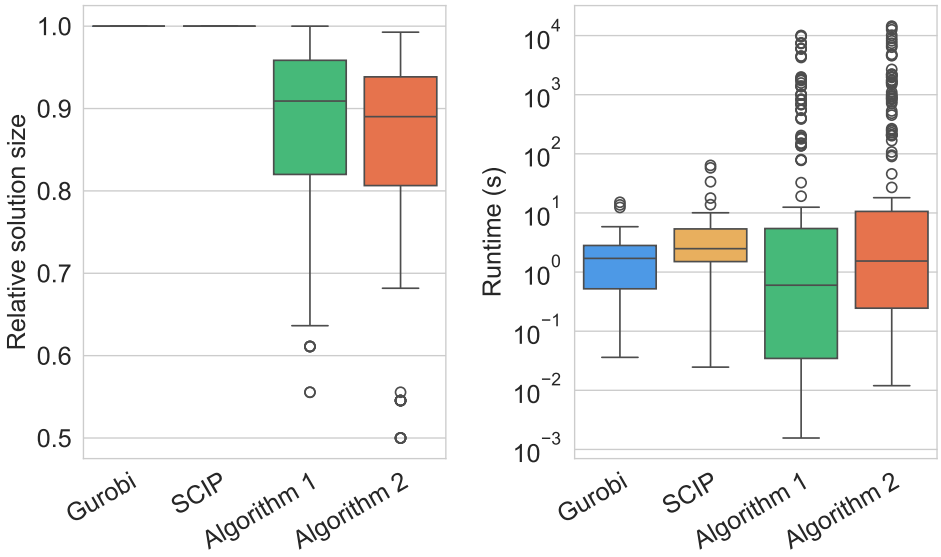


Figure 13: Boxplot summary of MCP benchmarking on random scale-free graphs categorised based on the solver type between the ILP formulation solvers (Gurobi and SCIP), and Algorithms 1 and 2. The different sets of solvers were compared with respect to normalised relative solution size (left graph) and runtime (right graph). For the solvers higher relative solution size and lower runtime are considered to be better. The ILP solvers consistently find the maximum cliques and consequently the Gurobi and SCIP boxplots of relative solution size collapse to lines near relative size of 1.

For the benchmarking on random scale-free graphs (Figures 13-15), the ILP formulations always find the maximum cliques from the graphs. This causes the boxplots to be portrayed as black lines. Similarly to the DIMACS set, SCIP is on average slightly slower than Gurobi. The heuristic algorithms perform worse than the ILP formulations with respect to the relative solution size, although the runtimes are comparable between the heuristics and the ILP formulations in Figure 13.

When all of the centrality methods are compared in terms of relative solution size and runtime (Figures 14 and 15), it is clear that betweenness and closeness have significantly higher runtimes than any of the other methods. This causes the runtime boxplots to be skewed in Figure 13 while the other centrality methods are on average faster than the ILP formulations. Algorithm 1 is slightly better when compared to Algorithm 2 in solution size while being the fastest method out of all considered approaches, if betweenness and closeness centralities are ignored.

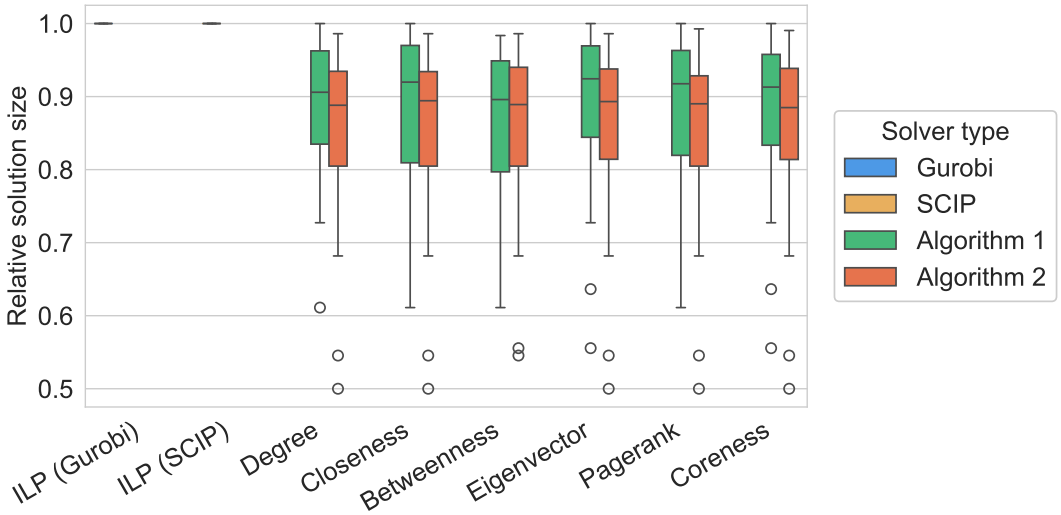


Figure 14: Boxplot of relative solution sizes obtained in MCP benchmarking on random scale-free graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 1 and 2 are separated for each of the centrality methods. For the methods, higher relative solution size is considered to be better. The ILP solvers consistently find the maximum cliques and consequently the Gurobi and SCIP boxplots collapse to lines.

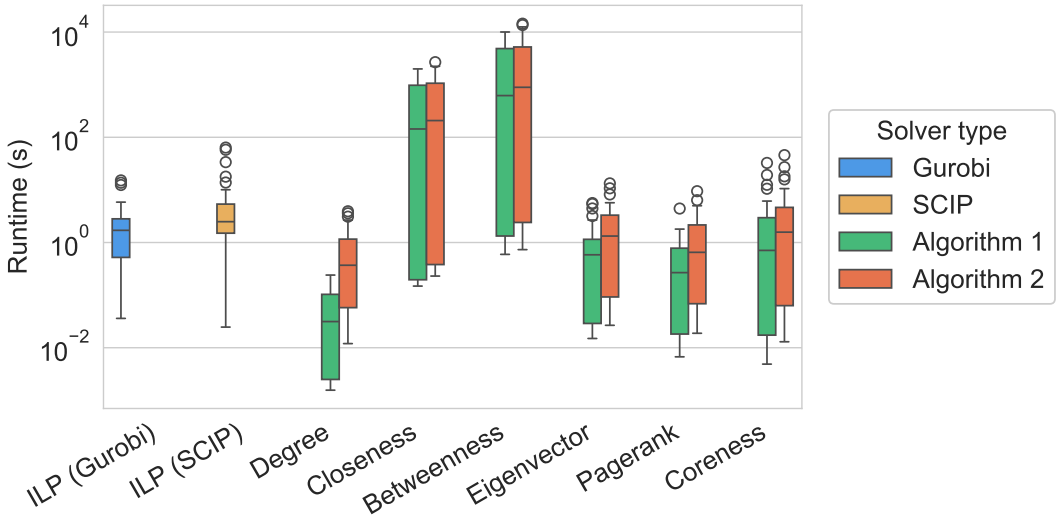


Figure 15: Boxplot of runtimes obtained in MCP benchmarking on random scale-free graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 1 and 2 are separated for each of the centrality methods. For the methods, lower runtime is considered to be better.

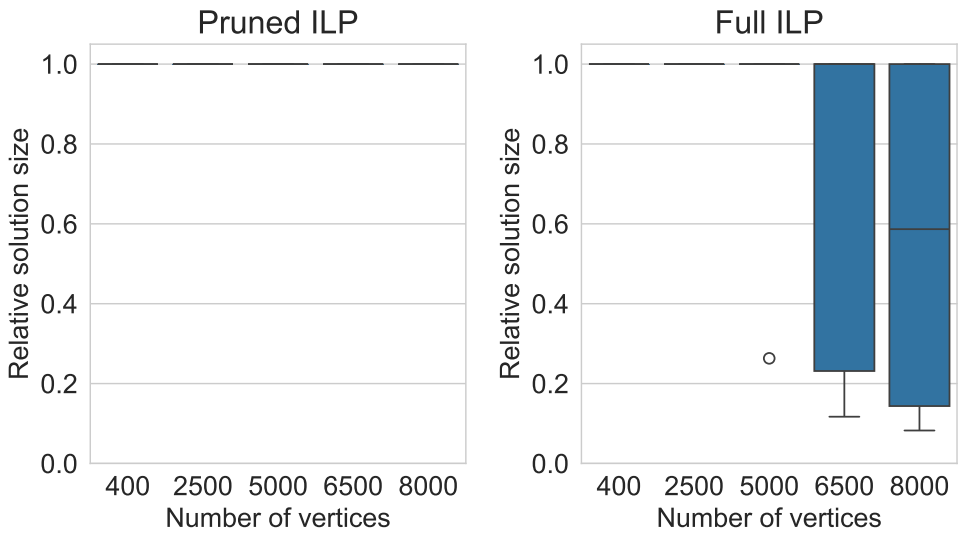


Figure 16: Boxplot comparison of relative solution size on pruned (left graph) and full (right graph) ILP formulations. Solving the MCP was done with Gurobi on instances with different number of vertices which are indicated on x-axis. Additionally, an optimisation time limit of 600 seconds is imposed on the solver. The lines around the solution size value of 1 imply that the solver consistently converges to the optimal value in the optimisation time limit. The pruned ILP formulation finds the maximum clique in all instances and the full ILP formulation is unable to find the maximum cliques on significant amount of instances with graphs that have 6500 or 8000 vertices.

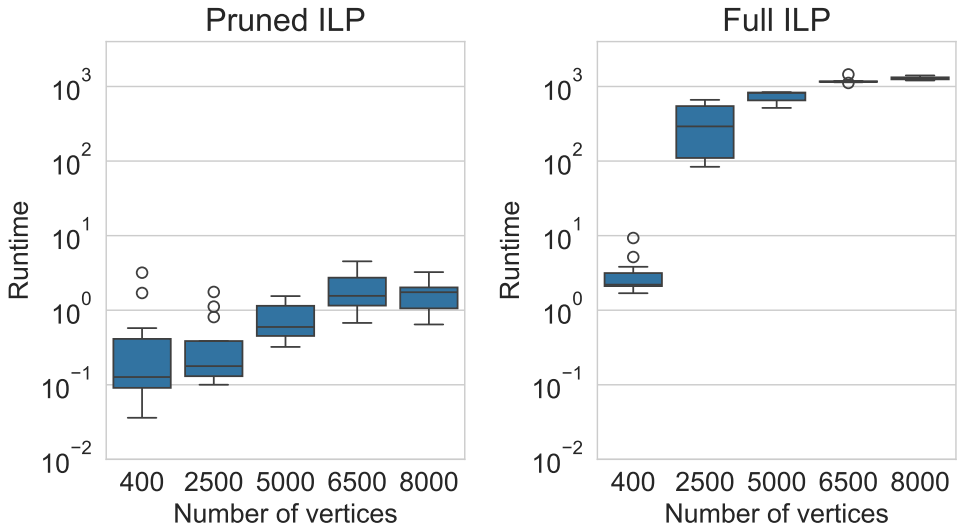


Figure 17: Boxplot comparison of runtime on pruned (left graph) and full (right graph) ILP formulations. Solving the MCP was done with Gurobi on instances with different number of vertices which are indicated on x-axis. Additionally, an optimisation time limit of 600 seconds is imposed on the solver. The runtime increases quickly when the number of vertices in the graphs increases. The time limit is consistently hit on most instances with graphs that have 6500 or 8000 vertices.

In general, the more sparse graphs seem to have a lot more approachable structures for the pruned ILP procedure, and thus it should be noted that ILP solvers rely on pruning to solve the problems consistently. To highlight this effect, full ILP formulations were tested on sets of random scale-free graphs which were formulated with combinations $n \times \bar{d} \times \gamma$ as described before but with different values for n. During the investigation, it was observed that any larger graphs, starting from the graphs that have 6500 vertices, consistently reach the optimisation time limit without converging (Figures 16 and 17). This is in line with the NP-complete nature of the MCP and highlights the necessity of pruning.

4.2 MVCP benchmarking summary

This section discusses the main results of benchmarking for the MVCP. The results of benchmarking on DIMACS graphs are presented in Figures 18-20. The heuristic algorithms consistently perform faster than ILP formulations even with the outliers in runtime caused by betweenness and closeness centralities.

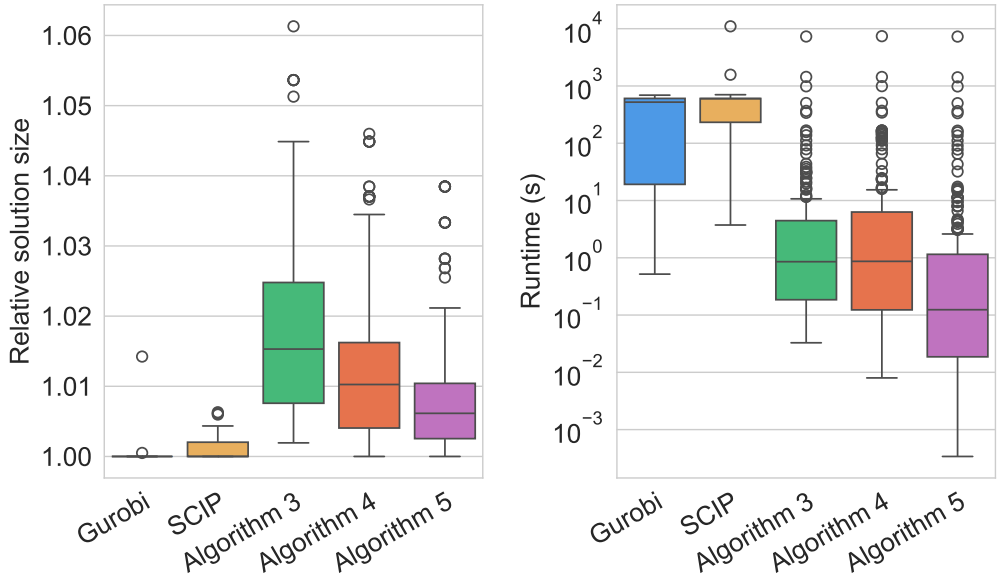


Figure 18: Boxplot summary of MCP benchmarking on DIMACS graphs categorised based on the solver type between the ILP formulation solvers (Gurobi and SCIP), and Algorithms 3-5. The different sets of solvers were compared with respect to normalised relative solution size (left graph) and runtime (right graph). For the solvers lower relative solution size and lower runtime are considered to be better.

The relative solution sizes of heuristics are slightly worse than those of the best solutions generated by Gurobi. In a large number of instances, both ILP formulations exceeded the optimisation time limit, resulting in median times of around 600 seconds for both ILP solvers. This is also reflected in a notable number of problem instances not converging to an optimal solution, as presented in Table 1.

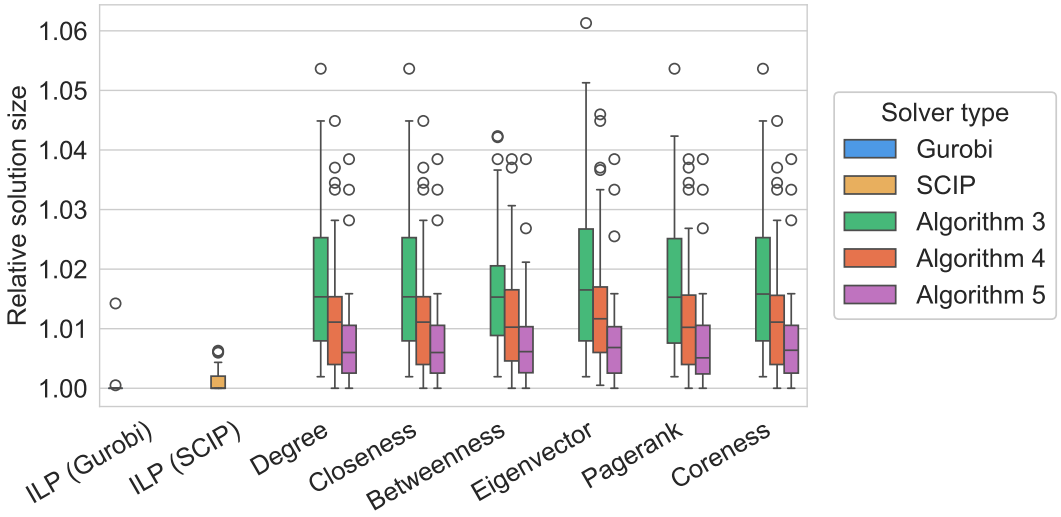


Figure 19: Boxplot of relative solution sizes obtained in MCP benchmarking on DIMACS graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 3-5 are separated for each of the centrality methods. For the methods, lower relative solution size is considered to be better.

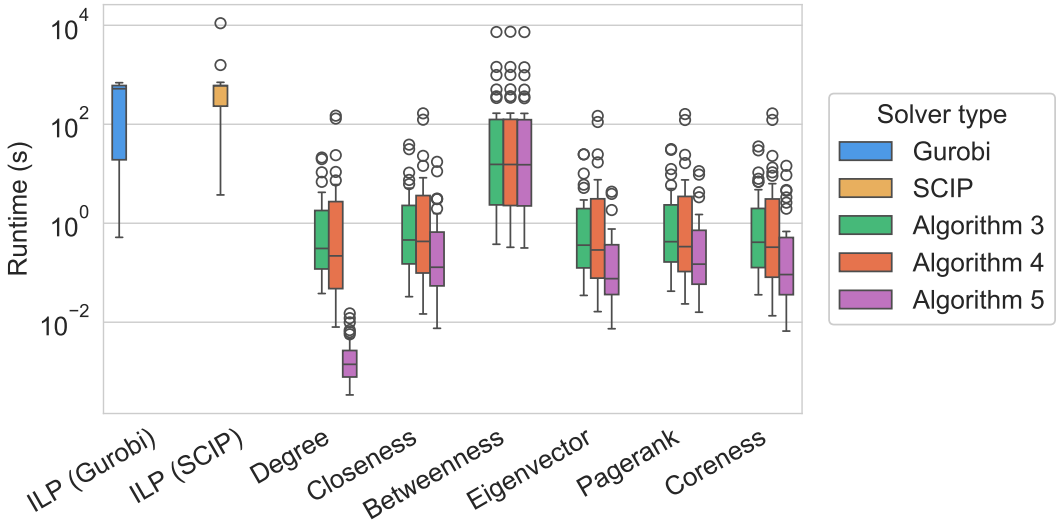


Figure 20: Boxplot of runtimes obtained in MCP benchmarking on DIMACS graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 3-5 are separated for each of the centrality methods. For the methods, lower runtime is considered to be better.

The ILP solvers still produce consistently better solutions than the heuristic algorithms. Among the heuristics, Algorithm 5 clearly outperforms the other heuristics in both relative solution size and runtime. If the different centrality methods are compared, it is clear that the algorithms produce very consistent results, as seen in Figures 19 and 20. This implies that the algorithms are the major contributing factor to the runtime and vertex cover size in most cases. The first notable outlier here is betweenness centrality, which is again much slower than the other methods. The second outlier is the degree centrality utilising Algorithm 5, which seems to be comparatively much faster than the other algorithms utilising degree centrality.

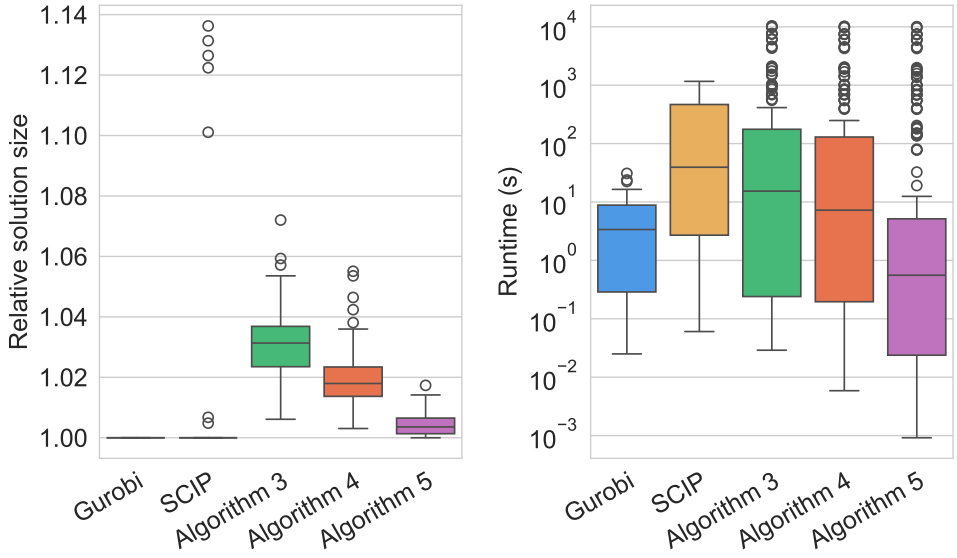


Figure 21: Boxplot summary of MCP benchmarking on random scale-free graphs categorised based on the solver type between the ILP formulation solvers (Gurobi and SCIP), and Algorithms 3-5. The different sets of solvers were compared with respect to normalised relative solution size (left graph) and runtime (right graph). For the solvers lower relative solution size and lower runtime are considered to be better.

Benchmarking results for random scale-free are presented in Figures 21-23. The results are comparable to the previous results as the ILP formulations find exact solutions outside of couple outliers for SCIP. Gurobi is able to solve the problems in around a second while SCIP has longer runtimes while hitting the optimisation time limit which also causes the outliers in relative solution size. For the heuristics, Algorithm 3 performs slightly worse on the random graphs when compared to the DIMACS benchmarking as the median vertex cover relative size is slightly above 1.025, while the median was closer to 1.015 for the DIMACS graphs. A similar effect is observed for Algorithm 4. Algorithm 5 is again the fastest heuristic and almost as good considering solution size as the ILP formulations.

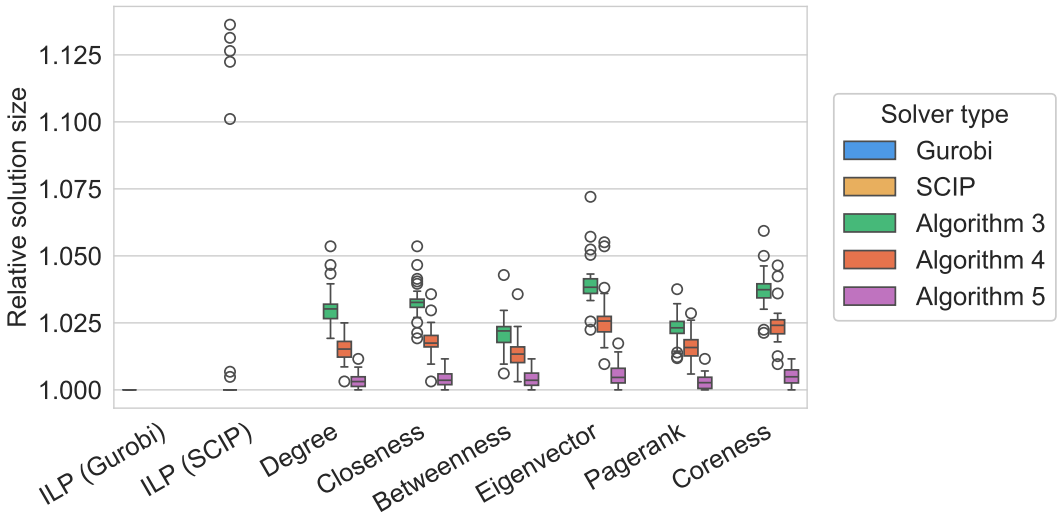


Figure 22: Boxplot of relative solution sizes obtained in MCP benchmarking on random scale-free graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 3-5 are separated for each of the centrality methods. For the methods, lower relative solution size is considered to be better.

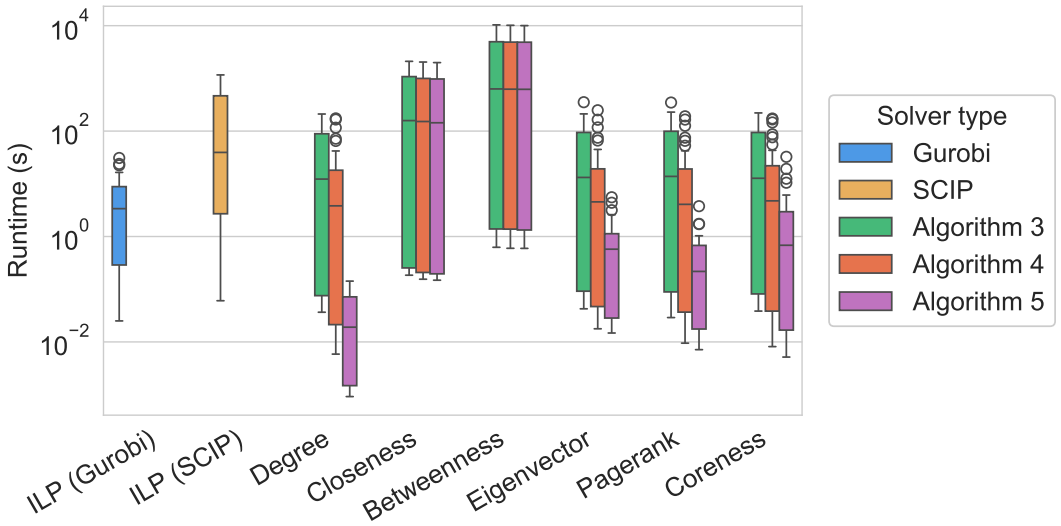


Figure 23: Boxplot of runtimes obtained in MCP benchmarking on random scale-free graphs. Data is grouped by the method type between the ILP formulation solvers (Gurobi and SCIP) and the centrality methods. Algorithms 3-5 are separated for each of the centrality methods. For the methods, lower runtime is considered to be better.

SCIP also seems to hit the optimisation time limit in some outlier cases, leading to some of the vertex covers found by SCIP being notably larger than any other solution. Considering the runtimes between the methods, it is clear again that closeness and betweenness centralities cause a significant skew in the runtimes, as the slower half of the instances have runtimes of at least 100 seconds for the heuristics. Not considering closeness and betweenness centralities, Algorithms 3 and 4 seem to contribute more to the runtime when compared to the centrality calculation. On the other hand, Algorithm 5 seems to be faster than the centrality method calculations such that the different centrality methods significantly affect the runtimes.

5 Experimental evaluation

This section discusses the formulation and quality of the PPINs, and all the experimental results from running the chosen solvers and heuristics on the datasets discussed in Section 5.1. The full procedure for processing the PPINs and running the ILP formulations and heuristics is represented in Figure 24.

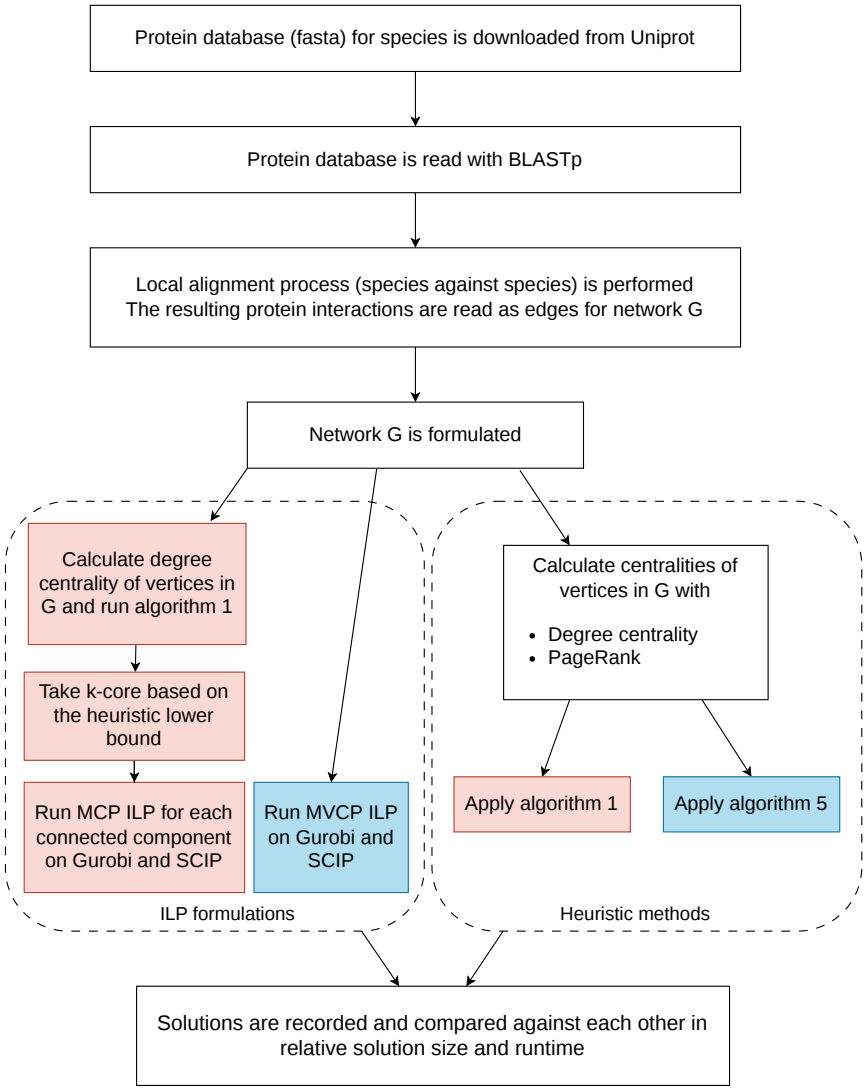


Figure 24: Flowchart of the full process for vital protein identification studied, starting from the FASTA files of species' proteomes downloaded from UniProt. Steps related to cliques and vertex covers are highlighted in red and blue, respectively.

The methods that are studied further are the ILP formulations, as well as Algorithms 1 and 5 paired with degree centrality and PageRank. ILP formulations are chosen because they are straightforward implementations of exact algorithms that also serve as a reference for the other methods being compared. Algorithms 1 and 5 performed consistently the best in benchmarking by producing good heuristic solutions to the corresponding optimisation problems with lower runtimes. Degree centrality and PageRank both perform well with the aforementioned algorithms. Coreness and eigenvector centralities would also be reasonable options, but are omitted here due to a slightly worse performance during benchmarking.

5.1 PPIN formulation

To build the PPINs as discussed initially in Section 2.1, the protein databases for specific species of genus *Leishmania* are downloaded from the UniProt Knowledgebase (abbreviated often as UniProtKB, abbreviated as UniProt in this thesis) [24] which hosts a manually reviewed protein set of over 570 000 proteins called Swiss-Prot and a larger unreviewed protein set of over 200 billion proteins called TrEMBL. The studied databases were downloaded on July 16, 2025, for the *L.* species *amazonensis*, *tropica*, *infantum*, and *donovani*, and on September 24, 2025, for the remaining species. For initial consideration and studying of the protein interactions of the species, basic local alignment search tools (BLAST) [3] will be used to find connections between proteins. For example, possible tools for this kind of alignment procedure include traditional protein-protein BLAST programs such as BLASTp and newer clustering-based programs such as DIAMOND [16].

In the context of this thesis, BLASTp is applied to FASTA files, which are text-based amino acid sequences that in this case represent the full set of proteins expressed by an organism, also called proteomes [49]. These protein sequences are first entered into a database, after which the file is run against the same database to find regions of local similarity between protein sequences, which are used to identify pairs of proteins that statistically could have a significant chance of interacting with one another. This process records the values of 12 fields described in Table 3. Among these fields, sequence identity, length, mismatches, gap openings, e-value, and bit score are considered important values for assessing the significance of protein similarity.

After running BLASTp, a list of protein-protein pairs is generated. The proteins of the species are used as the vertices, and the pairs are used to form edges between the corresponding vertices. The resulting networks (Table 4) are all relatively sparse, although *L. amazonensis* and *L. tropica* are smaller and slightly more dense than the networks for the other species. It is noteworthy that the entire network generation procedure for the larger files can take multiple days, even on high-performance computers. The most significant factor contributing to the long runtime of the whole process is aligning the FASTA file. For example, the alignment process for *L. donovani* took around 32 hours on the setup, which is used for all of the calculations in this thesis. More efficient alignment programs, such as DIAMOND, are often considered good alternatives, as they can speed up the process by orders of magnitude [15, 16].

Table 3: BLAST tabular format, which is given by default as the output of BLASTp, and explanations of the format parameters. Explanations are initially augmented from [9] with additional details from [26].

Field	**Explanation**
Query accession	ID of the protein used as the search query to the database.
Target accession	ID of the protein the query was aligned against.
Sequence identity	The percentage of identical amino acid residues between the query and the target proteins.
Length	The length of the local alignment, or overlapping sequence. This includes matching, mismatching and gap positions between the query and the target.
Mismatches	The number of non-identical amino acid residues aligned. These can be interpreted for example as point mutations.
Gap openings	The number of gap openings. This can be interpreted as insertion or deletion mutations.
Query start	The first index of the alignment in the query.
Query end	The last index of the alignment in the query.
Target start	The first index of the alignment in the target.
Target end	The last index of the alignment in the target.
e-value	Represents the number of alignments of similar or better quality that could be found randomly. A lower e-value implies a more significant alignment.
Bit score	Represents the overall quality of the alignment. A higher bit score corresponds to higher similarity between sequences.

Table 4: Data of the networks formulated for the *Leishmania* species through the generation procedure described in Figure 24. $|V|$ and $|E|$ correspond to the number of vertices and edge respectively, ρ is the density of the network and $\bar{d}$ is the average vertex degree of the network.

Network	$\lvert V\rvert$	$\lvert E\rvert$	ρ	$\bar{d}$	Max degree
L. amazonensis	229	1050	0.040221	9.17	32
L. braziliensis	16451	257912	0.001906	31.35	467
L. donovani	51791	1735808	0.001294	67.03	1259
L. infantum	16915	260566	0.001822	30.81	472
L. major	8527	90357	0.002486	21.19	260
L. mexicana	8254	90436	0.002655	21.91	281
L. tropica	424	5875	0.065514	27.71	98

5.1.1 Analysis of PPIN quality

As the researched networks are built directly from the output received from the BLAST software, the quality of the data should be examined. The initial quality is assessed by considering two values of the output format described in Table 3: e-values and bit scores. Both values quantify the statistical significance of the similarity between proteins in the protein pairs identified by BLAST. Additionally, this directly quantifies the quality of the PPINs examined as these protein pairs are used as the edges of the networks.

The e-value, or expect value, estimates the number of matches with a score equal to or better than the observed alignment that would be expected to occur by chance in an equivalently sized database search [48]. In this thesis, the proteomes of the species are used as the search databases and a level of 0.05 for e-values is often considered the default significance threshold level used in programs for protein interactions. A threshold of 10^{-5} is sometimes also used to identify homologous hits, indicating significant similarity between proteins [21].

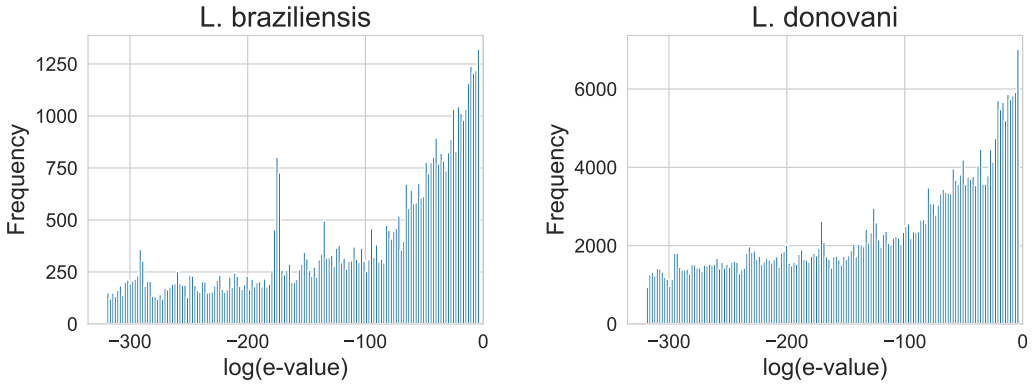


Figure 25: Distribution of log base 10 values of e-values of protein pairs for species *L. braziliensis* (on left) and *L. donovani* (on right). On the log-scaled x-axis, -2 is the limit for significance, and -5 is considered an indicator of an almost certain significant hit, with anything smaller being more significant.

The distributions of e-values within the protein networks (Figure 25) suggest that a large number of the found connections have statistical significance. Only a small portion of below 3% of the protein connections are found above the 10^{-5} level for most of the networks. Some networks, namely the smaller ones, have few protein pairs above 10^{-5}. Thus, the produced networks appear promising, as the reported significance of many pairs is well below the generally accepted limits. However, while e-values have the benefit of, for example, having a meaning for extremely small values when compared to the single-level statistical significance level of p-values [29], e-values also have issues, especially when it comes to being too conservative and liberal in different cases [43] when it comes to protein-protein sequence analysis.

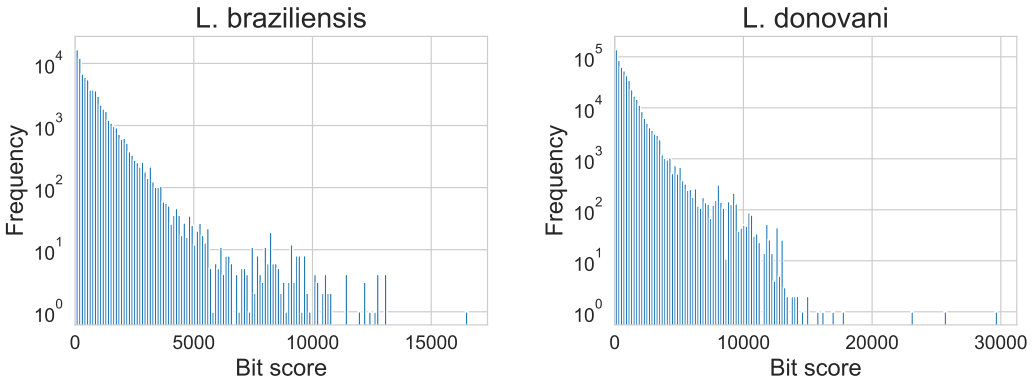


Figure 26: Distribution of bit scores of protein pairs for species *L. braziliensis* (on left) and *L. donovani* (on right). A score of 50 or above is considered to be the reference level of almost certain significance.

The bit score is an alternative, more standardised method for assessing the significance of a score, with 50 considered "almost always significant" for protein-protein interactions [52], which is similar to 10^{-5} for e-values. Based on the distributions of bit scores (Figure 26), a large majority of the predicted connections would be considered significant. Similarly to the e-value threshold of 10^{-5}, only about 3% of protein pairs have a bit score below 50. Thus, the networks formulated appear to capture only a small fraction of the predicted interactions, which are considered less significant, suggesting a reasonably good model of the real interactions.

The research here focuses on BLASTp because it is a more established and consistent program. Attempts were made to utilise DIAMOND alongside BLASTp, but there were some unexplained issues with the networks generated by DIAMOND. The e-values and bit scores for the DIAMOND pairs were still significant, and Further research on the faster sequence alignment methods and networks formulated based on them should be considered in the future. The massive speed-ups in PPIN formulation would significantly reduce the duration of the entire process of vital protein identification.

5.2 Effects of pruning on ILP formulations

To begin the consideration of ILP solvers, the efficiency of pruning with k-cores is considered. The formulated networks for the species PPINs are analysed to identify the shell indices of the vertices. From this, all k-cores are easily formulated. For all the different k-cores, an MCP ILP is formulated and solved. The results for *L. mexicana* are presented in Figure 27 and the results for *L. donovani* are presented in Figures 28 and 29. The remaining results are attached to the appendix as Figures A1-A5, as they are similar to those observed for *L. mexicana*. For the MCP ILP formulations, the pruning strategy seems effective, provided the pruned network is not too large to solve.

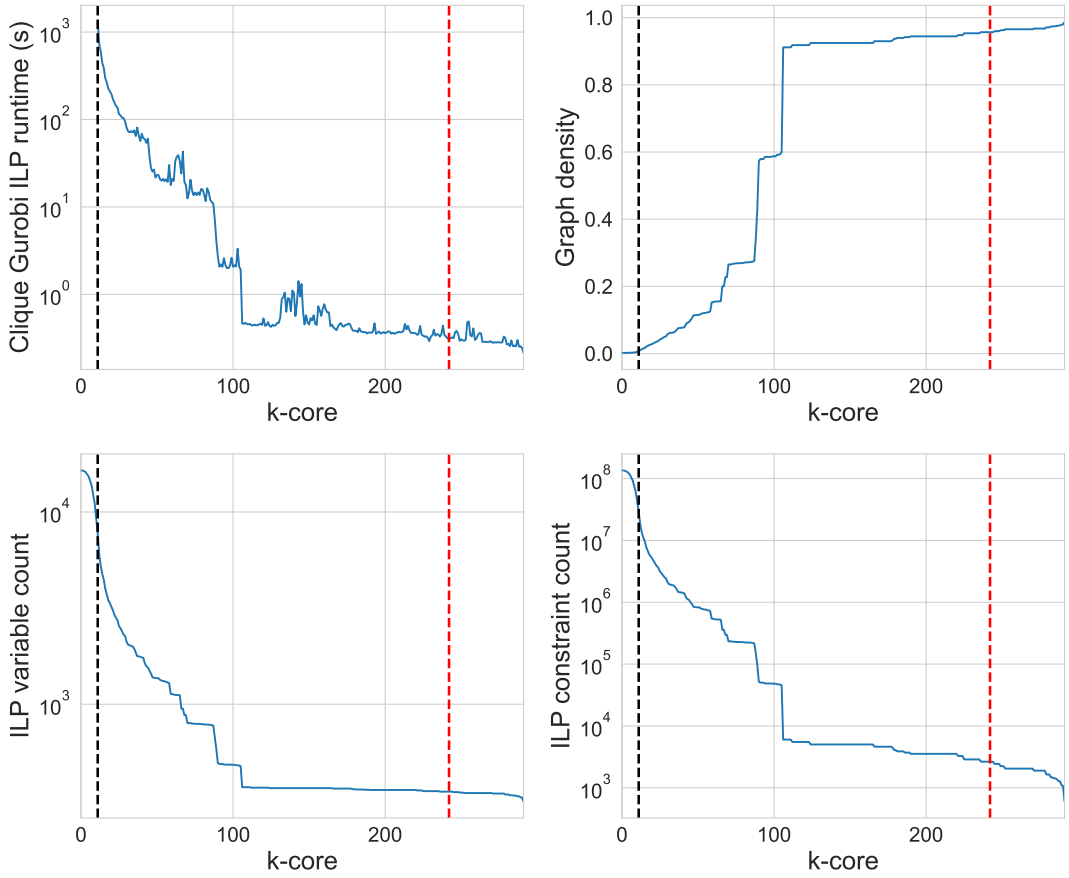


Figure 27: Results of pruning with k-cores on the PPIN of *L. braziliensis*. The maximum nonempty k-core in the network is the 291-core. The ILP solver procedure proposed in Figure 24 running Algorithm 1 and degree centrality finds a clique of size 243 as a lower bound for the MCP. $k = 242$ is denoted by the red vertical dashed line, which corresponds to the k-core on which the ILP formulation is run in the actual procedure outlined in Figure 24. ILP formulations run out of memory for $k \leq 11$, as indicated by the black dashed line.

In sufficiently easy cases, the k values corresponding to the k-cores seem to follow an inverse exponential relation when compared to ILP solution runtime and the size of the ILP formulation. Effectively, the pruning strategy turns the MCP on a large sparse network to a MCP on a smaller network with higher density. In the example described in Figure 27, network size is reduced by 98% during pruning when compared to the original network size. The constraint count is reduced by around 99.995% compared to the original network, and the Gurobi ILP runtime to convergence is around 10 000 times faster than on the 12-core. The ILP formulation could not be formulated for the k-cores with $k \leq 11$ due to memory constraints.

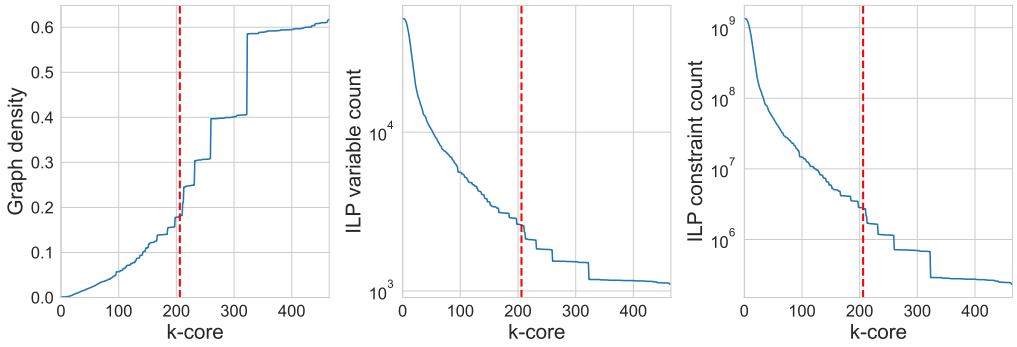


Figure 28: Effects of pruning with k-cores on the PPIN of *L. donovani*. The maximum nonempty k-core in the network is the 465-core. The ILP solver procedure proposed in Figure 24 runs Algorithm 1 and degree centrality, finding a clique of size 207 as a lower bound for the MCP. $k = 206$ is denoted by the red vertical dashed line, which corresponds to the k-core on which the ILP formulation is run in the actual procedure outlined in Figure 24.

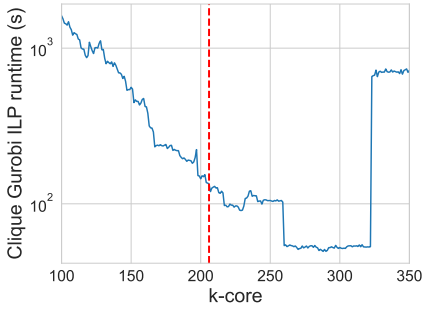


Figure 29: Gurobi clique ILP formulation runtimes until convergence for k-cores with $k \in [100, 350]$ for the PPIN of *L. donovani*. The ILP solver procedure proposed in Figure 24 runs Algorithm 1 and degree centrality, finding a clique of size 207 as a lower bound for the MCP. $k = 206$ is denoted by the red vertical dashed line, which corresponds to the k-core on which the ILP formulation is run in the actual procedure. Other k-core values are omitted due to the long runtimes.

The NP-completeness of the problem arises because the pruned network remains relatively hard to solve in the context of the MCP. This can be seen from the results of running Gurobi ILP formulation on the network of *L. donovani* (Figures 28 and 29). Even if the ILP formulation routine would not converge within the optimisation time limit, there is still a major improvement in the problem, given the reduction in memory usage. This often allows solvers to at least attempt solving the problem until the timeout limit, rather than running out of memory during the formulation of the optimisation problem or the ILP-solving process itself. This is evident from the constraint count still being reduced by around 99.9% and the variable count being reduced by around 95% compared to the original network. It should be noted that *L. donovani* seems to exhibit different characteristics compared to the other PPINs.

For example, network density increases only to 0.6 in k-cores with large k values, whereas the k-cores in the other graphs can reach density levels close to 1. Additionally, as observed in Figure 29, something happens around $k = 323$, where the MCP ILP formulation becomes harder to solve, leading to a sudden increase in runtime. When this effect is investigated further, it becomes evident that the entire network's maximum clique is removed because some of its vertices have a shell index of only 323. As some of the vertices of the maximum clique are removed, this also changes the maximum clique in the k-core when $k \geq 323$. This issue is never encountered with the proposed pruning strategy due to Theorems 3.1 and 3.2, which guarantee that the maximum clique of the original network is contained within the pruned network. On the other hand, this effect also highlights the importance of not just taking the largest nonempty k-cores, as they may remove important information from the network.

Vertex cover ILP formulations for sparse networks require less memory to define since there are only $|E(G)|$ constraints. Finding exact optima of an optimisation problem with $|V(G)|$ decision variables with respect to these constraints is still computationally intense in the scale of the networks discussed before, as there are no similar tricks to reduce the solution space, and most vertices are chosen for the vertex cover. For example, applying the strategy of automatically adding all neighbours of leaf vertices to a vertex cover only reduced the variable count of the ILP formulation by at most 1% for *L. amazonensis*, while the other networks had their variable count reduced by only around 0.1%. Similarly, only around 2% of the constraints were removed in most cases, while in one case the constraint count was reduced by only 0.2%. In the context of NP-complete problems, such changes do not significantly affect runtime or memory requirements for ILP formulations.

5.3 ILP and heuristic performance on PPINs

The results of running the procedures described in Figure 24 are presented in Tables 5 and 6 for the MCP and MVCP, respectively.

Table 5: Results of running procedures for the MCP pruned ILP formulations and Algorithm 1 paired with degree centrality and PageRank. Relative size is calculated based on the largest clique found by the methods. Methods that have not converged by the optimisation time limit are denoted with an asterisk next to the runtime. The best solution sizes and runtimes for each network are bolded.

Network	Solver	Method	Clique size	Relative size	Runtime (s)
L. amazonensis	Gurobi	Pruned ILP	**13**	**1**	0.0060
L. amazonensis	SCIP	Pruned ILP	**13**	**1**	0.0136
L. amazonensis	Alg 1	Degree	**13**	**1**	**0.0015**
L. amazonensis	Alg 1	PageRank	3	0.2308	0.0042
L. braziliensis	Gurobi	Pruned ILP	**250**	**1**	2.4202
L. braziliensis	SCIP	Pruned ILP	**250**	**1**	2.9664
L. braziliensis	Alg 1	Degree	243	0.972	**0.0933**
L. braziliensis	Alg 1	PageRank	19	0.076	1.2048
L. donovani	Gurobi	Pruned ILP	**320**	**1**	173.0183
L. donovani	SCIP	Pruned ILP	202	0.6313	*1091.8912
L. donovani	Alg 1	Degree	207	0.6469	**0.6269**
L. donovani	Alg 1	PageRank	39	0.1219	4.2676
L. infantum	Gurobi	Pruned ILP	**259**	**1**	2.1148
L. infantum	SCIP	Pruned ILP	**259**	**1**	2.6443
L. infantum	Alg 1	Degree	251	0.9691	**0.0552**
L. infantum	Alg 1	PageRank	15	0.0579	0.5843
L. major	Gurobi	Pruned ILP	**136**	**1**	0.5998
L. major	SCIP	Pruned ILP	**136**	**1**	0.7982
L. major	Alg 1	Degree	133	0.9779	**0.0216**
L. major	Alg 1	PageRank	10	0.0735	0.1976
L. mexicana	Gurobi	Pruned ILP	**159**	**1**	0.9497
L. mexicana	SCIP	Pruned ILP	**159**	**1**	0.8972
L. mexicana	Alg 1	Degree	153	0.9623	**0.0245**
L. mexicana	Alg 1	PageRank	13	0.0818	0.2073
L. tropica	Gurobi	Pruned ILP	**73**	**1**	0.0364
L. tropica	SCIP	Pruned ILP	**73**	**1**	0.0410
L. tropica	Alg 1	Degree	**73**	**1**	**0.0015**
L. tropica	Alg 1	PageRank	14	0.1918	0.0157

Table 6: Results of running procedures for the MVCP ILP formulations and Algorithm 5 paired with degree centrality and PageRank. Relative size is calculated based on the smallest vertex cover found by the methods. Methods that have not converged by the optimisation time limit are denoted with an asterisk next to the runtime. These solutions are not guaranteed to be the minimal vertex covers, as most ILP formulations do not converge within the optimisation time limit. Best solution sizes and runtimes for each network are bolded.

Network	Solver	Method	Vertex cover size	Relative size	Runtime (s)
L. amazonensis	Gurobi	ILP	**152**	**1**	0.3630
L. amazonensis	SCIP	ILP	**152**	**1**	2.5414
L. amazonensis	Alg 5	Degree	158	1.0395	**0.0006**
L. amazonensis	Alg 5	PageRank	158	1.0395	0.0030
L. braziliensis	Gurobi	ILP	**12893**	**1**	*603.6138
L. braziliensis	SCIP	ILP	16451	1.2760	*861.7983
L. braziliensis	Alg 5	Degree	13094	1.0156	**0.0454**
L. braziliensis	Alg 5	PageRank	13093	1.0156	0.9631
L. donovani	Gurobi	ILP	**45703**	**1**	*629.5616
L. donovani	SCIP	ILP	46579	1.0192	*632.2983
L. donovani	Alg 5	Degree	45823	1.0026	**0.2019**
L. donovani	Alg 5	PageRank	45821	1.0026	4.0612
L. infantum	Gurobi	ILP	**13423**	**1**	*604.8621
L. infantum	SCIP	ILP	16915	1.2602	*1458.1406
L. infantum	Alg 5	Degree	13605	1.0136	**0.0446**
L. infantum	Alg 5	PageRank	13604	1.0135	0.5705
L. major	Gurobi	ILP	**5825**	**1**	*602.1841
L. major	SCIP	ILP	5960	1.0232	*601.1626
L. major	Alg 5	Degree	6018	1.0331	**0.0176**
L. major	Alg 5	PageRank	5998	1.0297	0.2354
L. mexicana	Gurobi	ILP	**5610**	**1**	*601.5832
L. mexicana	SCIP	ILP	5742	1.0235	*602.0649
L. mexicana	Alg 5	Degree	5812	1.0360	**0.0174**
L. mexicana	Alg 5	PageRank	5795	1.0330	0.2430
L. tropica	Gurobi	ILP	**352**	**1**	0.0956
L. tropica	SCIP	ILP	**352**	**1**	0.2742
L. tropica	Alg 5	Degree	354	1.0057	**0.0007**
L. tropica	Alg 5	PageRank	354	1.0057	0.0130

Generally, the results seem to align with the benchmarking results. The heuristics are faster but result in smaller cliques or larger vertex covers than the ILP formulations. Among the ILP solvers, Gurobi is generally faster and produces higher-quality results than SCIP. Notable deviation from the benchmark is the performance of PageRank with the MCP heuristic. While the benchmarking suggested that both degree centrality and PageRank could produce good results when paired with Algorithm 1, PageRank performs rather poorly on actual PPINs. For the MCP, the ILP solvers are relatively fast in most cases, mainly due to the pruning strategy utilising degree centrality, which is in line with the results presented in Section 5.2. The size of the network *L. donovani* still proves to be a more difficult task in general, since the ILP solvers take significantly longer to converge. SCIP reaches the 600-second optimisation runtime limit while producing a slightly smaller clique than Algorithm 1 paired with degree centrality. However, the clique size of the heuristic is only around 65% of the network's maximum clique size, as found by the Gurobi ILP solver. In the small and more dense networks for species *L. amazonensis* and *L. tropica*, Algorithm 1 with degree centrality also finds a maximum clique in the graph.

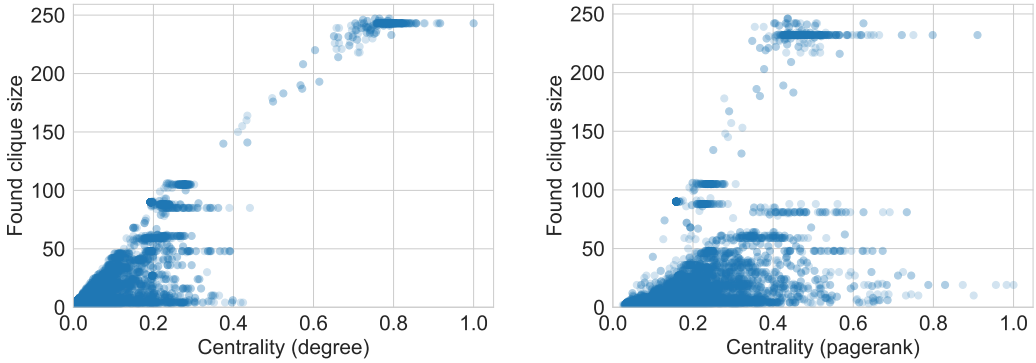


Figure 30: Clique sizes found from the network of *L. braziliensis* when all possible vertices are used as seeds for Algorithm 1 when utilising degree centrality (left graph) and PageRank (right graph). The size of the maximum clique within the network is 250. Degree centrality of the seed vertex is observed to correlate with the found clique size well. The seeds with PageRank values around 0.5 produce the largest cliques while the seed nodes with highest centrality values inconsistently find either small or large cliques.

Considering the results of the clique heuristics on the PPINs, the algorithm is sensitive to the starting seed. For demonstration purposes, all possible vertices were considered as seed vertices for the networks in Table 4 with the results plotted for combinations of Algorithm 1 with the chosen centrality methods. Figures 30 and 31 correspond to *L. braziliensis* and *L. donovani*, while the other results are more similar to those achieved for *L. braziliensis* are attached in the appendix as Figures B1-B5.

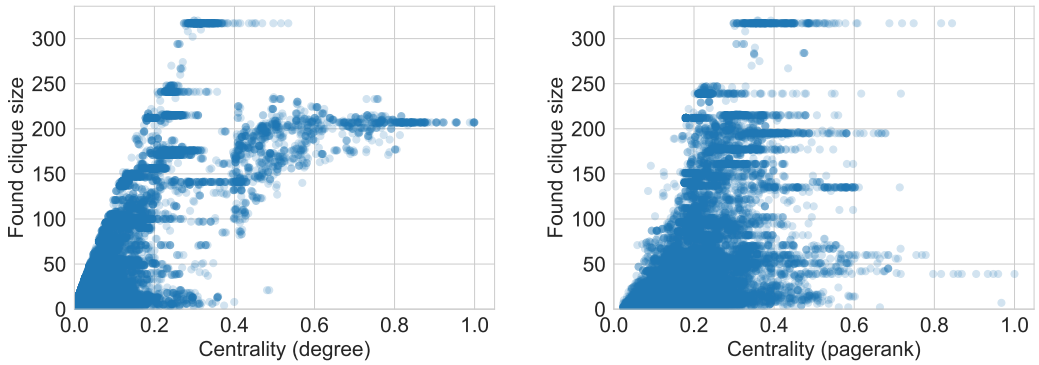


Figure 31: Clique sizes found from the network of *L. donovani* when all possible vertices are used as seeds for Algorithm 1 when utilising degree centrality (left graph) and PageRank (right graph). The size of the maximum clique within the network is 320. Degree centrality of the seed node still correlates with the found clique size but the vertices with highest centrality only find cliques around the size of 207. The seeds with different PageRank values produce inconsistent clique sizes similarly to *L. braziliensis* presented in Figure 30.

In general, the degree centrality of the seed vertex shows a clear positive correlation with the found clique size. For *L. donovani*, the clique sizes achieved with different seed vertices utilising degree centrality highlight the difference between *L. donovani* and the other networks, as the vertices with the highest centrality values only produce cliques of around size 200. In contrast, some vertices with lower degree centrality around 0.4 as seed nodes produce notably larger cliques. A similar effect is not observed in the other networks while utilising degree centrality. The algorithm utilising PageRank values for order produces a lot more inconsistent results. Even if there is a positive correlation between the PageRank value of the seed vertex and the found clique size, it does not consistently produce large cliques with the highest-valued seed vertices. The largest cliques are found by a few seed vertices in the region between 0.2 and 0.6 of the normalised centrality values for all of the studied networks.

For the MVCP, the ILP formulations only converge for the PPINs of *L. amazonensis* and *L. tropica*. In other cases, the optimisation time limit is reached without convergence. At the time limit, Gurobi finds the smallest vertex cover among the methods for all of the considered networks, but these are only the best upper bounds for the minimum vertex covers. PageRank and degree centrality both perform well in the MVCP heuristic for the networks, and Algorithm 5 seems to produce slightly smaller vertex covers with PageRank than with degree centrality. Both methods produce vertex covers that are around $1\% - 4\%$ larger than the best upper bounds found by Gurobi.

The heuristics take a fraction of the time it takes the ILP solvers to converge. For the PPINs of *L. infantum* and *L. donovani* SCIP was unable to find better vertex covers within the optimisation time limit when compared to the vertex covers produced by the heuristics. Since Gurobi outperforms the other MVCP methods, the solver convergence is analysed for cases where Gurobi is unable to reach convergence in the optimisation time limit. The evolution of the solution bounds for *L. donovani* and *L. braziliensis*

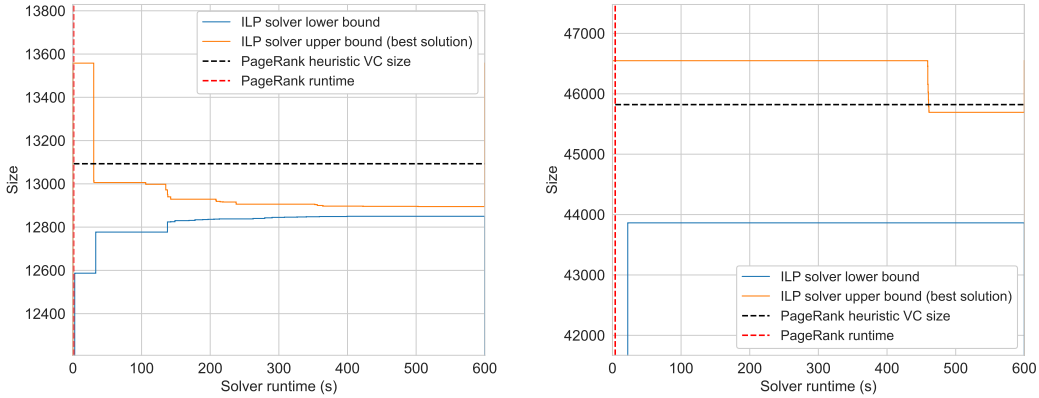


Figure 32: Progression of the upper and lower bounds obtained by the Gurobi ILP solver when applied to the MVCP on the *L. braziliensis* (left graph) and *L. donovani* (right graph) networks. The time taken to find the heuristic solution utilising PageRank, listed in Table 6, is denoted by the red vertical dashed line, and the black horizontal dashed line corresponds to the found vertex cover size.

are presented in Figure 32 and the other results, which are similar to the displayed networks, are attached in the appendix as Figures C1-C3. The convergence plots show that the ILP solvers consistently take significantly more time to find solution candidates of a similar size to the heuristic solutions. However, with the set optimisation time limit, it is difficult to estimate how far from convergence the ILP methods are at this limit, since the optimality gaps can be as large as 4%, as seen in the convergence plot for *L. donovani*.

6 Conclusion

PPINs provide a valuable tool for studying and identifying vital proteins in organisms. However, identifying the most vital proteins is not straightforward and, with our approach, is connected to computationally hard problems: the maximum clique problem and the minimum vertex cover problem. Given the large PPINs and their biological significance, there is a need for robust procedures to identify the most important vertices of a network. Centrality methods are a viable basis for identifying important vertices in networks, and our work also shows that they are viable for constructing heuristic algorithms for the more complex problems. To compare performance, ILP formulations were constructed using Gurobi and SCIP as straightforward exact solution methods. These approaches are first benchmarked, after which appropriately performing methods are run on real PPINs.

Degree centrality and PageRank were considered during the experimental evaluation as the basis for the heuristic algorithms. Degree centrality performed well in all of the heuristic algorithms, while PageRank performed well in the vertex cover heuristic but poorly in the clique heuristic for the PPINs. The other centrality methods, which did not have long runtimes, namely eigenvector and coreness centralities, were also considered as options. However, the analysis of their performance on PPINs is currently inconclusive. Based on initial testing on the PPINs, both eigenvector and coreness centralities perform similarly to degree centrality in terms of the clique and vertex cover sizes found. However, betweenness and coreness centralities were slower than degree centrality or PageRank.

6.1 Possible improvements on heuristics

As discussed in Sections 4.1 and 5.3, seed selection significantly affects the performance of the clique heuristics. Thus, some methods for selecting multiple seeds were considered. The first method utilises approaches similar to those used to calculate the h-index [19] for graphs, but generalised to arbitrary centrality methods. The second method utilises random selection of seed nodes, where the probability of a seed being chosen is proportional to its centrality value. Some pre-emptive runs of these methods were performed, which showed improvements in solution quality without significantly increasing the algorithm's runtime. These runs were left out of the scope of this thesis.

The idea of removing vertices from consideration through k-cores can also be further extended to create disjoint subsets of vertices, called cuts. If the graph has a clique of size k and a cut on the graph has fewer than $k - 1$ edges between the disjoint subsets, then the clique of size k must be fully contained in one of the subsets. This method is not considered further in the scope of this thesis.

Alternatively, for the vertex cover heuristics, some algorithms aim to reduce the size of the vertex cover. An example is the one-k-swap algorithm [42], which has been studied for the MISP. The goal of the algorithm is to swap one vertex from an independent set with at least two vertices that are not in the independent set, thereby increasing the size of the independent set. Due to Theorem 3.3, the algorithm would be almost directly applicable for the vertex covers. These algorithms can be used even

for ILP formulations if they fail to converge in the given optimisation time limit. The effects of the algorithm on the quality of the vertex covers and the heuristic runtimes have not been investigated.

6.2 Further research

Although the graphs consisted mainly of protein pairs with a high probability of being significant, a more thorough consideration of the biological reliability of the alignment process would be warranted. First, this would include considering the threshold limits for e-values and bit scores of protein pairs, as around 3% of protein pairs are below the generally accepted threshold for high quality, even if they are above the level of significance. Since metrics like e-values can be both conservative and liberal in the detection process, it is important to consider both stricter thresholds and how to identify potentially significant pairs excluded by the threshold. False-positive connections within the network are hard to detect with certainty, as this requires either experiments on individual proteins or the development of more sophisticated computational methods to make more accurate predictions. Additionally, the connections predicted by alignment may not directly translate into significant protein interactions, and their reliability should be examined further.

The sets of proteins found with the methods in Section 5 can also be further studied by considering what kinds of protein complexes and functionalities the sets of vital proteins may have. For example, UniProt offers specific keywords for many available proteins which were briefly analysed with the solutions but results were inconclusive and out of scope for the main research questions for this thesis.

A more significant goal of this research, which is likely to follow this thesis, is to modify the process described in Figure 24 to align the proteomes of the *Leishmania* species against the human proteome. This would provide a more robust approach to analysing interactions between humans and parasites, which, in the end, would be a significant contribution to the research. The main issue with applying the analysis method to the human protein database is its size. First, BLAST processing of the whole database takes notably longer than for the *Leishmania* species. Second, the resulting network size is unknown, as it depends heavily on the number of interactions BLAST can find. In worst-case scenarios, the networks can be a lot larger than what *L. donovani* is, which makes the ILP formulations even harder to process, even when pruning strategies are applied. This would necessitate even more heuristic approaches. At this time, this is only speculative, as it is not entirely clear how the properties of the human-*Leishmania* interaction networks would compare to the networks of the *Leishmania* species themselves. If comparisons between the species are done, careful consideration of the network formulation, along with a reconsideration of the performance of the different heuristics, is warranted.

References

[1] Konstantinos A. Theofilatos, Christos M. Dimitrakopoulos, Athanasios K. Tsakalidis, Spyridon D. Likothanassis, Stergios T. Papadimitriou, and Seferina P. Mavroudi. Computational approaches for the prediction of protein-protein interactions: A survey. *Current Bioinformatics*, 6(4):398–414, 2011. ISSN 2212-392X. doi: https://doi.org/10.2174/157489311798072981. URL https://www.benthamdirect.com/content/journals/cbio/10.2174/157489311798072981.

[2] Jeff Alstott, Ed Bullmore, and Dietmar Plenz. powerlaw: A python package for analysis of heavy-tailed distributions. *PLOS ONE*, 9(1):1–11, 01 2014. doi: 10.1371/journal.pone.0085777. URL https://doi.org/10.1371/journal.pone.0085777.

[3] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215 (3):403–410, 1990.

[4] Jorge Alvar, Iván D. Vélez, Caryn Bern, Mercé Herrero, Philippe Desjeux, Jorge Cano, Jean Jannin, Margriet den Boer, and the WHO Leishmaniasis Control Team. Leishmaniasis worldwide and global estimates of its incidence. *PLOS ONE*, 7(5):1–12, 05 2012. doi: 10.1371/journal.pone.0035671. URL https://doi.org/10.1371/journal.pone.0035671.

[5] Alexiou Athanasios, Vairaktarakis Charalampos, Tsiamis Vasileios, and Ghulam Md. Ashraf. Protein-protein interaction (ppi) network: recent advances in drug discovery. *Current drug metabolism*, 18(1):5–10, 2017.

[6] A. Ayala, A. Llanes, R. Lleonart, and C. M. Restrepo. Advances in leishmania vaccines: Current development and future prospects. *Pathogens*, 13(9):812, 2024. doi: 10.3390/pathogens13090812.

[7] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[8] Alex Bavelas. Communication patterns in task-oriented groups. *The Journal of the Acoustical Society of America*, 22(6):725–730, 11 1950. ISSN 0001-4966. doi: 10.1121/1.1906679. URL https://doi.org/10.1121/1.1906679.

[9] Benjamin Buchfink and Contributors. 1. tutorial — diamond wiki. https://github.com/bbuchfink/diamond/wiki/1.-Tutorial, 2023. Accessed: 2026-02-05.

[10] Thomas Bläsius, Tobias Friedrich, and Maximilian Katzmann. Efficiently approximating vertex cover on scale-free networks with underlying hyperbolic geometry. *Algorithmica*, 85(12):3487–3520, 2023.

[11] Guillaume Blin, Florian Sikora, and Stephane Vialette. Querying graphs in protein-protein interactions networks using feedback vertex set. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 7(4):628–635, 2010. doi: 10.1109/TCBB.2010.53.

[12] Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *The Journal of Mathematical Sociology*, 2(1):113–120, 1972. doi: 10.1080/0022250X.1972.9989806. URL https://doi.org/10.1080/0022250X.1972.9989806.

[13] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1):107–117, 1998. ISSN 0169-7552. doi: https://doi.org/10.1016/S0169-7552(98)00110-X. URL https://www.sciencedirect.com/science/article/pii/S0169755298001 10X. Proceedings of the Seventh International World Wide Web Conference.

[14] Anna D Broido and Aaron Clauset. Scale-free networks are rare. *Nature communications*, 10(1):1017, 2019.

[15] Benjamin Buchfink, Chao Xie, and Daniel H Huson. Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59–60, 2015.

[16] Benjamin Buchfink, Klaus Reuter, and Hajk-Georg Drost. Sensitive protein alignments at tree-of-life scale using diamond. *Nature Methods*, 18:366–368, 2021. doi: 10.1038/s41592-021-01101-x.

[17] Jaya Chakravarty and Shyam Sundar. Current and emerging medications for the treatment of leishmaniasis. *Expert Opinion on Pharmacotherapy*, 20 (10):1251–1265, 2019. doi: 10.1080/14656566.2019.1609940. URL https://doi.org/10.1080/14656566.2019.1609940. PMID: 31063412.

[18] S. Y. Chan, K. Morgan, and J. Ugon. Finding maximum cliques in large networks, 2022. URL https://arxiv.org/abs/2207.13010.

[19] James Cheng, Yiping Ke, Ada Wai-Chee Fu, Jeffrey Xu Yu, and Linhong Zhu. Finding maximal cliques in massive networks. *ACM Trans. Database Syst.*, 36 (4), December 2011. ISSN 0362-5915. doi: 10.1145/2043652.2043654. URL https://doi.org/10.1145/2043652.2043654.

[20] Tien-Ming Cheng, Yu-E Lu, Michele Vendruscolo, Pietro Lio', and Tom L. Blundell. Prediction by graph theoretic measures of structural effects in proteins arising from non-synonymous single nucleotide polymorphisms. *PLoS Computational Biology*, 4(7):e1000135, July 2008. doi: 10.1371/journal.pcbi.1000135.

[21] Supratim Choudhuri. Chapter 6 - sequence alignment and similarity searching in genomic databases: Blast and fasta. In Supratim Choudhuri, editor, *Bioinformatics for Beginners*, pages 133–155. Academic Press, Oxford, 2014. ISBN 978-0-12-410471-6. doi: https://doi.org/10.1016/B978-0-12-410471-6.00006-2. URL https://www.sciencedirect.com/science/article/pii/B9780124104716000062.

[22] Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

[23] Clay Mathematics Institute. The millennium prize problems. https://www.claymath.org/millennium-problems/, 2026. Accessed: February 4, 2026.

[24] UniProt Consortium. Uniprot: a worldwide hub of protein knowledge. *Nucleic acids research*, 47(D1):D506–D515, 2019.

[25] Kousik Das, Sovan Samanta, and Madhumangal Pal. Study on centrality measures in social networks: a survey. *Social network analysis and mining*, 8 (1):13, 2018.

[26] Jan Fassler and Peter Cooper. Blast glossary. In *BLAST® Help [Internet]*. National Center for Biotechnology Information (US), Bethesda (MD), July 2011. Available from: https://www.ncbi.nlm.nih.gov/books/NBK62051/.

[27] Andrés F Flórez, Daeui Park, Jong Bhak, Byoung-Chul Kim, Allan Kuchinsky, John H Morris, Jairo Espinosa, and Carlos Muskus. Protein network prediction and topological analysis in leishmania major as a tool for drug target selection. *BMC bioinformatics*, 11(1):484, 2010.

[28] Linton C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, 40(1):35–41, 1977. ISSN 00380431, 23257938. URL http://www.jstor.org/stable/3033543.

[29] Peter D. Grünwald. Beyond neyman–pearson: E-values enable hypothesis testing with a data-driven alpha. *Proceedings of the National Academy of Sciences*, 121(39):e2302098121, 2024. doi: 10.1073/pnas.2302098121. URL https://www.pnas.org/doi/abs/10.1073/pnas.2302098121.

[30] Gurobi Optimization, LLC. Gurobi optimizer reference manual. https://www.gurobi.com, 2026. Accessed: Februrary 8, 2026.

[31] Takeshi Hase, Hiroshi Tanaka, Yasuhiro Suzuki, So Nakagawa, and Hiroaki Kitano. Structure of protein interaction networks and their implications on drug design. *PLoS computational biology*, 5(10):e1000550, 2009.

[32] Taher Haveliwala, Sepandar Kamvar, Dan Klein, Chris Manning, and Gene Golub. Computing pagerank using power extrapolation. Technical report, Stanford, 2003.

[33] Fereydoun Hormozdiari, Petra Berenbrink, Nataša Pržulj, and S Cenk Sahinalp. Not all scale-free networks are born equal: the role of the seed graph in ppi network evolution. *PLoS computational biology*, 3(7):e118, 2007.

[34] Chengbang Huang, Faruck Morcos, Simon P. Kanaan, Stefan Wuchty, Danny Z. Chen, and Jesus A. Izaguirre. Predicting protein-protein interactions from protein domains using a set cover approach. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):78–87, 2007. doi: 10.1109/TCBB.2007.1001.

[35] Hawoong Jeong, Sean P Mason, A-L Barabási, and Zoltan N Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001.

[36] David S Johnson and Michael A Trick. *Cliques, coloring, and satisfiability: second DIMACS implementation challenge, October 11-13, 1993*, volume 26. American Mathematical Soc., 1996.

[37] Ali Karci, Selman Yakut, and Furkan Öztemiz. A new approach based on centrality value in solving the minimum vertex cover problem: Malatya centrality algorithm. *Computer Science*, 11 2022. doi: 10.53070/bbd.1195501.

[38] Richard M. Karp. *Reducibility among Combinatorial Problems*, pages 85–103. Springer US, Boston, MA, 1972. ISBN 978-1-4684-2001-2. doi: 10.1007/978-1-4684-2001-2_9. URL https://doi.org/10.1007/978-1-4684-2001-2_9.

[39] Leo Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953. ISSN 1860-0980. doi: 10.1007/BF02289026. URL https://doi.org/10.1007/BF02289026.

[40] Raya Khanin and Ernst Wit. How scale-free are biological networks. *Journal of computational biology*, 13(3):810–818, 2006.

[41] Michael Kotlyar, Chiara Pastrello, Francesca Pivetta, et al. In silico prediction of physical protein interactions and characterization of interactome orphans. *Nature Methods*, 12:79–84, 2015. doi: 10.1038/nmeth.3178.

[42] Yu Liu, Jiaheng Lu, Hua Yang, Xiaokui Xiao, and Zhewei Wei. Towards maximum independent sets on massive graphs. *Proceedings of the VLDB Endowment*, 8:2122–2133, 09 2015. doi: 10.14778/2831360.2831366.

[43] Yao-Yu Lu, William Stafford Noble, and Uri Keich. A BLAST from the past: Revisiting blastp's E-value. *Bioinformatics*, 40(12):btae729, November 2024. doi: 10.1093/bioinformatics/btae729.

[44] David W. Matula and Leland L. Beck. Smallest-last ordering and clustering and graph coloring algorithms. *J. ACM*, 30(3):417–427, July 1983. ISSN 0004-5411. doi: 10.1145/2402.322385. URL https://doi.org/10.1145/2402.322385.

[45] Natarajan Meghanathan. A comprehensive analysis of the correlation between maximal clique size and centrality metrics for complex network graphs. *Egyptian Informatics Journal*, 22(3):339–355, 2021. ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2016.06.004. URL https://www.sciencedirect.com/science/article/pii/S1110866516300305.

[46] Xiangmao Meng, Wenkai Li, Xiaoqing Peng, Yaohang Li, and Min Li. Protein interaction networks: centrality, modularity, dynamics, and applications. *Frontiers of Computer Science (print)*, 15, 01 2021. doi: 10.1007/s11704-020-8179-0.

[47] Yoichi Murakami, Lokesh P Tripathi, Philip Prathipati, and Kenji Mizuguchi. Network analysis and in silico prediction of protein–protein interactions with applications in drug discovery. *Current opinion in structural biology*, 44: 134–142, 2017.

[48] National Center for Biotechnology Information. Blast statistics: The expect value. https://www.nlm.nih.gov/ncbi/workshops/2023-08_BLAST_evol/e_value.html, 2023. NCBI Workshop: BLAST Evolution, Accessed: Februrary 5, 2026.

[49] National Center for Biotechnology Information. Fasta format for nucleotide sequences. https://www.ncbi.nlm.nih.gov/genbank/fastaformat/, 2025. NCBI GenBank documentation, Accessed: October 28, 2025.

[50] NetworKit Documentation. Networkit python api: networkit.generators. https://networkit.github.io/dev-docs/python_api/generators.html, 2026. Accessed: Febraury 2, 2026.

[51] Bharath Pattabiraman, Md. Mostofa Ali Patwary, Assefaw H. Gebremedhin, Wei keng Liao, and Alok Choudhary. Fast algorithms for the maximum clique problem on massive graphs with applications to overlapping community detection. *Internet Mathematics*, 11(4-5):421–448, 2015. doi: 10.1080/15427951.2014.986778. URL https://doi.org/10.1080/15427951.2014.986778.

[52] William R. Pearson. An introduction to sequence similarity ("homology") searching. *Current Protocols in Bioinformatics*, 42(1):3.1.1–3.1.8, June 2013. doi: 10.1002/0471250953.bi0301s42.

[53] V Srinivasa Rao, K Srinivas, GN Sujini, and GN Sunand Kumar. Protein-protein interaction detection: methods and analysis. *International journal of proteomics*, 2014(1):147648, 2014.

[54] Venkata Subramanyam Rao, Koppula Srinivas, G. N. Sujini, and G. N. Kumar. Protein–protein interaction detection: Methods and analysis. *International Journal of Proteomics*, 2014:147648, 2014. doi: 10.1155/2014/147648.

[55] Kamal Rawal, Robin Sinha, Bilal Ahmed Abbasi, Amit Chaudhary, Swarsat Kaushik Nath, Priya Kumari, P Preeti, Devansh Saraf, Shachee Singh,

Kartik Mishra, et al. Identification of vaccine targets in pathogens and design of a vaccine using computational approaches. *Scientific reports*, 11(1):17626, 2021.

[56] Trilochan Rout, Anjali Mohapatra, Madhabananda Kar, Sabyasachi Patra, and Dillip Muduly. Centrality measures and their applications in network analysis: Unveiling important elements and their impact. *Procedia Computer Science*, 235:2756–2765, 2024. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2024.04.260. URL https://www.sciencedirect.com/science/article/pii/S1877050924009384. International Conference on Machine Learning and Data Engineering (ICMLDE 2023).

[57] Gert Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966. doi: 10.1007/BF02289527.

[58] Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983. ISSN 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90028-X. URL https://www.sciencedirect.com/science/article/pii/037887338390028X.

[59] Victor Spirin and Leonid A. Mirny. Protein complexes and functional modules in molecular networks. *Proceedings of the National Academy of Sciences*, 100(21):12123–12128, 2003. doi: 10.1073/pnas.2032324100. URL https://www.pnas.org/doi/abs/10.1073/pnas.2032324100.

[60] Michael PH Stumpf and Mason A Porter. Critical truths about power laws. *Science*, 335(6069):665–666, 2012.

[61] Sabine Tornow and HW Mewes. Functional modules by relating protein interaction networks and gene expression. *Nucleic acids research*, 31(21): 6283–6289, 2003.

[62] Onur Ugurlu. New heuristic algorithm for unweighted minimum vertex cover. In *2012 IV International Conference "Problems of Cybernetics and Informatics" (PCI)*, pages 1–4, 2012. doi: 10.1109/ICPCI.2012.6486444.

[63] Alexei Vazquez, Alessandro Flammini, Amos Maritan, and Alessandro Vespignani. Global protein function prediction from protein-protein interaction networks. *Nature biotechnology*, 21(6):697–700, 2003.

[64] Yi Wang, Tao Cui, Cong Zhang, Min Yang, Yuanxia Huang, Weihui Li, Lei Zhang, Chunhui Gao, Yang He, Yuqing Li, et al. Global protein- protein interaction network in the human pathogen mycobacterium tuberculosis h37rv. *Journal of proteome research*, 9(12):6665–6677, 2010.

[65] Stanley Wasserman. Social network analysis: Methods and applications, 1994.

[66] World Health Organization. Leishmaniasis: Key facts. https://www.who.int/news-room/fact-sheets/detail/leishmaniasis, January 2023. Accessed January 12, 2026.

[67] World Health Organization. The global epidemiology of leishmaniasis. https://www.who.int/health-topics/leishmaniasis/the-global-epidemiology-of-leishmaniasis, 2026. Accessed January 8, 2026.

[68] Lei Yang and Xiaojun Tang. Protein–protein interactions prediction based on iterative clique extension with gene ontology filtering. *The Scientific World Journal*, 2014:523634, January 2014. doi: 10.1155/2014/523634.

[69] Lei Yang, Xudong Zhao, and Xianglong Tang. Predicting disease-related proteins based on clique backbone in protein-protein interaction network. *Int J Biol Sci*, 10:677–688, 2014. doi: 10.7150/ijbs.8430. URL https://www.ijbs.com/v10p0677.htm.

[70] Zuse Institute Berlin (ZIB). Scip - solving constraint integer programs. https://scip.zib.de, 2026. Accessed: February 8, 2026.

A Effects of pruning graphs for MCP ILP

Red dashed lines in the figures correspond to the size of the clique achieved with applying Algorithm 1 with degree centrality ordering. This is effectively the rate of reduction in the final procedure.

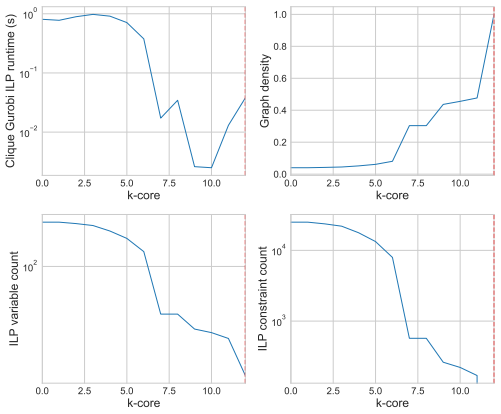


Figure A1: Results for *L. amazonensis*

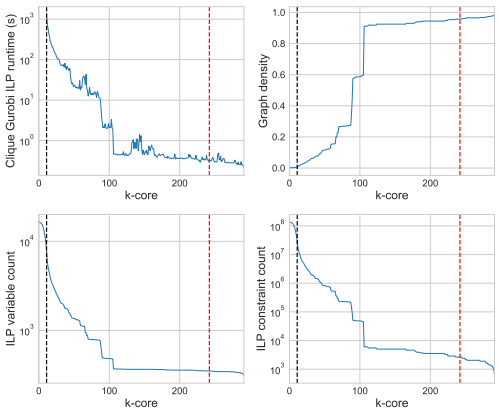


Figure A2: Results for *L. braziliensis*

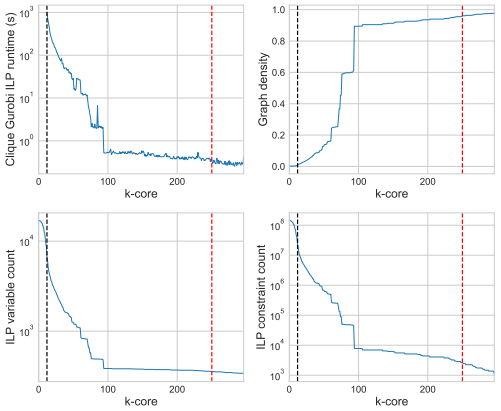


Figure A3: Results for *L. infantum*

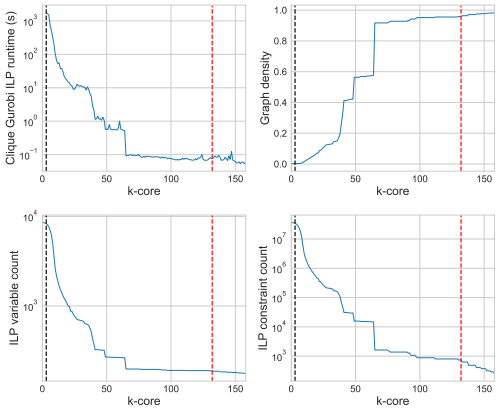


Figure A4: Results for *L. major*

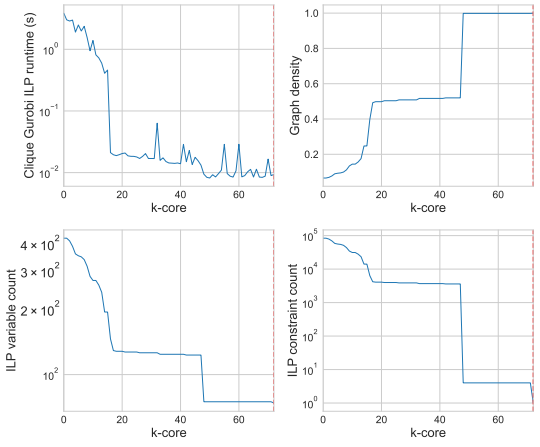


Figure A5: Results for *L. tropica*

B Results of running clique heuristic on PPINs with different seed vertices

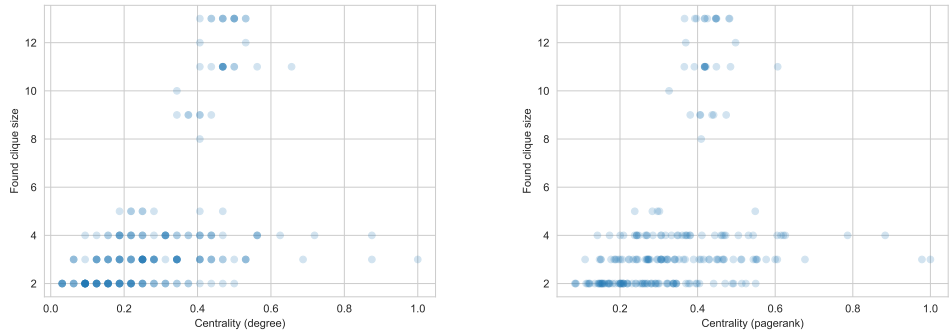


Figure B1: Clique sizes achieved for the graph of *L. amazonensis*. Maximum clique size in the graph is 13.

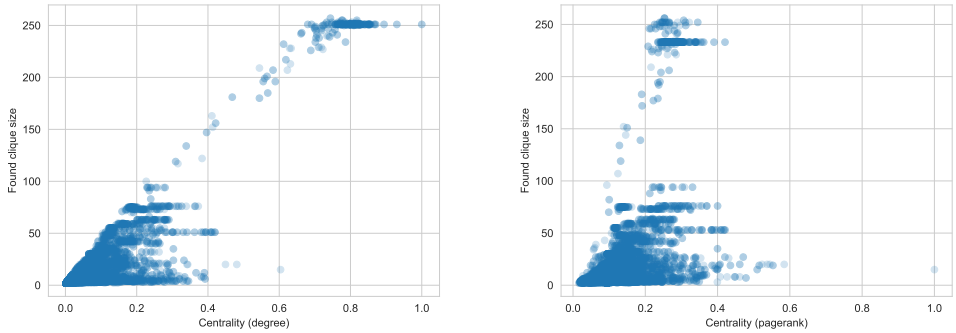


Figure B2: Clique sizes achieved for the graph of *L. infantum*. Maximum clique size in the graph is 259.

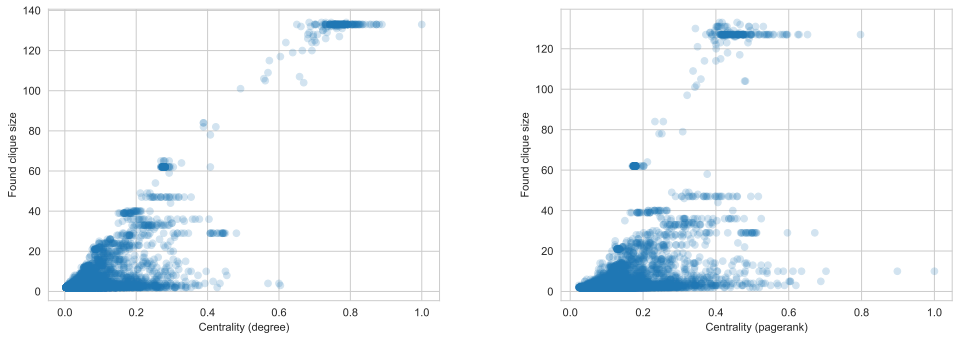


Figure B3: Clique sizes achieved for the graph of *L. major*. Maximum clique size in the graph is 136.

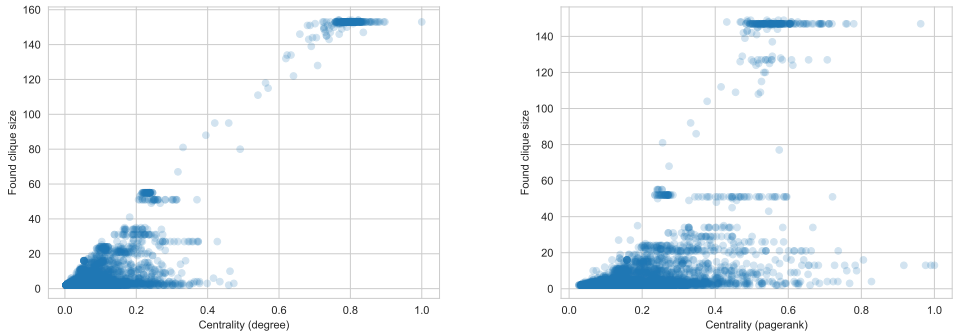


Figure B4: Clique sizes achieved for the graph of *L. mexicana*. Maximum clique size in the graph is 159.

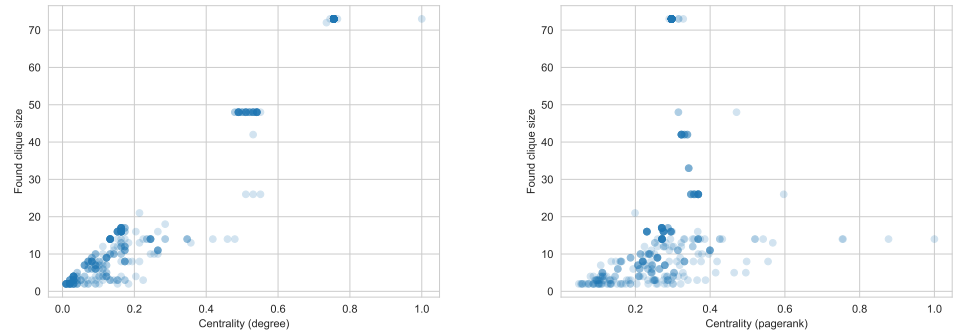


Figure B5: Clique sizes achieved for the graph of *L. tropica*. Maximum clique size in the graph is 73.

C Convergence plots for non-convergent MVCP ILP procedures

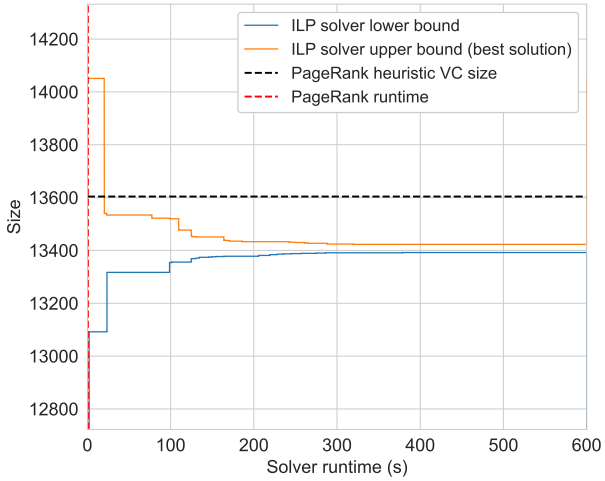


Figure C1: Convergence of Gurobi MVCP ILP for *L. infantum*

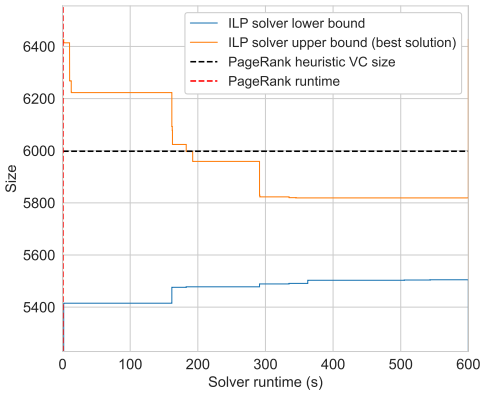


Figure C2: Convergence of Gurobi MVCP ILP for *L. major*

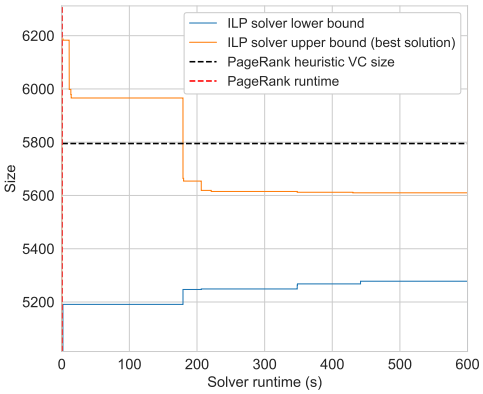


Figure C3: Convergence of Gurobi MVCP ILP for *L. mexicana*